

ADOBE® MEDIA SERVER

Configuration and Administration Guide

Legal notices

For legal notices, see http://help.adobe.com/en_US/legalnotices/index.html.

Contents

Chapter 1: Deploying the server

Configure ports	1
Load balancing	7
Deploying edge servers	8
Deploying 64-bit servers	13

Chapter 2: Configuring the server

Configure the server for virtual hosting	14
Working with configuration files	17
Configuring performance features	20
Configuring security features	33
Performing general configuration tasks	45
Configuring content storage	49
Configuring Apache HTTP Server	53
Configure Apache for Adobe HTTP Dynamic Streaming and Apple HTTP Live Streaming	59
Configure HTTP Streaming failover	63
Using a third-party web server	76
Configuring Differentiated Services (DiffServ)	77

Chapter 3: Using the Administration Console

Connecting to the Administration Console	82
Inspecting applications	84
Manage administrators	91
Managing the server	93

Chapter 4: Monitoring and Managing Log Files

Working with log files	97
Access logs	98
Application logs	105
Dagnostic logs	106

Chapter 5: Administering the server

Start and stop the server	108
Checking server status	109
Checking video files	111
Managing the server on Linux	115
Scramble tool	116

Chapter 6: Using the Administration API

Working with the Administration API	118
---	-----

Chapter 7: XML configuration files reference

Changes to configuration files from 4.5 to 5.0.1	125
Changes to configuration files from 4.0 to 4.5	125

Adaptor.xml file	127
Application.xml file	141
Logger.xml file	188
Server.xml file	194
Users.xml file	234
Vhost.xml file	238
 Chapter 8: Diagnostics Log Messages	
Message IDs in diagnostic logs	256

Chapter 1: Deploying the server

Configure ports

When you install Adobe Media Server, you can accept the default ports on which Adobe Media Server and Adobe Media Administration Server listen. You can also enter new ports. The default ports for Adobe Media Server are 1935 and 80. The default port for Adobe Media Administration Server is 1111.



To test which ports a client can connect to on a Adobe Media Server hosted by Stefan Richter of FlashComGuru, see [FlashComGuru Port Tester](#).



To test which ports a client can connect to on your own Adobe Media Server, use a [port tester](#) developed and hosted by Jake Hilton.

Port requirements

The following table describes ports on which clients must be able to establish connections to the server:

Port number	Protocol	Transport	Description
1935	RTMP/E	TCP	By default, Flash Player and AIR clients make RTMP connections to Adobe Media Server over port 1935 over TCP. To communicate with Adobe Media Server over the RTMP protocol, clients attempt to connect to ports in the following order: 1935, 80 (RTMP), 80 (RTMPT).
1935	RTMFP	UDP	By default, Flash Player and AIR clients make RTMFP connections to Adobe Media Server over port 1935 over UDP.
80	RTMP/E, RTMTP, HTTP	TCP	By default, Flash Player and AIR clients that cannot connect to Adobe Media Server over port 1935 attempt to tunnel over port 80 (RTMPT). If Apache is installed and enabled, HTTP requests made over port 80 are proxied to Apache over port 8134. HTTP Streaming requests to port 80 can hang when proxying to port 8134. Use port 8134 in the request or configure Apache to listen directly on port 80 and configure Adobe Media Server not to listen on port 80.

Port number	Protocol	Transport	Description
19350-65535	RTMFP	UDP	<p>By default, clients use port 1935 and ports 19350-65535 to communicate with Adobe Media Server over the RTMFP protocol. The RTMFP protocol communicates over UDP. Clients connect to the server over 1935 and the server redirects the client to a port between 19350 and 65535.</p> <p>Allow inbound UDP traffic on 1935 and outbound traffic on ports 19350 - 65535.</p> <p>Allowing UDP inbound traffic from unknown hosts on ports 19350 - 65535 is optional. Adobe Media Server sends a packet out to a new client to redirect it to a port in the range 19350 - 65535. This packet attempts to punch a hole to allow the client traffic back in over the correct port. To use server-side RTMFP NetConnections, allow outbound UDP traffic on all ports.</p>
8134	HTTP	TCP	<p>Adobe Media Server proxies HTTP requests to Apache HTTP Server over port 8134.</p> <p>HTTP Streaming requests to port 80 can hang when proxying to port 8134. Use port 8134 in the request or configure Apache to listen directly on port 80 and configure Adobe Media Server not to listen on port 80.</p>
1111	HTTP, RTMP	TCP	<p>By default, Flash Player, AIR, and HTML clients connect to Adobe Media Administration Server over port 1111. Clients must specify port 1111 in connection URLs.</p> <p>Clients cannot access Adobe Media Administration Server over RTMPS. As a result, Adobe recommends that you block all external access to port 1111 so that access to the Administration Server is restricted to clients that are inside your firewall.</p>
443	RTMPS	TCP	<p>The default port for RTMPS. RTMPS adheres to SSL standards for secure network connections and enables connections through a TCP socket on a secure port.</p>

Configure IP addresses and ports

Use the `ams.ini` file to configure the IP addresses and ports on which Adobe Media Server listens for requests.

- 1 Open `rootinstall/conf/ams.ini` in a text editor.
- 2 Edit the `ADAPTOR.HOSTPORT` parameter. The default ports are:

```
ADAPTOR.HOSTPORT = :1935,80
```

- 3 Save the file and restart the server.

The `Adaptor.xml` configuration file uses the `ADAPTOR.HOSTPORT` parameter in the `Adaptor/HostPortList/HostPort` element:

```
<HostPort name="edge1" ctl_channel="localhost:19350"
rtmfp=":${ADAPTOR.HOSTPORT}">${ADAPTOR.HOSTPORT}</HostPort>
```

See also

[“Start and stop the server” on page 108](#)

[“Add an adaptor” on page 15](#)

Configure IP addresses and ports for RTMFP

RTMFP connection flow

- 1 An RTMFP client connects to a Adobe Media Server `amsedge` process over UDP port 1935.

Specify this port in the `ADAPTOR.HOSTPORT` parameter of the `ams.ini` file. To change this value, see [“Configure IP addresses and ports for RTMFP” on page 3](#).

Important: *RTMFP and RTMP/E clients use the same port to connect to Adobe Media Server. However, RTMFP clients use UDP and RTMP/E clients use TCP.*

- 2 The `amsedge` process redirects the connection to an `amscore` process listening on a UDP port in the range 19350-65535.

Each `amscore` process has its own RTMFP listener. Each RTMFP listener binds to one UDP port in the range. When an `amscore` process starts, the listener binds to the next available UDP port in the range specified in the `Adaptor/RTMFP/Core/HostPortList/HostPort` element of the `Adaptor.xml` configuration file. For example, if there are three `amscore` processes, they listen on ports 19350, 19351, and 19352. These ports are called the *RTMFP redirect ports* or the *RTMFP migration ports*.

The number of ports in use depends on the number of `amscore` processes in use. The number of `amscore` processes in use depends on how application instances are distributed. See [“Configure how applications are assigned to server processes” on page 28](#).

When the server redirects the client, it sends the client an IP address and port number. If the server is behind a NAT, specify the public IP address of the server in the `public` attribute of the `HostPort` element. The server passes this address to the client and the client uses it to connect to the `amscore` process.

Configure RTMFP redirect ports

- 1 Open `rootinstall/conf/_defaultRoot_/Adaptor.xml` in a text editor.
- 2 Edit the value of the `HostPort` element. The default ports are as follows:

```
<Adaptor>
  ...
  <RTMFP>
    ...
    <Core>
      <HostPortList>
        <HostPort>:19350-65535</HostPort>
      </HostPortList>
    </Core>
  </RTMFP>
</Adaptor>
```

- 3 If the RTMFP adaptor is behind a NAT, specify the in-front-of-NAT IP address that clients connect to in the `public` attribute of the `HostPort` element. The following example uses 12.34.56.78 for the in-front-of-NAT IP address:

```
<HostPort public="12.34.56.78:19350-65535">:19350-65535</HostPort>
```

To redirect the client to an `amscore` process, the `amsedge` process sends the client an IP address and a port number. The server knows the behind the NAT IP address, but the client can't connect to that. The client needs to know the in-front-of NAT IP address. To tell the server its in-front-of NAT IP address, specify it in the `public` attribute.

4 Save the file and restart the server. See [Start and stop the server](#)

About the HostPort element

The value of the `HostPort` element has the following format:

```
<value-of-HostPort> := [ <host-port-range> [ ; <host-port-range> [ ; ... ] ] ]  
<host-port-range> := [ <host> ] [ : <port-range> [ , <port-range> ] ]  
<port-range> := <start-port> [ - <end-port> ]
```

In this example of `HostPortList`, each core listens to two ports: one port from either `host1:2000-2010` or `host2:3000-3010` and one port from either `host2:5000` or `host2:3010-4000`:

```
<HostPortList>  
  <HostPort public="12.34.56.78:2000-2010">host1:2000-2010; host2:3000-3010</HostPort>  
  <HostPort>host2:5000; host2:3010-4000</HostPort>  
</HostPortList>
```

Configure a public IP address if the server is behind a NAT

If an RTMFP adaptor is behind a NAT, specify the in-front-of-NAT IP address that clients connect to in the `public` attribute of the `HostPort` element:

```
<HostPort public="<in-front-of-NAT-server-IP>:19350-65535">:19350-65535</HostPort>
```

If you don't specify the `public` attribute, the server doesn't know its in-front-of-NAT-address. The `amsedge` process can redirect the client to the correct port (for example, 19351) but it tells the client the behind-NAT address, which the client can't contact.

Each `HostPort` element can specify a public address that corresponds to the specified port. This is the address that is advertised to clients for the given `HostPort`. To advertise an address, specify a value for the `public` attribute of the `HostPort` element. The `public` attribute has the same format as the `HostPort` element. The number of ports specified by the `public` attribute must equal the number of ports specified by the `HostPort` element. If the core listens on the *n*-th port of the `HostPort` value, the *n*-th port of the `public` attribute is advertised as its value.

In this example of `HostPortList` with a `public` attribute, if a core listens on `host1:1005`, its publicly advertised address is `host2:4005`.

```
<HostPortList>  
  <HostPort public="host2:4000-5000">host1:1000-2000</HostPort>  
</HostPortList>
```

More Help topics

[“Add an adaptor”](#) on page 15

NAT and firewall traversal

NAT (network address translation) and firewall filtering can block peer-to-peer connections. In an intranet application, in which you have control over the entire network, do the following to ensure that clients can create peer-to-peer connections:

- Allow UDP traffic through any firewalls
- Use a NAT or firewall that complies with the NAT implementation recommendations of the IETF BEHAVE working group.
- Use the TURN proxy support in Flash Player to send traffic to a proxy in a DMZ that can comply with the previous recommendations. See [Best practices for real-time collaboration using Adobe Media Server](#).

In an Internet application, the application developer must choose how to handle cases in which a firewall or NAT blocks a direct peer-to-peer connection. To create an application that works for connections that aren't peer-to-peer, create a protocol fallback to client-server RTMP and/or RTMPT. To create an application that never relays media through the server (even though some clients may not see the media), don't create a protocol fallback.

Understanding types of NAT

It's important to understand the different types of NAT behavior:

Cone Reuses the same address and port when talking to all peers.

Multiple IP address, symmetric Picks a new address and port when talking to a new peer.

Single IP address, symmetric Uses the same address but a new port when talking to a new peer.

It's also important to understand the filtering behaviors of NATs and firewalls:

None A cone NAT with no filtering is called "full cone".

Address-restricted The peer is restricted to talking only to addresses it has talked to already.

Address and port-restricted The peer is restricted to talking only to addresses and ports it has talked to already.

In addition, some NAT and firewall behaviors aren't easily defined. For example, a NAT could act as a symmetric NAT that preserves port numbers. When it runs out of resources, it could switch and act as a cone NAT.

In another example, a NAT could act as one type of NAT for the first client that connected to a server. It could act as a different type of NAT for the second client that tried to connect to the same server. In this case, a simple analysis can fail to predict whether a client can make a peer-to-peer connection.

Note: A firewall can filter and not be a NAT. A NAT can act as a firewall and have filtering. A firewall may block UDP completely.

RTMFP Connectivity checker

RTMFP inventor Matthew Kaufman hosts a website called RTMFP Connectivity Checker at <http://cc.rtmfp.net/>. Use this site to try to determine whether a client on a particular network can create a peer-to-peer connection.

If the connection to cc.rtmfp.net has the same properties as the connection to and from the peer, use the results to determine whether a peer-to-peer connection can be formed. The results are not always definitive because, in some cases, the connection does not have the same properties. For example, two peers in the same organization behind the same firewall can have different properties between each other than they each have to http://cc.rtmfp.net.

To test a connection, the client connection must have the same properties as the client you're testing, and the NAT or firewall must have predictable behavior. The following are the RTMFP Connectivity Checker tests:

Test	Description
Knows public IP address of self	Flash Player has a local address that matches the address that cc.rtmfp.net saw when the connection came in. In this case, there is no address translation.
Public UDP port number same as local UDP port number	Flash Player's idea of which UDP port number it is using matches what http://cc.rtmfp.net saw when the connection came in. In this case, there is no port translation. If this answer and the previous answer are "Yes", there probably isn't a NAT (but there may be a firewall).
Can receive from same IP address, same UDP port number	This value is always "Yes", because if a client couldn't complete this test, it couldn't establish the initial connection.

Test	Description
Can receive from same IP address, different UDP port number	Indicates whether your firewall is “port restricted”. A port restricted firewall requires an outbound connection to the same address and port number before inbound traffic is permitted from that address and port number. This requirement is true even when previous traffic was sent to the same address but different port number.
Can receive from different IP address, different UDP port number	Indicates whether your firewall is “address restricted”. An address restricted firewall requires that an outbound connection is made to a new IP address before inbound traffic is permitted from that IP address.
Can send to different IP address after server introduction	This value is always be “Yes” if the initial connection can be made. This test is like opening a new RTMFP connection. If this test fails, there's a problem with how Flash Player received or treated the introduction request, or the firewall is unpredictable.
Source IP address is preserved from original connection	This test means that you have one of the following: a cone NAT, a symmetric NAT with only one IP address, or a symmetric NAT with multiple IP addresses but the same address happened to be used this time. If repeated tests cause the value to change, you have a symmetric NAT with multiple IP addresses, and sometimes you happen to use the same address.
Source UDP port number is preserved from original connection	This test means that you have a cone NAT. If the value is “No”, you have a symmetric NAT.

Understanding the RTMFP Connectivity test

In some cases, symmetric NATs break peer-to-peer connectivity.

Flash Player can work with most cone NAT configurations and many firewall configurations. (There are some issues with multiple layers of NAT and lack of “hairpinning” support.) However, symmetric NAT in combination with certain firewall or NAT cases at the other end blocks the ability to establish a peer-to-peer connection. If one end is a symmetric NAT with a single IP address, Flash Player cannot connect to peers behind other symmetric NATs or behind port-restricted cone NATs (or port-restricted firewalls).

If one end of a connection is a symmetric NAT with multiple IP addresses, connections to peers behind other symmetric NATs or behind address-restricted (and probably port-restricted) cone NATs (or address-restricted or port-restricted firewalls) are impossible. No matter what Flash Player tries to do to “punch a hole” through the restricted cone NAT or restricted firewall to let the other peer through, the other end moves to a new address and/or port number that doesn't match. The hole that was created is no longer applicable.

Configure ports for HTTP streaming

By default, Adobe Media Server is configured to listen on port 80. Adobe Media Server proxies HTTP traffic to Apache HTTP Server over port 8134. However, HTTP Dynamic Streaming and HTTP Live Streaming connections can hang when proxying through the server.

Do not proxy HDS and HLS traffic through Adobe Media Server to Apache HTTP Server. Either specify the port number in the request URL, or configure Apache to use port 80 and configure Adobe Media Server not to use port 80. You do not need to use both techniques.

Specify the port number in request URLs

- ❖ Connect clients to Apache HTTP Server directly through its own port (the default is port 8134). For example, use the following request URL:

http://ams.example.com:8134/hds-vod/somefile.f4v.f4m

Configure Apache to use port 80 and configure Adobe Media Server not to use port 80

- 1 Configure Apache to use port 80.
 - a Open `rootinstall/Adobe2.2/conf` in a text editor.
 - b Change the line `Listen 8134` to `Listen 80`.
 - c Restart Apache. See [Start and stop the server](#).
- 2 Configure Adobe Media Server not to use port 80.
 - a Open `rootinstall/conf/ams.ini` in a text editor.
 - b Remove 80 from the `ADAPTOR.HOSTPORT` parameter so the parameter looks like the following:

```
ADAPTOR.HOSTPORT = :1935
```
 - c Restart Adobe Media Server. See [Start and stop the server](#).

Load balancing

Workflow for deploying servers in a cluster

You can deploy multiple servers behind a load balancer to distribute the client load over multiple servers. Deploying multiple servers enables you to scale an application for more clients and creates redundancy, which eliminates single points of failure. You can deploy any version of the server (Adobe Media Interactive Server, Adobe Media Development Server, or Adobe Media Streaming Server) in a cluster.

- 1 Install Adobe® Flash® Media Server and verify the installation on each computer.

Ensure that you deploy all servers on computers that meet the minimum system requirements.

Note: For cross-platform compatibility, use lowercase names for all folders and applications.
- 2 Configure a load balancer to see the servers hosting Adobe Media Interactive Server or Adobe Media Streaming Server.

Clustering multiple servers behind a load balancer

Add all the servers in the cluster to the pool (*server farm*) in the load balancer. The load balancer distributes traffic among all the servers in the pool. Configure the load balancer to distribute the load in round-robin mode and to monitor over TCP port 1935.

If the server does not have an externally visible IP address, then for HTTP tunnelling to work, you should enable cookies when you deploy servers behind a load balancer. The load balancer checks the cookie and sends requests with this cookie to the same server. Cookies can be enabled in the load balancer or in the `Adaptor.xml` configuration file in the `Adaptor/HTTPTunnel/SetCookie` element.

Note: For tunnelling connections, cookies are currently supported only on Adobe® Flash® Player 9.0.28 or later in Windows only.

Load balancing peer-assisted networking applications

Use one of the following two techniques to load balance a peer-assisted networking applications:

- Distribute introductions across a server-only RTMFP NetGroup.

This technique is new in Adobe Media Server 5. Use a Server-Side ActionScript API to distribute peer lookup requests across multiple servers. For more information, see [Distributing introductions across multiple servers](#).

- Set up an edge-origin configuration.

In the Vhost.xml configuration file on each edge server, set `<Mode>remote</Mode>`. For more information, see [“Configure edge servers”](#) on page 9.

On the client, the NetConnection URL changes from `"rtmfp://ams-server/application"` to `"rtmfp://edge-ams-server/?rtmfp://origin-ams-server/application"`. Alternately, you can configure the `RouteTable` and `RouteEntry` in the Vhost.xml file on the edge server to point to the origin server. In this case, the client URL is still `"rtmfp://ams-server/application"`.

Set up DNS entries to distribute the load across the edge servers. For the server to function properly as an RTMFP introducer, use one origin server per application. For peer lookup to work, clients must be connected to the same server process, but not necessarily to the same application. Peer lookups do not work across origin servers.

However, peerIDs are cached on the edge server which reduces the load on the origin. When clients are connected to the same edge, the introduction can happen at the edge and there is not a lookup call to the origin.

More Help topics

[“Deploying edge servers”](#) on page 8

Deploying edge servers

Workflow for deploying edge servers

Note: Adobe Media Server Standard cannot be configured as an edge server.

By default, the server runs as an origin server. To run the server as an edge server, you must configure an XML file. Typically you would run Adobe Media Server Professional as an origin server on one computer and run Adobe Media Server Professional as an edge server on another computer.

1. Install Adobe Media Server Professional and verify the installation on each computer.

Deploy all edge and origin servers on computers that meet the minimum system requirements. For information about installing and verifying installation, see the *Installation Guide*.

Note: For cross-platform compatibility, use lowercase names for all folders and applications.

2. Configure an edge server and restart.

On the edge server, edit the Vhost.xml file of the virtual host you want to run as an edge server.

3. Verify that the edge server can communicate with the origin server.

The easiest way to verify is to create an explicit connection. Create a SWF file with an explicit connection to the edge server and run the vod or live service.

4. If you're installing multiple edge servers, copy the `Vhost.xml` file to the same directory on each edge server.
5. Verify that each edge server can communicate with the origin server.
6. Place the origin server and those edge servers nearest to it on the same subnet.
7. If you're deploying more than one edge server, configure a load balancer.

Place the load balancer between the clients and the edges. Configure the load balancer to access the pool of edge servers in round-robin mode and to monitor over TCP port 1935. Use the virtual IP (VIP) address of the pool as the IP address in the `RouteEntry` element of each edge server's `Vhost.xml` file. For detailed information on how to configure the `RouteEntry` element, see the comments in the `RouteEntry` element of the default `Vhost.xml` file, or see [RouteEntry](#).

Configure edge servers

To configure the server to run as an edge server, edit the `Vhost.xml` configuration file of the virtual host you want to run as an edge server. The `Vhost.xml` file defines how the edge server connects clients to the origin server.

Note: You can also configure some virtual hosts to run applications locally (as origins), while others run applications remotely (as edges); this is called *mixed mode* or *hybrid mode*.

Configure a virtual host to run as an edge server

- 1 Open the `Vhost.xml` file of the virtual host you want to configure and locate the following code (comments have been removed):

```
<VirtualHost>
  ...
  <Proxy>
    <Mode>remote</Mode>
    <Anonymous>>false</Anonymous>
    <CacheDir enabled="false" useAppName="true"></CacheDir>
    <LocalAddress></LocalAddress>
    <RouteTable protocol="">
      <RouteEntry></RouteEntry>
    </RouteTable>
    <EdgeAutoDiscovery>
      <Enabled>>false</Enabled>
      <AllowOverride>>true</AllowOverride>
      <WaitTime>1000</WaitTime>
    </EdgeAutoDiscovery>
  </Proxy>
</VirtualHost>
```

Note: The default `VHost.xml` file is located in the `RootInstall/conf/_defaultRoot/_defaultVHost_` directory.

- 2 Edit the following XML elements, as needed.

Element	Required/optional	Description
Mode	Required	Enter <code>local</code> to configure the server to run as an origin server. Enter <code>remote</code> to configure the server to run as an edge server.
Anonymous	Optional	A Boolean value specifying whether the edge server connection is implicit (<code>true</code>) or explicit (<code>false</code>). The default value is <code>false</code> .
CacheDir	Optional	Enables or disables the caching of streams to disk, in addition to caching in memory, on an edge server and allows you to specify the cache location. There are two attributes: <code>enabled</code> and <code>useAppName</code> . To enable caching, set the <code>enabled</code> attribute to <code>"true"</code> . When enabled, the server places streams in the <code>RootInstall/cache/appName</code> directory by default. Use the <code>useAppName</code> attribute to specify whether to use the application name as the name of the cache for the application. Vod applications get significant performance gains when caching is enabled.
LocalAddress	Optional	Specifies the local IP address to which to bind a proxy's outgoing connection (the proxy's loopback address). This element allows the administrator to control network traffic by isolating incoming and outgoing traffic to separate network interfaces.
RouteTable	Optional; create a routing table when it is not necessary or desirable for application developers to see the origin server URL or when you want to use implicit connections.	Specifies, in each <code>RouteEntry</code> element, how to route connections from the origin to the edge. There is one attribute, <code>protocol</code> , that indicates the protocol of the outgoing connection. Set this attribute to either <code>"rtmp"</code> or <code>"rtmps"</code> . To override the <code>RouteTable</code> protocol for a specific <code>RouteEntry</code> element, add a <code>protocol</code> attribute to the <code>RouteEntry</code> element you want to change.
RouteEntry	Optional	Each <code>RouteEntry</code> element maps a <code>host/port</code> pair to a different <code>host/port</code> pair. In the following example, connections to <code>host1:port1</code> are routed to <code>host2:port2</code> : <code>host1:port1;host2:port2</code> Typically, <code>host1:port1</code> is your edge server and <code>host2:port2</code> is your origin server. The following example routes connections destined for host "edge" on port 1935 to host "origin" on port 80: <code><RouteEntry>edge:1935;origin:80</RouteEntry></code> You can specify a wildcard character (*) for any host or port. The following example routes connections destined for any host on any port to host "origin" on port 1935: <code><RouteEntry>*:*;origin:1935</RouteEntry></code> You can also specify a wildcard for the host/port to which connections are being routed. The following example routes connections destined for any host on any port to the same host on port 80: <code><RouteEntry>*:*:*:80</RouteEntry></code> To reject connections, you can specify that a <code>host/port</code> combination be routed to null: <code><RouteEntry>edge:80>null</RouteEntry></code> The <code>RouteEntry</code> element has a <code>protocol</code> attribute. This attribute overrides the <code>RouteTable</code> protocol for a specific <code>RouteEntry</code> element. For example, <code>RouteTable</code> may have one <code>RouteEntry</code> element that specifies an encrypted outgoing RTMPS connection and another <code>RouteEntry</code> tag that specifies the regular RTMP connection. If a <code>protocol</code> is not specified, the outgoing connection uses the same protocol as the incoming connection.

3 Validate the XML and save the `Vhost.xml` file.

- 4 Restart the server.

More Help topics

[“Adaptors and virtual hosts”](#) on page 14

Set up caching on edge servers

Streams that are played via an edge server can optionally be cached on that edge server’s hard drive to avoid extra network traffic back to the origin. Edge servers manage their disk caches automatically using a least-recently-used (LRU) scheme.

Edge servers group the cached files on disk into buckets; the files within each bucket are not ordered, but the buckets themselves are ordered by LRU. When an edge server needs to free up disk space, it deletes the entire contents of the least-recently-used buckets. When a file is added, it gets put in the “newest” bucket, and likewise when an existing file is accessed, it is moved to the newest bucket. This maintains the LRU ordering.

A disk cache is divided up into N buckets, but since all newly accessed content goes into the newest bucket, only the newest bucket actually grows in size. Once that bucket reaches K/N bytes, where K is the maximum size of the disk cache, the server triggers a rollover. A rollover means the server deletes the oldest bucket, and creates a new bucket where new content will go.

For example:

- 1 Start with an empty disk cache, configured to have 8 buckets and max size of 80 GB.
- 2 All content goes into bucket “00” until it grows to 10 GB.
- 3 New bucket “01” is created. All content now goes into this bucket. In addition, content accessed from bucket “00” may be moved to bucket “01”.
- 4 When bucket “01” reaches 10 GB, the server creates bucket “02”, and so on.
- 5 Repeat until there are 8 buckets, “00” through “07”.
- 6 When bucket “07” reaches 10 GB, the server creates bucket “08”, and also deletes bucket “00” and all the content it contains.

The deletion is only triggered by the size of the “new” bucket, not the total size of the disk. Because content can be moved from one of the older buckets into the newest bucket, the older buckets do not remain at their maximum size of N/K bytes, and thus the total size of the disk will be less than its max K when the server starts deleting old buckets.

You set up caching on an edge server in the `CacheDir` element of the `Vhost.xml` file. The following table describes the subelements that you use to set up caching:

Element	Description
MaxSize	Specifies the maximum allowed size of the disk cache, in gigabytes. The server does LRU (least recently used) cleanup of the cache to keep it under the maximum size. The default value is 32 gigabytes. A value of 0 disables the disk cache. A value of -1 specifies no maximum.
NumBuckets	Specifies the number of buckets to divide the cache into. The aggregate cache size is defined by MaxSize. Can be any value from 2 to 128; the default value is 8. More buckets mean that a smaller portion of the disk content will be deleted at any one time, but also that files will need to be moved to new buckets more often, which results in more disk activity and lower performance. You can define the number of buckets that contain contents that can be moved with NumBucketsAtRisk.
NumBucketsAtRisk	Specifies the number of buckets considered to be "at risk" of deletion. Can be any value from 0 to NumBuckets - 1; the default is NumBuckets/2. To avoid moving files too frequently at the expense of disk performance, only segments that are in the oldest NumBucketsAtRisk buckets will be moved to the newest bucket when accessed. A value of 0 means that segments, once pulled from the origin, are never moved to a newer bucket, effectively turning the cache into a LRU cache. The default value of NumBuckets/2 means that only segments in the "older half" of content will be moved. Segments in the "newer half" of content are more likely to be accessed again before they are deleted, thus making it less important to move them.

Connect to an edge server

There are two types of edge server connections: explicit and implicit (also called *anonymous*).

An *explicit* edge server prefixes its address to the origin server's URL in the client `NetConnection.connect()` call. For example, for applications running on `ams.foo.com`, instead of clients connecting to an application with a connection string such as `rtmp://ams.foo.com/app/inst`, clients are directed through the edge, which prefixes its protocol and hostname to the origin URL, as in the following:

```
rtmp://amsedge.foo.com/?rtmp://ams.foo.com/app/inst
```

An *implicit* edge server does not change or modify the origin server's URL in the client `NetConnection.connect()` call. The identity (the IP address and port number) of the implicit edge is hidden from the application developer.

Create an explicit connection

- ❖ Use the following syntax in a client-side `NetConnection.connect()` call to make an explicit connection to an edge server:

```
rtmp://edge/?rtmp://origin/app
```

A question mark (?) separates the edge's prefix from the main URL. The prefix contains only the protocol, hostname, and optionally the port number. The prefix must always end with a trailing slash.

Create an implicit connection

- 1 In the `Vhost.xml` configuration file, set the `Proxy/Anonymous` element to `true`.

Note: Restart the server after changing the `Vhost.xml` file.

- 2 In the `Vhost.xml` file, create a routing table in the `RouteTable` element; for more information, see the comments about `RouteEntry` tags in the `Vhost.xml` file installed with Adobe Media Server.

- 3 Use the following syntax in a client-side `NetConnection.connect()` call to make an implicit connection to an edge server:

```
rtmp://edge/app/appinstance
```

Connect edge servers in a chain


- ❖ You can chain together any number of edges when you make connections to the origin server. Use the following syntax to chain two explicit edges to direct connection requests to the origin server:

```
rtmp://edge1/?rtmp://edge2/?rtmp://origin/app/inst
```

As the connection moves through each edge in the chain, the server consumes the first token in the string. For example, after making the connection to `edge1`, the connection string changes to:

```
rtmp://edge2/?rtmp://origin/app/inst
```

Note: You can specify the RTMPT protocol only for the edges, not for the origin.

 When you use URL decoration to chain edges, Flash Player 7 and earlier versions may have problems with shared objects because of the embedded question mark (?) character in the URL. Call the following function to encode or escape the question marks from the URL before passing the URL to the shared object:

```
function escapeURI(uri) {  
    index = uri.indexOf('?');  
    if (index == -1) return uri;  
    prefix = uri.substring(0, index);  
    uri = uri.substring(index);  
    return prefix += escape(uri);  
}
```

Deploying 64-bit servers

Adobe Media Server installs on 32-bit and 64-bit operating systems. The 64-bit server uses addressable memory space greater than 4GB. This enables Adobe Media Server to use a larger file cache which in turn reduces disk IO.

To deploy 64-bit servers, know the following:

- You can use any combination of 64-bit and 32-bit edge and origin servers in a network topology.
- To serve on-demand media at the highest possible performance, increase the size of the recorded media cache.
- If you have created custom C++ plug-ins for a 32-bit system, they do not work with a 64-bit system. To use 32-bit plug-ins on a 64-bit server, recompile the plug-in with the 64-bit libraries.

More Help topics

[“Configure a 64-bit server”](#) on page 21

Chapter 2: Configuring the server

Configure the server for virtual hosting

Adaptors and virtual hosts

To host multiple tenants on a server, configure the server so that each customer has their own section. Administrators can configure their own sections to serve their content most effectively. You can configure virtual hosting in several ways, depending on your needs.

The server is divided into hierarchical levels: server, adaptor, virtual host (also called *vhost*), and application. The server is at the top level and contains one or more adaptors. Each adaptor contains one or more virtual hosts. Each virtual host hosts one or more applications. Each application has one or more instances. You can add adaptors and virtual hosts to organize the server for hosting multiple applications and sites.

Note: For information about registering applications with the server and storing media, see [“Configuring content storage”](#) on page 49.

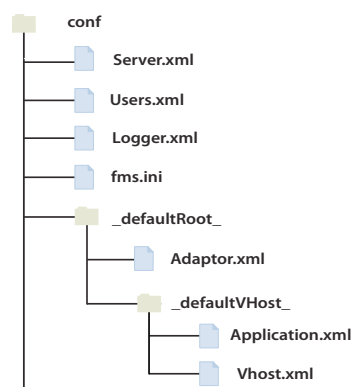
If you’re hosting multiple websites on a server, use virtual hosts to give customers their own root folders. For example, you could use two virtual hosts to host `www.test.com` and `www.example.com` on the same server.

You can assign an IP address or a port number to an adaptor, but not to a virtual host. For this reason, use adaptors to organize virtual hosts by IP address or port number. For example, if a virtual host needs its own IP address to configure SSL, assign it to its own adaptor.

You can also configure one virtual host to run as an edge server and one to run as an origin server. This is called running the server in *hybrid mode*.

Configuration folder structure

Each of these levels—server, adaptor, virtual host, application, and application instances—has distinct configuration settings stored in XML files in the `rootinstall/conf` directory: `Server.xml`, `Adaptor.xml`, `Vhost.xml`, and `Application.xml`. There are also configuration files for information about administrators and logging: `Users.xml` and `Logger.xml`. The most important configuration parameters have been pulled out to the `ams.ini` file, which enables you to use one file to configure the server.



Default structure of the server's configuration (`conf`) directory

Edit any of these XML files in a text or XML editor and restart the server for the changes to take effect. If you modify `Users.xml` or `ams.ini`, you also must restart Adobe Media Administration Server. For more information, see [“Working with configuration files”](#) on page 17.

The following rules define the `conf` directory structure:

- The root configuration folder is `rootinstall/conf`. You cannot remove or modify the name of this directory. The server must have a `Server.xml` file, a `Logger.xml` file, and a `Users.xml` file in the `conf` directory.
- The server has one initialization file, `ams.ini`, in the `rootinstall/conf` directory. This file contains commonly used settings.
- The default adaptor’s root directory is `rootinstall/conf/_defaultRoot_`. You cannot remove or modify the name of this directory. Each adaptor must have an `Adaptor.xml` file in its root directory.
- The default virtual host’s root directory is `rootinstall/conf/_defaultRoot_/_defaultVHost_`. You cannot remove or modify the name of this directory. Each virtual host must have a `Vhost.xml` file in its root directory. Each adaptor must have a default virtual host.
- Virtual host directories may also contain an `Application.xml` file that serves as a default to all applications in that virtual host and a `Users.xml` file that contains information about administrators of that virtual host.
- You may place an `Application.xml` file in an application’s registered directory to create an application-specific configuration. For more information about registered application directories, see the *Developer Guide*.

Add an adaptor

- 1 Create a new directory with the name of the adaptor in the `rootinstall/conf` folder; for example, `rootinstall/conf/adaptor2`.
- 2 In the `adaptor2` directory, create or paste a copy of the `_defaultVHost_` directory and an `Adaptor.xml` file. Each adaptor directory must contain a `_defaultVHost_` directory and an `Adaptor.xml` file.
- 3 In the `_defaultVHost_` directory, create or paste a copy of an `Application.xml` file and a `Vhost.xml` file.
- 4 In the `Adaptor.xml` file in the adaptor directory, add a `HostPort` element to listen on a new port for this adaptor:

```
<HostPort name="edge2" ctl_channel=":19351" rtmfp=":1936">:1936</HostPort>
```

The `name`, `ctl_channel`, and `rtmfp` attribute values must be unique on the server. The `rtmfp` attribute and the `HostPort` element specify the ports to which an IP address should bind. If an IP address is not specified, the adaptor listens on all available interfaces. The server uses the control channel (`ctl_channel`) attribute internally to communicate between server processes (adding a `HostPort` element creates a new `AMSEdge` process).

The server uses the `HostPort` value to listen for clients—no two adaptors can listen on the same port, either internally or externally, unless they use different IP addresses. If a host has multiple IP addresses, multiple adaptors can listen on port 1935. In addition, the control channels of two adaptors must be different, or they cannot inter-operate. Ensure that the control channels on which separate adaptors listen are different from each other. The following example is for a host with multiple IP addresses:

```
<HostPort name="edge1" ctl_channel=":19350"
rtmfp="xx.xx.xx.xx:1935">xx.xx.xx.xx:1935</HostPort>
```

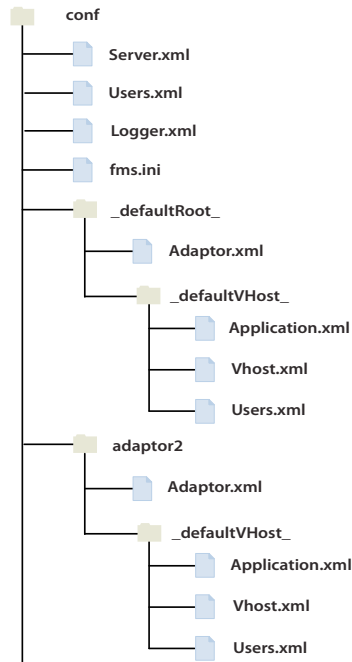
```
<HostPort name="edge2" ctl_channel=":19351"
rtmfp="yy.yy.yy.yy:1935">yy.yy.yy.yy:1935</HostPort>
```

- 5 If you’re running peer-assisted networking applications, add an `Adaptor/RTMFP/Core/HostPortList/HostPort` element. See [“Configure IP addresses and ports for RTMFP”](#) on page 3.
- 6 Restart the server.

- 7 To log in to the Administration Console on the new adaptor, use the syntax *adaptorname/username* in the Username box; for example, *adaptor2/admin*.

For information about logging in to the Administration Console, see “[Connecting to the Administration Console](#)” on page 82.

Administrators are defined in the `UserList` section of the `Users.xml` file. Administrators are either server-level users (similar to a root user) or virtual host-level users. If you log in to an adaptor other than the default adaptor, you are considered a virtual host administrator and don’t have privileges to manage the server or users.



The `conf` directory with an additional adaptor called `adaptor2`

Add a virtual host

- 1 Create a folder with the name of the virtual host in an adaptor folder, for example, `rootinstall/conf/_defaultRoot_/www.example.com`.
- 2 Copy an `Application.xml` file, a `Vhost.xml` file, and a `Users.xml` file to the new virtual host folder. (The `Users.xml` file is required only if you are defining administrators for this virtual host.)
- 3 In the `Vhost.xml` file, specify an application directory in the `AppsDir` element, for example:

```
<AppsDir>C:\www.example.com\AppsDir</AppsDir>
```

Note: It is possible to use the same applications directory for multiple virtual hosts, but it causes namespace conflicts and is not recommended.

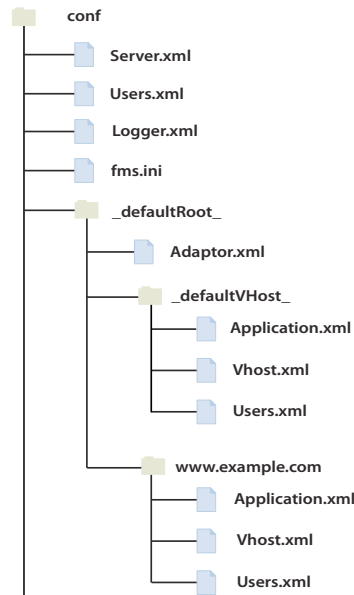
- 4 Validate the XML and save the `Vhost.xml` file.
- 5 Restart the server.

Note: You can call the `startVHost()` Administration API or log in to the Administration Console without restarting the server.

- 6 Log in to the Administration Console.

For information about logging in to the Administration Console, see “[Connecting to the Administration Console](#)” on page 82.

- 7 Connect to the new virtual host by specifying the virtual host name, for example, `www.example.com`, in the Server name field.
- 8 Connect a test application to the new virtual host to make sure it works.



The `conf` directory with an additional virtual host called `www.example.com`

Configuring a single application

You can place an `Application.xml` file in an application’s folder. Values set in the application-level `Application.xml` file override the values in the vhost-level `Application.xml` file.

To see an example, look at the vod and live applications included with Adobe Media Server. Navigate to `rootinstall\applications\vod` and `rootinstall\applications\live`. Each folder contains an `Application.xml` file. The values in these files override the values in the vhost-level file `rootinstall\conf_defaultRoot__defaultVHost_Application.xml`.

To prevent settings in the vhost-level `Application.xml` settings from being overridden by the application-level `Application.xml` file, add an `override` attribute to a tag and set it to `"no"`, as in the following:

```
<LoadOnStartup override="no">false</LoadOnStartup>
```

When `override="no"` for an element in a vhost-level `Application.xml` file, you cannot override that element, or any elements contained within that element, in an application-level `Application.xml` file.

Working with configuration files

Edit a configuration file

Note: Configuration files are located in the `rootinstall/conf` folder. For information about configuration file names, locations, and hierarchy, see “[Configuration folder structure](#)” on page 14.

To edit a configuration file, including `ams.ini`, do the following:

- 1 Open the file in a text editor.
- 2 Edit the file.
- 3 Save the file.
- 4 Validate the XML.
- 5 Restart Adobe Media Server.

If you modify the `Users.xml` file, restart Adobe Media Administration Server, too.

Editing parameters in the `ams.ini` file

The file `rootinstall/config/ams.ini` is the Adobe Media Server INI file. An INI file is a default configuration file. This file contains the most commonly edited configuration parameters. Edit the `ams.ini` file to change the server admin username and password, the adaptor host port, the folder that holds applications, the folder that holds media files, and so on.

The `ams.ini` file contains a list of parameters and their values, as in the following:

```
SERVER.ADMIN_USERNAME = admin
```

The XML configuration files use these parameters. The following is an excerpt from the `User.xml` file:

```
<UserList>
  <!-- This tag defines an administrator for the server. -->
  <User name="{SERVER.ADMIN_USERNAME}">
    <!-- Salted Password Hash for this vhost administrator. -->
    <Password encrypt="true">6cb340fd77d3297cb6d82f57bb085a13d
f45cd9513d042355a9a304c1d47ec433c97a8bdc2584424</Password>
```

When the server starts, it replaces the parameter in the XML configuration files with the value from the `ams.ini` file.

Edit the `ams.ini` file

- 1 Open `rootinstall/conf/ams.ini` in a text editor.
- 2 Save a copy to another location as a backup.
- 3 Enter a new value for a parameter.
- 4 Save the file.
- 5 Restart Adobe Media Server. When you edit a value in the `User.xml` file, restart Adobe Media Administration Server.
- 6 Open the Administration Console and log in with your new password.

Using symbols in configuration files

To simplify configuration, you can use symbols as values for XML elements in configuration files. Create a file named `substitution.xml` in the `rootinstall/conf` folder that maps the symbols to strings that the server substitutes when it reads the configuration files. After you've set up a map file, future updates are faster: you can edit the map file instead of editing each configuration file.

The installer defines a few mappings during the installation process and stores them in the `ams.ini` file. When the server starts, it looks for the `ams.ini` file and the `substitution.xml` file in the `rootinstall/conf` directory. You can also create additional map files and reference them from the `substitution.xml` file.

The server has two predefined symbols, `ROOT` and `CONF`, that are always available. The `ROOT` symbol is mapped to the location of the `AMSMaster.exe` file and the `CONF` symbol is mapped to the location of the `Server.xml` file.

The server builds the symbol map in the following order:

- 1 The predefined symbols `ROOT` and `CONF` are evaluated.
- 2 The `ams.ini` file is evaluated.
- 3 If the `substitution.xml` file exists, the server looks for the `Symbols` tag and processes the child tags in the order in which they appear.
- 4 The server processes the additional map files in the order in which they appear (in `KeyValueFile` elements in the `substitution.xml` file).
- 5 Symbols defined in external map files are processed in the order in which they appear in each file.

Create a substitution.xml file:

- 1 Create a new XML file and save it in the `rootinstall/conf` folder as `substitution.xml`.
- 2 Enter the following XML structure:

```
<Root>
  <Symbols>
    <SymbolName>StringToMapTo</SymbolName>
  </Symbols>
</Root>
```

Add a `SymbolName` element for each symbol you want to create.

- 3 For example, this `substitution.xml` file maps the symbol `TESTUSERNAME` to the value `janedoe`:

```
<Root>
  <Symbols>
    <TESTUSERNAME>janedoe</TESTUSERNAME>
  </Symbols>
</Root>
```

- 4 Open the `rootinstall/conf/Users.xml` file in a text editor.
- 5 Locate the line `<User name="{SERVER.ADMIN_USERNAME}">` and replace the symbol `SERVER.ADMIN_USERNAME` with the symbol `TESTUSERNAME`.

When the server reads the XML file, it substitutes the value from the `substitution.xml` file as follows:

```
<User name="janedoe">
```

Note: Because this symbol is used as an attribute, it is surrounded by quotation marks. If the symbol were used as a regular value, it would not be surrounded by quotation marks.

- 6 Restart the Administration Server.

Note: If you change the `Users.xml` file, you must restart the Administration Server. If you change any other XML configuration file, you must restart the server.

Creating additional map files

You can specify all of your text substitution mappings under the `Symbols` tag in `substitution.xml`. However, you can also create additional map files. To do this, create one or more `KeyValueFile` elements in the `substitution.xml` file. Each element can hold the location of one external file.

For example, the following references the file `C:\testfiles\mySymbols.txt`:

```
<Root>
  <KeyValueFile>C:\testfiles\mySymbols.txt</KeyValueFile>
</Root>
```

These external files are not in XML format. They simply contain a collection of symbol-value pairs, where each pair appears on a separate line and takes the following form:

```
symbol=value
```

The following example shows three symbol-value pairs:

```
USER_NAME=foo
USER_PSWD = bar
HELLO= "world and worlds"
```

Place comments on separate lines that begin with a number sign (#). Do not place comments on the same line as a symbol definition.

The first equal sign (=) in a line is considered the delimiter that separates the symbol and the value. The server trims leading or trailing white space from both the symbol and the value, but no white space within double quotation marks.

Using environment variables

To refer to an environment variable in one of the XML configuration files, use the name of the environment variable within percent (%) characters. The % characters indicate to the server that the symbol refers to an environment variable and not to a user-defined string.

The syntax for specifying an environment variable as a symbol is `${%ENV_VAR_NAME%}`.

For example, the server maps the following symbol to the `COMPUTERNAME` variable:

```
${%COMPUTERNAME%}
```

When you use an environment variable, you don't have to define it in the `substitution.xml` file.

Configuring performance features

In addition to the performance tips in this section, you can also optimize the underlying Apache web server. For more information, see [Apache Performance Tuning](#).

Configure the server to deliver live media

The following three configuration settings impact scale and latency for live media delivery:

- “[Configure the size of stream chunks](#)” on page 22
- “[Send aggregate messages](#)” on page 23
- “[Combine audio samples](#)” on page 25

Note: *You can't send aggregate messages and combine audio samples in the same application.*

To configure the server to deliver live media, choose your use case:

Large Scale Broadcast (scale is more important than latency) In this use case, you want to reach as many people as possible and are willing to allow some latency. You may also want to keep costs down while still reaching as many people as possible. Send large aggregate messages and configure large stream chunk sizes.

Large Scale Broadcast (low latency is more important than scale) In this use case, you want to keep latency as low as possible while reaching as many people as possible. You are willing to trade some scale for decreased latency. Send smaller aggregate messages. Or, combine audio samples and do not send aggregate messages.

Interactive (lowest latency) Interactive applications require very low latency. Do not send aggregate messages. For the lowest latency, do not combine audio samples.

If data latency is your concern, combine audio samples. Combining audio samples introduces audio latency but doesn't affect the rest of the stream. Aggregating messages introduces latency to the whole stream.

Interactive applications are the most difficult to scale and require more hardware than the large scale broadcast use cases.

Configure the server to deliver on-demand media

To configure the server to deliver on-demand (vod) media, complete the following tasks:

- 1 “[Configure the recorded media cache](#)” on page 21
- 2 “[Configure the size of stream chunks](#)” on page 22

Configure a 64-bit server

64-bit servers can address more physical memory than 32-bit servers. When delivering on-demand video on 64-bit servers, increase the size of the recorded media cache size. Only the amount of RAM available on the computer limits the recorded file cache size.

Note: The system warns that a process is close to the memory limit when the process memory is more than 90% of the system memory.

- ❖ “[Configure the recorded media cache](#)” on page 21

Configuring the server to deliver audio-only media

To deliver media that contains only audio,

- 1 “[Combine audio samples](#)” on page 25
- 2 “[Send aggregate messages](#)” on page 23

Configure the recorded media cache

When a stream is requested from the server, segments of the stream are stored in a cache on the server. As long as the cache has not reached capacity, the server places segments in the cache. Each time a stream attempts to access a segment, the server checks the cache. If the segment is available, the server gives the stream a reference to the segment for playback. If the segment is not available, the server retrieves the segment from its source, inserts it into the cache, and returns a reference to that segment to the stream.

When the cache is full, the server removes unused segments, starting with the least recently used. After removing all unused segments, if there still isn't enough room for a new segment, the server notifies the client that the stream is not available and makes an entry in the core log file.

If you have cache-full events in the core log file, increase the size of the cache or limit the number of streams playing. For more information about the core log file, see “[Monitoring and Managing Log Files](#)” on page 97.

- 1 Open the `rootinstall/conf/ams.ini` file.

2 Edit the `SERVER.FLVCACHE_MAXSIZE` parameter.

This is the maximum size of the cache, in megabytes. The default value is 500. The file cache shares process address space with the core process. Each core process has a separate file cache. For 32-bit processes, it has a limit of 2 GB in Windows and 3 GB in Linux. For 64-bit installations, the limit is greater. For more information, see “[Configure a 64-bit server](#)” on page 21.

The size of the cache limits the number of unique streams the server can publish. To increase the probability that a requested stream will be located in the recorded media cache, increase the value of `SERVER.FLVCACHE_MAXSIZE`. To decrease the amount of memory the server process uses, decrease the value of `SERVER.FLVCACHE_MAXSIZE`. While a large cache size is useful, Adobe recommends that you ensure that your total system memory usage does not exceed the process limit of your OS. Consider memory limits and desired memory and stream performance when utilizing the memory cache.

Note: *Cache settings have no effect on live streams, as live streams do not need or utilize the cache.*

3 Restart the server.

There is no exact way to calculate the value of the cache size because it varies depending on the amount of RAM available and the number of other processes that are running. However, you can follow some general guidelines to approximate the recommended size of the cache.

Out of the total system RAM, some amount will be used by the OS and other non-Adobe Media Server processes:

Amount of RAM available for AMS (R) = RAM size - RAM used by the OS and non-AMS processes

The RAM available for Adobe Media Server is divided among the number of cores started:

Amount of RAM per core (Rc) = Amount of RAM available for AMS (R) / Number of Cores

Out of this, some amount of RAM is used up by the core process, and is not available to the file cache:

Recommended File Cache Size = Amount of RAM per core (Rc) - RAM used by each core process

RAM used by each Adobe Media Server process varies based on load, and there is no exact figure. You can use 512 MB as a ballpark figure.

The following example attempts to determine a reasonable file cache size:

```
RAM size = 8 GB
RAM used by OS and non-AMS processes = 1.2 GB (approximately)
Number of AMS core processes allowed = 3
Amount of RAM available for AMS (R) = 8 - 1.2 = 6.8
Amount of RAM per core (Rc) = 6.8 / 3 = 2.27 GB
Recommended file cache size = 2.27 - 0.5 = 1.76 GB
```

Configure the size of stream chunks

Streams break into chunks as they are written to the network. You can specify the size of a chunk. Large values reduce CPU usage because there are fewer writes. However, large values can delay other content on lower bandwidth connections. The larger the content size and the higher the bandwidth of the receiving connection, the more benefit is gained from larger chunk sizes.

1 Open the `Application.xml` file.

Note: *You can set these values in an `Application.xml` file at the `VHost` level or at the application level. To set the value at the application level, copy an `Application.xml` file to the application’s folder.*

2 In the `Client` element, set the `OutChunkSize` element to a value between 128 and 65536 bytes. The default value is 4096 bytes.

For more information, see “[Application.xml file](#)” on page 141.

3 Restart the server.

Send aggregate messages

Important: Do not send aggregate messages and combine audio samples in the same application.

An aggregate message is a single message that contains a list of submessages. Sending aggregate messages reduces CPU usage and increases server capacity. You can configure applications to deliver aggregate messages to clients running on Flash Player 9.0.60.0 and above. When this setting is disabled, the server breaks up aggregate messages into individual messages before delivering them to clients.

Aggregate messages can be used with live and recorded streams.

An origin server can deliver aggregate messages to an edge server, and an edge server can deliver aggregate messages from the FLV data that is has cached on the disk. An edge server can deliver aggregate messages even if the origin server did not.

To ensure that aggregate messaging is enabled, be sure the following settings are enabled:

- `EnableAggMsgs` in the `Server.xml` file
- `AggregateMessages` in the `Application.xml` file
- `AggregateMessages` in the `Vhost.xml` file (if applicable)

The following table describes these settings, plus other configuration settings that are related to using aggregate messages:

Element	Configuration file	Description
<code>Streams/EnableAggMsgs</code>	<code>Server.xml</code>	Enables the creation of aggregate messages in the FLV module. When this setting is “true”, the FLV module returns aggregate messages when loading segment data. When this setting is “false”, the FLV module returns regular audio, video, and data messages. The default value is “true”.
<code>Streams/MaxAggMsgSize</code>	<code>Server.xml</code>	Determines the maximum size (in bytes) of aggregate messages returned from the FLV module (when they are enabled). The actual message size returned by the module might be slightly larger because the module will not fragment individual messages, but instead includes whole messages until this size is exceeded. The default value is 65536.
<code>Client/AggregateMessages</code>	<code>Application.xml</code>	Enables the delivery of aggregate messages to clients that support them. When this setting is disabled, the server breaks up aggregate messages into individual messages before delivering them to clients. The default value is “true”.

Element	Configuration file	Description
StreamManager/Live/Queue	Application.xml	<p>Queues incoming messages that are published to live streams. This is so that the server can create aggregate messages before transmission to subscribing clients. If queuing is not enabled, aggregate messages are not created.</p> <p>The default value is "true".</p>
StreamManager/Live/Queue/MaxQueueSize	Application.xml	<p>Defines the maximum size (in bytes) that the live queue can grow to before the messages it contains are transmitted.</p> <p>Increasing the size of the queue allows for larger aggregates to be created, which increases the efficiency of the server, but introduces latency in the transmission. Decreasing the queue size reduces latency but is less efficient.</p>
StreamManager/Live/Queue/MaxQueueDelay	Application.xml	<p>Defines the maximum time (in milliseconds) that the live queue delays messages before the messages are transmitted.</p> <p>This setting can be used to ensure that unacceptable amounts of latency are not introduced into the publishing stream.</p> <p>Increasing the delay allows for larger aggregates to be created, which increases the efficiency of the server. Decreasing the delay reduces latency but is less efficient.</p>
StreamManager/Live/Queue/AggregateMessages	Application.xml	<p>Determines whether or not aggregate messages are generated when the queue is flushed.</p> <p>When this setting is enabled, the messages in the queue are grouped together as aggregates to improve the efficiency of transmission.</p> <p>It is possible to enable queuing and disable the generation of aggregate messages, but this does not result in performance benefits.</p> <p>The default value is "true".</p>

Element	Configuration file	Description
StreamManager/Live/Queue/AggregateMessages/MaxAggregateMsgSize	Application.xml	<p>Defines the maximum size of an aggregate message that the server creates. The server can create smaller aggregates, as necessary.</p> <p>The server cannot create messages that are larger than the MaxQueueSize, so this number is generally set to something less than or equal to the configured size of the queue.</p>
Proxy/AggregateMessages	Vhost.xml	<p>Determines whether aggregate messages are delivered from the edge cache when a vhost is configured as an edge proxy.</p> <p>The default value is "false".</p> <p>If the edge server receives aggregate messages from the origin when this setting is disabled, the messages are broken up before being cached.</p>
Proxy/AggregateMessages/MaxAggregateMsgSize	Vhost.xml	<p>Determines the size (in bytes) of aggregate messages returned from the edge cache when aggregate messages are enabled.</p> <p>This setting only applies to messages retrieved from the disk cache. Aggregate messages received directly from the origin server are returned as-is, and therefore their size is determined by the origin server's settings for aggregate message size.</p>

Even when you configure the server and application to use aggregate messages, the following circumstances will prevent them from being delivered:

- The client's Flash Player version is earlier than 9.0.60.0.
- The stream is set to filter audio, filter video, or control FPS.
- The stream requires transcoding.

If any of these conditions are present, Adobe Media Server breaks the aggregated message into its submessages, and the messages are delivered individually.

After you confirm that aggregate messaging is enabled, you can use the settings in the Server.xml file to configure the size of the messages. You can also enable or disable the messages based on the type of stream. For more information, see [“Streams”](#) on page 232.

You can also configure message queuing in the Application.xml file. For more information, see [“Queue”](#) on page 175.

Combine audio samples

Important: Do not send aggregate messages and combine audio samples in the same application.

To handle more connections while broadcasting a live stream, combine audio samples.

- 1 Open the `rootinstall/conf/_defaultRoot/_defaultVHost/Application.xml` file.

Note: You can set these values in an `Application.xml` file at the `VHost` level or at the application level. To set the value at the application level, copy an `Application.xml` file to the application's folder.

- 2 Locate the `StreamManager/Audio/CombineSamples` section of the file and set values the following elements:

Element	Description
Subscribers	If there are more than this number of subscribers to a stream, audio samples are combined. The default value is 8. To increase live streaming capacity, set this value to 1.
LoCPU	If the CPU is lower than this value, audio samples are not combined. The default value is 60. To increase live streaming capacity, set this value to 1.
HiCPU	If the CPU is higher than this value, audio samples are not combined. The default value is 80. To increase live streaming capacity, set this value to 1.
MaxSamples	Combine this many samples into one message. The default value is 4. To increase live streaming capacity, set this value to 8.

3 Restart the server.

Limit connection requests

A high connection rate to the server can negatively impact the experience of users already connected to the server.

1 Locate the following code in the Server.xml configuration file:

```
<Root>
  <Server>
    ...
    <Protocol>
      <RTMP>
        <Edge>
          <MaxConnectionRate>10</MaxConnectionRate>
```

Element	Description	Impact
MaxConnectionRate	<p>The maximum number of incoming connections per second the server accepts, per listener. Listeners are defined in the <code>HostPort</code> element in the <code>Adaptor.xml</code> file. Each port the server is configured to listen on represents a listener. You can set a fractional maximum connection rate, such as 12.5. A value of 0 or -1 disables this feature.</p> <p>The value of this element is a global setting for all listeners. If the element is set to 10 connections per second, each listener has a limit of 10 connections per second. If there are three listeners and the <code>MaxConnectionRate</code> is set to 10, the server imposes a maximum total combined rate of 30 connections per second.</p>	Connections requested at a rate above this value remain in the TCP/IP socket queue and are silently discarded by the operating system whenever the queue becomes too long.

2 Validate the XML and save the XML file.

3 Restart the server.

Close idle connections

Sometimes clients have left an application without the server-end of the socket knowing. This behavior can happen when a router is unplugged or crashes without sending TCP close messages for the sockets it is managing, when a laptop is undocked from a docking station, and so on. To reclaim these resources for new and active clients, the server can close the idle clients.

Adobe Media Server sends a small ping message over sockets that have been quiet for awhile. If the client at the far end has gone away, the server's TCP eventually stops retransmitting that data. When that happens, the socket is shut down and the server can clean up any associated client object, and so on.

Elements in the `Server.xml`, `Vhost.xml`, and `Application.xml` configuration files specify how often the server should check for idle clients. When a client has been idle longer than the maximum idle time (10 minutes, by default), the server sends a status message to the `NetConnection` object (the client) with the `code` property set to `NetConnection.Connect.Idle` followed by `NetConnection.Connect.Closed`. The server closes the client connection to the server and writes an `x-status` code of 432 in the access log. The server also writes a message such as "Client *x* has been idle for *y* seconds" in the core and event logs.

To close idle connections, enable the feature in the `Server.xml` file. Once you enable the feature in the `Server.xml` file, you can disable the feature for individual virtual hosts or individual applications in the `Vhost.xml` files and `Application.xml` files. The values defined in the `Server.xml` configuration file apply to all clients connected to the server, unless the values are defined in the `Vhost.xml` file (the `Vhost.xml` values override the `Server.xml` values). The values defined in the `Application.xml` file override the values defined in the `Vhost.xml` file.

Enable closing idle connections

- 1 Locate the following code in the `Server.xml` file:

```
<AutoCloseIdleClients enable="false">
  <CheckInterval>60</CheckInterval>
  <MaxIdleTime>600</MaxIdleTime>
</AutoCloseIdleClients>
```

- 2 Edit the following elements.

Element	Description	Impact
<code>AutoCloseIdleClients</code>	Set the <code>enable</code> attribute to <code>true</code> to close idle clients. If the <code>enable</code> attribute is omitted or not set to <code>true</code> , the feature is disabled. The default value is <code>false</code> .	
<code>CheckInterval</code>	Specifies the interval, in seconds, at which the server checks for active client connections. The default interval is 60 seconds.	A client is disconnected the first time the server checks for idle connections if the client has exceeded the <code>MaxIdleTime</code> value. A shorter interval results in more reliable disconnection times.
<code>MaxIdleTime</code>	Specifies the maximum idle time allowed, in seconds, before a client is disconnected. If this element is 0 or less, the default idle time is used. The default idle time is 600 seconds (10 minutes).	A low value may cause unneeded disconnections. When you configure this element, consider network latency. An idle time that is less than a typical round trip between the server and the client can result in a disconnection.

Configure settings for virtual hosts

You can disable this feature for a virtual host or specify a different maximum idle time for a virtual host in the `Vhost.xml` file.

- 1 Locate the following code in the `Vhost.xml` file and remove the comments:

```
<VirtualHost>
  <AutoCloseIdleClients enable="false">
    <MaxIdleTime>600</MaxIdleTime>
  </AutoCloseIdleClients>
</VirtualHost>
```

- 2 Edit the following elements.

Element	Description
AutoCloseIdleClients	Disable this feature for an individual virtual host by setting the <code>enable</code> attribute to <code>false</code> . If this element is disabled in <code>Server.xml</code> , the feature is disabled for all virtual hosts, even if you specify <code>true</code> in the <code>Vhost.xml</code> file.
MaxIdleTime	Specifies the maximum idle time allowed, in seconds, before a client is disconnected. The default idle time is 600 seconds (10 minutes). You can set a different value for each virtual host. If no value is set for this element, the server uses the value set in the <code>Server.xml</code> file. The value of the <code>MaxIdleTime</code> element in the <code>Vhost.xml</code> file overrides the value of the <code>MaxIdleTime</code> element in the <code>Server.xml</code> file.

3 Restart the server.

Configure settings for applications

You can disable this feature for an application or specify a different maximum idle time for an application in the `Application.xml` file.

1 Locate the following code in the `Application.xml` file and remove the comments:

```
<VirtualHost>
  <AutoCloseIdleClients enable="false">
    <MaxIdleTime>600</MaxIdleTime>
  </AutoCloseIdleClients>
</VirtualHost>
```

2 Edit the following elements.

Element	Description
AutoCloseIdleClients	Disable this feature for an individual application by setting the <code>enable</code> attribute to <code>false</code> . If this element is disabled in <code>Server.xml</code> , the feature is disabled for all applications, even if you specify <code>true</code> in the <code>Application.xml</code> file.
MaxIdleTime	Specifies the maximum idle time allowed, in seconds, before a client is disconnected. The default idle time is 600 seconds (10 minutes). You can set a different value for each application. If no value is set for this element, the server uses the value set in the <code>Vhost.xml</code> file. If no value is set for this element in the <code>Vhost.xml</code> file, the server uses the value in the <code>Server.xml</code> file. The value of the <code>MaxIdleTime</code> element in the <code>Vhost.xml</code> file overrides the value of the <code>MaxIdleTime</code> element in the <code>Server.xml</code> file.

3 Restart the server.

Configure how applications are assigned to server processes

Note: This section does not apply to Adobe Media Streaming Server because it doesn't support multiple processes.

In some scenarios, you might want to change how applications are assigned to server processes. When you start the server, you are starting a process called `AMSMaster.exe` (Windows) or `amsmaster` (Linux). Application instances run in processes called `AMSCore.exe` (Windows) and `amscore` (Linux). The master process is a monitor that starts core processes when necessary. There can be only one master process running at a time, but there can be many core processes running at a time.

Note: The number of core processes you can run is limited by system memory. Do not run more than 100, and you probably won't need more than 20. If you are configuring the number of core processes and using the `reloadApp()` Administration API or HTTP command, see [Tech Note kb403044](#).

You can configure how applications are assigned to server processes in the `Process` section of the `Application.xml` configuration file. Settings in an `Application.xml` file in a virtual host folder (for example, `rootinstall/conf/_defaultRoot/_defaultVHost/_Application.xml`) apply to all the applications running in that virtual host. Settings in an `Application.xml` file in an application's folder (for example, `rootinstall/applications/myApp/Application.xml`) apply only to that application. The following is the XML structure:

```
<Application>
  <Process>
    <Scope>vhost</Scope>
    <Distribute numprocs="3"></Distribute>
    <LifeTime>
      <RollOver></RollOver>
      <MaxCores></MaxCores>
    </LifeTime>
    <MaxFailures>2</MaxFailures>
    <RecoveryTime>300</RecoveryTime>
  </Process>
  ...
</Application>
```

Configure a process scope

- ❖ The `Scope` tag specifies at which level application instances are assigned to core processes. An application instance can run by itself in a process or it can run in a process with other instances. Enter one of the following values for the `Scope` element.

Value	Description
adaptor	All application instances in an adaptor run together in a process.
vhost	All application instances in a virtual host run together in a process. This is the default value.
app	All instances of a single application run together in a process.
inst	Each application instance runs in its own process. This provides the best application isolation and uses the most system resources. Running every instance in its own process can create many processes. You can set the <code>Distribute numprocs</code> attribute to a value greater than 1 to distribute instances across that number of processes.

Distribute a process scope among multiple core processes

The four process scopes don't provide a good distribution for all scenarios. For example, if you have one application and want to run 25 instances of that application, you could either distribute those instances to 1 core process (`<Scope>app</Scope>`) or to 25 core processes (`<Scope>inst</Scope>`). In this scenario, you could set `Scope` to `app` and `Distribute numprocs` to 3 to distribute the application instances among three core processes.

Note: There is no limit to the value of the `numprocs` attribute, but you should never need more than 40. Depending on available RAM, a number between 3 and 11 is realistic for most cases. Adobe recommends using prime number values because they result in a more even distribution of connections to processes.

Scopes have an enclosing relationship with a strict ordering: adaptors contain virtual hosts, which contain applications, which contain instances, which contain clients. The value of the `Distribute` tag must be a scope that is lower in order than the value in the `Scope` tag. In other words, if the value of `Scope` is `adaptor`, the value of `Distribute` can be `vhosts`, `apps`, `insts`, or `clients`. If the value of `Scope` is `app`, the value of `Distribute` can be `insts` or `clients`. By default, the server uses the value immediately lower than the one specified in the `Scope` tag.

- 1 Set the `scope` value.

- 2 Set the `numprocs` value to a value higher than 1. The default value of `numprocs` is 3, which means that the default behavior is to distribute application instances to three core processes.
- 3 Enter one of the following values for the `Distribute` element.

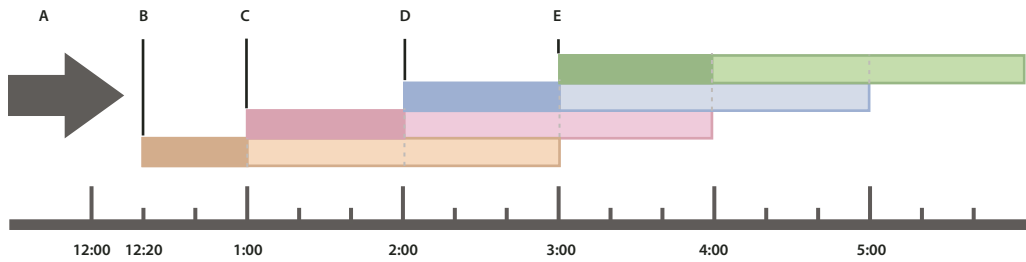
Value	Description
<code>vhost</code> or <code>vhosts</code>	All instances of applications in a virtual host run together in a process.
<code>app</code> or <code>apps</code>	All instances of an application run together in a process.
<code>inst</code> or <code>insts</code>	Each application instance runs in its own process. This is the default value. If you choose this value, you must also set the <code>Distribute numprocs</code> attribute to a value greater than 1.
<code>client</code> or <code>clients</code>	Each client connection runs in its own process. Use this value for stateless applications only. Stateless applications don't require clients to interact with other clients and don't have clients accessing live streams. Most vod (video on demand) applications are stateless because each client plays content independently of all other clients. Chat and gaming applications are not stateless because all clients share the application state. For example, if a shared chat application were set to <code>client</code> , the messages wouldn't reach everyone in the chat because they'd be split into separate processes.

Configure the number of core processes and how long each process runs

- ❖ Specify the number of core processes in the `MaxCores` tag (the maximum number of core processes that can exist concurrently) and the number of seconds that a core process can run in the `RollOver` tag. When a core process reaches the limit, any new connections roll over to a new core process.

The following diagram depicts the way in which the server rolls over processes. In the XML, the rollover time is set to 3600 seconds (1 hour), indicating that every hour a new process should start, and the maximum core processes value is set to 3, indicating that the maximum number of processes at any given time is 3:

```
<Process>
  <Scope>app</Scope>
  <LifeTime>
    <RollOver>3600</RollOver>
    <MaxCores>3</MaxCores>
  </LifeTime>
  ...
```



A. Client connections B. Process 1 starts C. Process 2 starts D. Process 3 starts E. Process 4 starts; Process 1 ends, because the maximum core processes limit was reached

When each process starts, it accepts new connections until the next process starts: that is, when process 1 starts, it accepts new client connections to the application until process 2 starts; process 2 then accepts new client connections until process 3 starts; and so on.

Note that the duration of process 1 might or might not be the full duration specified by the rollover value, because rollover values are calibrated to the real clock time. The duration of process 1 is partially determined by the current time when process 1 starts. For example, as shown in the diagram, when process 1 starts, the current time is 12:20, so the duration of process 1 is only 40 minutes (because it is 40 minutes until the beginning of the hour in real time). The duration of the first process is determined by the clock time; subsequent processes have a duration equal to the specified rollover time.

To disable this feature, set `RollOver` to 0. This feature is disabled by default.

Note: If you have multiple VHosts with `Process/Scope` set to `adaptor`, you must set an identical `RollOver` value for each VHost.

In stateless applications, such as vod applications, old core processes continue to serve earlier connections. In this case, you can specify a value in the `MaxCores` tag to limit the maximum number of core processes that can run simultaneously. If the application is not stateless, the server ignores any value you assign to `MaxCores` and sets it to 1. This ensures that an application instance is not split across multiple processes, but clients are disconnected periodically. To disable this feature, set `MaxCores` to 0. This feature is disabled by default.

Note: An application is considered stateless if you configure it to distribute clients over multiple processes. To do this, set the `Distribute numprocs` attribute to a value greater than 1, then set the `Distribute` tag to `clients` or set the `Scope` tag to `inst`.

Check for process failures

- 1 Enter a value in the `MaxFailures` tag to specify the maximum number of process failures allowed before a core process is disabled. The default value is 2.
- 2 Once disabled, a master process will not launch a core process until a minimum recovery time elapses. Enter a value in the `RecoveryTime` tag to specify the minimum recovery time for contained elements; set the tag to 0 to disable checking for process failures.

Use this feature to guard against a behavior in which a faulty core process can use the entire CPU by being repeatedly launched very quickly.

Note: Applications that are loaded using the Administration API (including applications loaded using the Administration Console) are not checked for process failures.

Configure the RAW adaptor

Flash Media Server 3.5.3

The RAW (Record and Watch) file format records live media into configurable chunks that stream to any version of Flash Player. Use the RAW file format to serve long-length, multi-bitrate DVR streams without running into performance issues. The RAW file format records and plays back all streams that Adobe Media Server supports, including H.264 video, data-only, audio-only, and so on.

- 1 Open the `rootinstall/conf/Server.xml` file in a text editor.
- 2 To configure how the server uses the RAW adaptor, edit the following XML parameters:

```
<Server>
  <Streams>
    <StreamLogLevel>warning</StreamLogLevel>
    <Raw>
      <EnableAggMsgs>true</EnableAggMsgs>
      <MaxAggMsgSize>65536</MaxAggMsgSize>
    </Raw>
  . . .
```

Element	Default value	Description
StreamLogLevel	warning	Controls log levels for all stream adaptors (FLV, MP4, and RAW). The default value is warning. Possible values are verbose, warning, and error.
Raw	Container element.	Contains elements that control the RAW adaptor.
EnableAggMsgs	true	Specifies whether the RAW adaptor generates aggregate messages (true) or not (false). The default value is true. Aggregating messages improves server performance.
MaxAggMsgSize	65536	The maximum size of an aggregate message, in bytes. The default value is 65536. You can use any positive integer.

- 3 Save and validate the XML file.
- 4 Restart the server.

Disable RTMPE

Important: By default, RTMPE is enabled in the server's `Adaptor.xml` file. For maximum security, it's best to leave RTMPE enabled. For more information, see the [Adobe Media Server Hardening Guide](#).

In some scenarios, however, you might want to disable RTMPE (encrypted Real-Time Messaging Protocol). Because RTMPE uses encrypted channels, there is a minor impact on performance; RTMPE requires about 15% more processing power than RTMP. If you don't control the applications that connect to Adobe Media Server and you don't want them to use RTMPE, you can disable RTMPE at the adaptor level.

To request an encrypted or encrypted tunnelling channel, applications specify `rtmpe` or `rtmppte`, respectively, in the `NetConnection.connect()` URL, for example, `nc.connect("rtmpe://www.example.com/myMediaApplication")`. If an application specifies RTMPE without explicitly specifying a port, Flash Player scans ports just like it does with RTMP, in the following order: 1935 (RTMPE), 443 (RTMPE), 80 (RTMPE), 80 (RTMPTE).

Note: RTMPE cannot currently be used between servers or from edge to origin. In these cases, RTMPS can be used instead.

- 1 Open the `Adaptor.xml` file for the adaptor you want to disable (located in `rootinstall/conf`).

2 Locate the following XML:

```
<RTMP>
<!-- RTMPE is the encrypted RTMP protocol covering both RTMPE and RTMPTE -->
<!-- This is enabled by default, setting enabled to "false will not -->
<!-- allow these protocols on this adaptor -->
    <RTMPE enabled="true"></RTMPE>
</RTMP
```

3 Set the `RTMPE enabled` attribute to "false".

4 Restart the server.

See also

[“XML configuration files reference”](#) on page 125

[Adobe Media Server Hardening Guide](#)

Configuring security features

For information about security, see the [Adobe Media Server Hardening Guide](#).

Restrict which domains can connect to a virtual host

If desired, you can restrict which domains are allowed to connect to a virtual host. By default, connections are allowed from all domains.

1 Open the `rootinstall/conf/ams.ini` file.

2 Set the `VHOST.ALLOW` parameter to a comma-delimited list of domains that are allowed to connect to the server. The default value is `all`.

If a value is set, only the domains listed are accepted. For example, `VHOST.ALLOW = example.com, example2.com` allows connections from the `example.com` and `example2.com` domains. To allow localhost connections, specify `localhost`. For more information, see [“Vhost.xml file”](#) on page 238.

3 Restart the server.

Verify SWF files

Flash Media Server 3.0, Flash Player 9 Update 3, AIR 1

Note: To perform SWF verification for HTTP streaming, see [SWF verification for Protected HTTP Dynamic Streaming](#).

You can configure the server to verify client SWF files before allowing them to connect to an application. Verifying SWF files prevents third parties from creating their own applications that attempt to stream your resources.

You can verify SWF files at the following server scopes:

- Individual applications. Configure the application-specific `Application.xml` file.
- All applications in a virtual host. Configure the `Application.xml` file in the `Vhost` folder.
- Administrative applications that have access to all server levels. Configure the `Server.xml` file and an `Application.xml` file.

SWF verification is disabled by default. To verify SWF files, you must set `<SWFVerification enable = "true">` in the `Application.xml` configuration file. When verification is enabled, the server verifies that the client SWF file requesting a connection to the server matches the *verifying SWF file*. The verifying SWF file is a copy of the client SWF file that you place on the server or in an external content repository.

By default, the server looks in `rootinstall/applications/application_name/SWFs` folder for the verifying SWF file. You can override this default in the `SWFFolder` tag of the `Application.xml` file, the `Server.xml` file, or in the File plug-in. If you installed Apache with Adobe Media Server, you can add a path to the folder from which you serve SWF files. For example, on Windows, adding the path `<SWFFolder>C:\Program Files\Adobe\Adobe Media Server 4\webroot</SWFFolder>` prevents you from needing to maintain two copies of each SWF file.

Note: *If you're deploying an Adobe AIR application, copy the SWF file you compiled into the AIR package to the server or external content repository.*

When the client SWF file connects to the server, the server verifies it. If the SWF file is verified, it is allowed to connect to the application. If a client SWF file fails verification, the server disconnects the `NetConnection` object and logs an event to the access log. The server does not send a `NetStream` status message to the client.

For client applications that comprise multiple SWF files or SWF files that dynamically load other SWF files, the SWF file that contains the `NetConnection` object to connect to the server is the one that the server attempts to verify.

Note: *SWF files connecting to Adobe Media Administration Server cannot be verified.*

Locating SWF files for verification

The server looks for verifying SWF files in the following locations, in order:

- 1 The application-specific `Application.xml` file, if it exists. The `Application.xml` file defines the location of verifying SWF files in the `SWFFolder` tag. If `SWFFolder` contains a file path, the server looks for verifying SWF files in that file path. If `SWFFolder` is empty, the server looks in the default location: `/applications/application_name/SWFs`.
- 2 The Vhost-level `Application.xml` file. This `Application.xml` file defines the location of verifying SWF files for all applications on the virtual host. The `SWFFolder` tag defines the location. If `SWFFolder` contains a file path, the server looks for verifying SWF files in that file path. If `SWFFolder` is empty, the server looks in the default location: `/applications/application_name/SWFs`.
- 3 The server checks the `SWFFolder` tag in the `Server.xml` file. This tag specifies a location for global SWF files, which means SWF files that are common to all applications.
- 4 If a File plug-in is enabled for retrieval of SWF files, the server calls the File plug-in. The server passes any values present in `Application.xml` or `Server.xml` to the plug-in. The plug-in developer overrides these values with paths to the external content repository.

Configure SWF verification for applications

- 1 Locate the following section of the `Application.xml` file:

```
<Application>
...
<SWFVerification enabled="false">
  <SWFFolder></SWFFolder>
  <MinGoodVersion></MinGoodVersion>
  <DirLevelSWFScan>1</DirLevelSWFScan>
  <MaxInitDelay>5</MaxInitDelay>
  <FirstHashTimeout>5</FirstHashTimeout>
  <FinalHashTimeout>60</FinalHashTimeout>
  <UserAgentExceptions>
    <Exception to="" from=""/>
  </UserAgentExceptions>
  <Cache>
    <TTL></TTL>
    <UpdateInterval></UpdateInterval>
  </Cache>
</SWFVerification>
</Application>
```

2 Edit the following elements.

Element	Attribute	Description
SWFVerification	enabled	Set the <code>enabled</code> attribute to <code>"true"</code> or <code>"false"</code> to turn this feature on or off. The default value is <code>"false"</code> .
SWFFolder	None.	<p>A single folder or a semicolon-delimited list of folders that contain copies of client SWF files. These SWF files are used to verify connecting SWF files. They are the <i>verifying</i> SWF files.</p> <p>The default value is the application's folder appended with <code>/SWFs</code>. For example, for an application called <code>myApplication</code>, if there isn't a value set for this element, verifying SWF files should be placed in the <code>applications/myApplication/SWFs</code> folder.</p> <p>If you are using the File plug-in to verify SWF files in an external content repository, leave this field blank.</p> <p>If you installed Apache with Adobe Media Server, enter the path to the folder from which you are serving SWF files. For example, if you're serving SWF files from the webroot directory on Windows, enter the path <code>C:\Program Files\Adobe\Adobe Media Server 4\webroot</code>.</p> <p>Important: All SWF files in the specified folder are hashed and cached in memory when the server starts and every time the cache TTL expires. A large number of SWF files can result in start-up time delays and high memory usage.</p>
MinGoodVersion	None.	Specifies the minimum version of this feature to accept. The default value is <code>0</code> , which allows this and all future versions. This is a reserved field that works with Flash Player; Adobe recommends keeping the default value.
DirLevelSWFScan	None.	Specifies the number of levels of subfolders within a parent folder to scan for SWF files. The parent folder is specified in the <code>SWFFolder</code> element. Specifying a positive value scans that number of subfolder levels. Specifying zero scans the parent folder and no subfolders. Specifying a negative value scans all subfolder levels. The default value is <code>1</code> , which means that the server scans only one subfolder level.
MaxInitDelay	None	The maximum amount of time to process SWF files, in seconds. The default value is 5 seconds.
FirstHashTimeout	None	The maximum amount of time that a client can use to provide its first verification to the server.

Element	Attribute	Description
FinalHashTimeout	None	The maximum amount of time that a client can use to provide its final verification to the server.
UserAgentExceptions	None.	Container.
Exception	from to	A user agent to except from verification. Use the <code>from</code> and <code>to</code> attributes to indicate the lowest and highest versions to except. This is a string comparison, with editing to make all numeric fields equal in length. For more information, see the comments in the <code>Application.xml</code> file and “Create verification exceptions” later in this topic.
Cache	None.	Container.
TTL	None.	The time to live for the SWF file, in minutes. The default value is 1440 minutes (24 hours). If a SWF file is removed from the server, the verification values stay in the cache for 24 hours; users can connect to the application until the cache expires.
UpdateInterval	None.	The maximum time in minutes to wait for the server to scan the SWFs folders for updates when there is a miss in the cache. The default value is 5 minutes, which means a SWF file copied to the SWFs folder is picked up by the server within 5 minutes.

Create verification exceptions

- ❖ Add `Exception` elements to the `UserAgentExceptions` section of the `Application.xml` file.

Certain applications—for example, Adobe® Flash® Media Live Encoder—do not support the form of SWF verification used by the server. You can add one or more exceptions to the SWF verification rules that allow specified user agents, such as Adobe Media Live Encoder, to bypass SWF verification, as in the following:

```
<SWFVerification enabled="true">
  ...
  <UserAgentExceptions>
    <Exception to="FME/0.0" from="FME/2.5"/>
  </UserAgentExceptions>
</SWFVerification>
```

Configure SWF verification globally

You can configure verification for a group of SWF files on the server that are common to all applications.

- ❖ In the `Server.xml` file `Root/Server/SWFVerification/SWFFolder` tag, specify folders that hold the verifying SWF files. You can also configure other values for these SWF files.

Note: In addition to configuring the `SWFFolder` tag in the `Server.xml` file, enable verification in the `Vhost-level` or `application-specific Application.xml` file. See “Configure SWF Verification for Applications” in this topic.

Create folders for the verifying SWF files

- 1 If the `SWFFolder` value is the default, create a folder called `SWFs` in the application’s folder on the server.
 For example, for an application called `myMediaApp`, create the folder `applications/myMediaApp/SWFs`. SWF files in the `SWFs` folder verify connections to any instance of the `myMediaApp` application.
- 2 To verify SWF files for application instances, create instance folders in the `SWFs` directory, for example, `applications/myMediaApp/SWFs/chat01`, `applications/myMediaApp/SWFs/chat02`, and so on.

SWF files in the `SWFs` directory can verify all instances of the application; SWF files within an instance folder can verify only that instance.

Note: Multiple SWF files may exist in either directory. A SWF file can be renamed and still used for verification as long as it's a copy of the client SWF file.

Restrict access to the server with a cross-domain file

You can restrict access to an edge server or the Administration Server with a cross-domain XML file. The cross-domain XML file lets you specify a list of domains from which clients can access the edge server or Administration Server.

- 1 Open `rootinstall/conf/Server.xml` and locate the `CrossDomainPath` element in the `Server` element (or the `AdminServer` element, for the Administration Server):

```
<Server>
  ...
  <CrossDomainPath></CrossDomainPath>
```

- 2 Specify the location of the cross-domain file in the `CrossDomainPathElement`, for example:

```
<CrossDomainPath>C:/Security/config/files/ams/crossdomain.xml</CrossDomainPath>
```

- 3 Validate the `Server.xml` file, save it, and restart the server.

Limit access to Adobe Media Administration Server

The Administration Console connects to Adobe Media Administration Server, which connects to Adobe Media Server. By default, the Administration Server is installed on port 1111. Adobe recommends that you block all external access to port 1111 so that access to the admin server is restricted only to clients that are within your firewall.

Additionally, you can restrict access to the admin server by using domain based restrictions.

By default, a client can connect to Adobe Media Administration Server from any domain or IP address, which can be a security risk. If desired, you can change this in the `AdminServer` section of the `Server.xml` file.

- 1 Open `rootinstall/conf/Server.xml` and locate the following code:

```
<AdminServer>
  ...
  <Allow>all</Allow>
  ...
</AdminServer>
```

- 2 Edit the `Allow` element to specify which connections to Adobe Media Administration Server the server responds to. This is specified as a comma-delimited list of host names, domain names, and full or partial IP addresses, as well as the keyword `all`. For example: `<Allow>x.foo.com, foo.com, 10.60.1.133, 10.60</Allow>`.

- 3 Validate the XML, save the `Server.xml` file, and restart the server.

Set limits on unsuccessful login attempts

You can use the `LogInLimits` settings in the `rootinstall/conf/Server.xml` file to set the number of unsuccessful attempts a client is allowed when trying to log in to the admin server. With these settings, you define how many failed attempts are allowed before the client must wait to attempt to log in again. You also define the amount of time they must wait. In addition, you can lock the user out of the server entirely after a certain number of attempts.

To set the number of login attempts the client can have before waiting, use the `MaxFailures` sub-element. After that number of failures, the client must wait the number of seconds defined by the `RecoveryTime` sub-element. If the client tries unsuccessfully to log in a number of times equal to `LockOutLimit`, they are locked out of the server until the server is restarted.

In the following example, a user can try to log in 10 times. If they fail to log in successfully on the 10th attempt, they must wait 5 minutes (300 seconds) before attempting to log in again. If they fail 20 times total, they are barred from logging in until the server is restarted:

```
<Security>
  <LoginLimits>
    <!-- Default value is 3 -->
    <MaxFailures>10</MaxFailures>
    <!-- Default value is 30 -->
    <RecoveryTime>300</RecoveryTime>
    <!-- Default value is 100-->
    <LockoutLimit>20</LockoutLimit>
  </LoginLimits>
</Security>
```

For more information, see “[LogInLimits](#)” on page 209.

Configure SSL

Note: Adobe Media Server uses OpenSSL libraries internally. To upgrade OpenSSL libraries, see [TechNote 90293](#).

Secure Sockets Layer (SSL) is a protocol for enabling secure communications over TCP/IP. Adobe Media Server provides native support for both incoming and outgoing SSL connections. An incoming connection is a connection between Flash Player and the server. An outgoing connection is a connection between two servers.

RTMPS adheres to SSL standards for secure network connections and enables connections through a TCP socket on a secure port. Data passed over the secure connection is encrypted to avoid eavesdropping by unauthorized third parties. Because secure connections require extra processing power and may affect the server’s performance, use RTMPS only for applications that require a higher level of security or that handle sensitive or critical data.

By default, when Flash Player connects to Adobe Media Server, it scans the following ports in order: 1935, 443, 80, 80 (RTMP tunneling). To configure SSL, specify that a port is secure. To specify that a port is secure, place a minus (“-”) sign in front of the port number. The default secure port for RTMPS is 443. To configure the default secure port, enter -443 in the `ADAPTOR.HOSTPORT` parameter in the `rootinstall/conf/ams.ini` file, as follows:

```
ADAPTOR.HOSTPORT = :1935, -443
```

When Flash Player encounters an “`rtmps://amsdomain/application`” string, it communicates with Adobe Media Server over port 443. Adobe Media Server returns data to the client over port 443. Traffic with an “`rtmp://`” string uses port 1935.

To connect to Adobe Media Server over an SSL connection, clients must do the following:

- 1 Set the `NetConnection.proxyType` property to “best” before connecting.
- 2 Specify the RTMPS protocol in the call to the `NetConnection.connect()` method.

If SSL is configured, the following client code connects securely:

```
var nc:NetConnection = new NetConnection();
nc.proxyType = "best";
nc.connect("rtmps://amsexamples.adobe.com/vod/instance1");
```

If you configure any port other than 443 as secure, for example, -1935, the client must specify the port in the URI, for example, “`rtmps://host:1935/app`”.

Note: All server editions support a version of RTMP called RTMPE, which is an 128-bit encrypted protocol. RTMPE is more lightweight than SSL and is enabled by default. For more information, see the `NetConnection.connect()` entry in the *ActionScript® 3.0 Language Reference* or in the *ActionScript 2.0 Language Reference*.

Certificates

You can get an SSL certificate from a certificate authority or create a certificate yourself. If a certificate is signed by an intermediate Certificate Authority (CA), it must include the intermediate certificate as part of the certificate that the server returns to the client. Your certificate file (if signed by an intermediate CA) should look something like the following:

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

The first BEGIN CERTIFICATE/END CERTIFICATE pair is your certificate. The next BEGIN CERTIFICATE/END CERTIFICATE pair is the intermediate CA that signed your certificate. You can add additional sections as needed.

Secure incoming connections

Specify the location of the SSL certificate and the certificate's private key, and configure the adaptor to listen on a secure port.

- 1 Open the Adaptor.xml file for the adaptor you want to configure and locate the following code:

```
<Adaptor>
...
<SSL>
  <SSLServerCtx>
    <SSLCertificateFile></SSLCertificateFile>
    <SSLCertificateKeyFile type="PEM"></SSLCertificateKeyFile>
    <SSLPassPhrase></SSLPassPhrase>
    <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
    <SSLSessionTimeout>5</SSLSessionTimeout>
  </SSLServerCtx>
</SSL>
...
</Adaptor>
```

- 2 Edit the following elements.

Element	Description
SSLCertificateFile	The location of the certificate file to send to the client. Specify an absolute path or a path relative to the adaptor folder.
SSLCertificateKeyFile type="PEM"	The location of the private key file for the certificate. Specify an absolute path or a path relative to the adaptor folder. The type attribute specifies the type of encoding used for the certificate key file. This can be either "PEM" or "ASN1". The default value is "PEM". The private key and the certificate can be combined into one file. If the key file is encrypted, the pass phrase must be specified in the SSLPassPhrase tag.

Element	Description
SSLPassPhrase	The pass phrase to use for decrypting the private key file. If the private key file is not encrypted, leave this tag empty.
SSLCipherSuite	The SSL ciphers: a colon-delimited list of components. A component can be a key exchange algorithm, authentication method, encryption method, digest type, or one of a selected number of aliases for common groupings. Do not change the default settings unless you are very familiar with SSL ciphers. The possible values are listed here: " SSLCipherSuite " on page 139.
SessionTimeout	The amount of time in minutes a session remains valid. Any value less than 1 is read as 1. The default value is 5. If a client reconnects to a session before SessionTimeout is reached, the cipher suite list isn't sent during the SSL handshake.

- 3 To configure a secure port for an adaptor, specify a minus sign before the port number in the ADAPTOR.HOSTPORT parameter in the `rootinstall/conf/ams.ini` file, as follows:

```
ADAPTOR.HOSTPORT = :1935, -443
```

This configuration tells the server to listen on ports 1935 and 443, and that 443 is a secure port that receives only RTMPS connections. An RTMPS connection to port 1935 fails: the client attempts to perform a SSL handshake that the server fails to complete. An RTMPS connection to port 443 succeeds: the client performs a SSL handshake that the server completes. An RTMP connection to port 443 also fails: the server tries to perform a SSL handshake that the client fails to complete.

- 4 Restart the server.
- 5 Check the `rootinstall/logs/edge.00.log` file to verify that no errors have been reported.

Configure incoming connections when multiple virtual hosts are assigned to one adaptor

You can configure the server to return a certificate based on which port a client connects to. This lets you assign multiple virtual hosts to one adaptor and return a different certificate for each virtual host.

Note: Generally, if you're hosting multiple customers, each virtual host has its own domain name. Each domain name must have its own certificate.

- 1 Locate the following code in the `Adaptor.xml` file:

```
<Adaptor>
  ...
  <HostPortList>
    <HostPort name="edge1" ct1_channel=":19350">${ADAPTOR.HOSTPORT}</HostPort>
  </HostPortList>
  ...
</Adaptor>
```

- 2 Create new `HostPort` elements with unique name and `ct1_channel` attributes and unique port values for RTMP and SSL.

For example, add the following `HostPort` tag in addition to the default `HostPort` tag:

```
<HostPort name="edge2" ct1_channel=":19351">:1936, -444</HostPort>
```

- 3 For each `HostPort` element, enter an `Edge` element under the `SSL` element with an identical name attribute. If you don't specify an `Edge` element, the edge uses the default SSL configuration.

This sample code demonstrates how to configure `edge1` to return `cert2.pem` when a client connects to it on port 443. Since there is no `Edge` tag for `edge2`, `edge2` will use the default configuration specified in the `SSLServerCtx` section that is directly under the `SSL` container tag. The `edge2` server returns `cert.pem` when a client connects to it on port 444.

```
<SSL>
  <SSLServerCtx>
    <SSLCertificateFile>cert.pem</SSLCertificateFile>
    <SSLCertificateKeyFile>private.pem</SSLCertificateKeyFile>
    <SSLPassPhrase></SSLPassPhrase>
    <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
    <SSLSessionTimeout>5</SSLSessionTimeout>
  </SSLServerCtx>
  <Edge name="edge1">
    <SSLServerCtx>
      <SSLCertificateFile>cert2.pem</SSLCertificateFile>
      <SSLCertificateKeyFile>private2.pem</SSLCertificateKeyFile>
      <SSLPassPhrase></SSLPassPhrase>
      <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
      <SSLSessionTimeout>5</SSLSessionTimeout>
    </SSLServerCtx>
  </Edge>
</SSL>
```

4 Validate the XML and save the file.

5 Restart the server.

Secure outgoing connections

1 Open the `Server.xml` file and locate the following code:

```
<Root>
  <Server>
    <SSL>
      <SSLEngine></SSLEngine>
      <SSLRandomSeed></SSLRandomSeed>
      <SSLClientCtx>
        <SSLVerifyCertificate>true</SSLVerifyCertificate>
        <SSLCACertificatePath></SSLCACertificatePath>
        <SSLCACertificateFile></SSLCACertificateFile>
        <SSLVerifyDepth>9</SSLVerifyDepth>
        <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
      </SSLClientCtx>
    </SSL>
  </Server>
</Root>
```

2 Edit the following elements.

Element	Description
SSLRandomSeed	The number of bytes of entropy to use for seeding the pseudo-random number generator (PRNG). You cannot specify anything less than 8 bytes, and the default value is 16. Entropy is a measure of randomness. The more entropy, the more random numbers the PRNG will contain. The server may take longer to start up if you specify a large number.
SSLSessionCacheGC	How often to flush expired sessions from the server-side session cache, in minutes.
SSLVerifyCertificate	A Boolean value specifying whether to verify the certificate returned by the server being connected to (<code>true</code>) or not (<code>false</code>). The default value is <code>true</code> . Disabling certificate verification can result in a security hazard. Do not disable verification unless you are certain you understand the ramifications.
SSLCACertificatePath	A folder containing certificates. Each file in the folder must contain only a single certificate, and the file name must be hash and the extension ".0", for example, e98140a6.0. On a Windows 32-bit operating system, if this tag is empty, the server looks for certificates in the <i>rootinstall\certs</i> directory. You can import the Windows certificate store to the certs directory by running <code>AMSMaster -console -initialize</code> from a command line. In Linux, you must specify the location of the certificates.
SSLCACertificateFile	Specifies the name of a file containing one or more certificates in PEM format.
SSLVerifyDepth	Specifies the maximum depth of an acceptable certificate. If a self-signed root certificate cannot be found within this depth, certificate verification fails. The default value is 9.
SSLCipherSuite	The SSL ciphers: a colon-delimited list of components. A component can be a key exchange algorithm, authentication method, encryption method, digest type, or one of a selected number of aliases for common groupings. Do not change the default settings unless you are very familiar with SSL ciphers. The possible values are listed here: " SSLCipherSuite " on page 227.

Configure adaptors to manage outgoing SSL connections independently

The `SSL` section in the `Server.xml` file configures all adaptors to use the same settings. However, you might want to use a different certificate for each virtual host. In this case, assign one virtual host to each adaptor and configure your adaptors individually to override the settings in the `Server.xml` file.

- ❖ Copy the `SSL` section in the `Server.xml` file to the `Adaptor.xml` files and enter the new values. You don't need to copy the `SSLRandomSeed` tag, as this tag is a server-level setting that cannot be overridden in `Adaptor.xml`.

Configure virtual hosts to manage outgoing SSL connections independently

For example, you can disable certificate checking in one virtual host, use a certificate in a different folder for one virtual host, and implement a different set of ciphers in a third virtual host.

- 1 Uncomment the `SSL` section under the `Proxy` tag in the appropriate `Vhost.xml` file:

```
<VirtualHost>
  ...
  <Proxy>
    <SSL>
      <SSLClientCtx>
        <SSLVerifyCertificate></SSLVerifyCertificate>
        <SSLCACertificatePath></SSLCACertificatePath>
        <SSLCACertificateFile></SSLCACertificateFile>
        <SSLVerifyDepth></SSLVerifyDepth>
        <SSLCipherSuite></SSLCipherSuite>
      </SSLClientCtx>
    </SSL>
  </Proxy>
</VirtualHost>
```

When the `SSL` tag is present, the entire `SSL` section is used to configure the virtual host. If an `SSL` tag is omitted from this section, the server uses the default settings.

2 Restart the server.

Configure Adobe Media Server to work with a hardware SSL proxy

When you use a hardware SSL proxy, you don't need to configure Adobe Media Server for SSL. The hardware sits between Adobe Media Server and the Internet. Data sent between Adobe Media Server and the hardware is unencrypted. The hardware encrypts the data and forwards it to the Internet.

Configure the hardware to listen externally on port 443 to receive encrypted data sent over Internet from clients. Configure the hardware to forward data to Adobe Media Server on port 1935. Adobe Media Server listens on port 1935 by default. If you've changed the default port, configure the hardware to forward data on that port.

Configure RPCs (Remote Procedure Calls)

Adobe Media Server provides settings in the `Application.xml` file for configuring which RPC methods can be called. The default is to block all RPC calls.

You can enable RPCs by setting the `<RPC>` element's `enable` attribute to `true`; for example:

```
<RPC enable="true">
```

After enabling RPCs, you must then list individual methods that are allowed. Each type of class—`Client`, `NetConnection`, `Stream`, and `SharedObject`—has its own whitelist of methods that can be called. If a method or object is not explicitly listed in the whitelist, then it is considered blocked.

The settings for RPCs are in the `<Security><RPC>` element of the `Application.xml` file. This element contains the following sub-elements:

- `<Client>` — All methods blocked by default.
- `<NetConnection>` — Only the `onStatus()` method is allowed by default, if RPCs are enabled.
- `<SharedObject>` — All methods blocked by default.
- `<Stream>` — Only the `onStatus()` method is allowed by default, if RPCs are enabled.

Each of these sub-elements takes a `<Method>` sub-element. Each `<Method>` sub-element takes an `<Allow>` sub-element. The `<Allow>` sub-element is a comma-delimited list of allowed methods.

To add a method to a class's whitelist:

1 Enable RPCs by setting the `<RPC>` element's `enabled` attribute to `true`; for example:

```
<RPC enable="true">
```

- 2 Add the method you want to allow to the class's `<Allow>` element. The following example allows the `getLocal()` and `setFps()` methods to be called on the `SharedObject` class:

```
<SharedObject>  
  <Method>  
    <Allow>getLocal,setFps</Allow>  
  </Method>  
</SharedObject>
```

- 3 Ensure that the defaults for the other classes meet your security standards. After enabling RPCs for all classes, you should review the default allowed methods.

The method names are case-sensitive and should be in one of the following formats: `[objectPath.]method` or `[objectPath.]object`

If you allow an object but do not list its methods, then all methods of that object are allowed. For example, `<Allow>a.b</Allow>` matches methods `a.b.c()` and `a.b.d()`.

Enable virtual directory mappings for server-side File objects

You can specify virtual directory mappings for server-side File objects in the `Application.xml` configuration file. However, the feature must be enabled at the server level in the `Server.xml` configuration file.

To enable virtual directories for server-side File objects, set the `<VirtualDirectoryForFile>` element's `enable` attribute to `true` in the `Server.xml` file. In the `Application.xml` file, add a `<VirtualDirectory>` sub-element to the `<ScriptingEngine><FileObject>` element.

Only enable `VirtualDirectoryForFile` when you must provide virtual directories for server-side scripting; it is not required for adding virtual directories for streams. When virtual directories for File objects are enabled, anyone who has access to a server-side script can write files anywhere on the server disk. As a result, it is disabled by default. Activating it requires a configuration from the `Server.xml` file administrator.

If you map File objects in the `Application.xml` file and don't have the feature enabled in the `Server.xml` file, the following message appears in the log:

```
"Virtual directories for file objects is not supported due to the Server level security setting."
```

The following example enables virtual directory mappings for server side File objects in the `Server.xml` file:

```
<Root>  
  <Server>  
    <Security>  
      <VirtualDirectoryForFile enable="true"></VirtualDirectoryForFile>  
      ...  
    ...  
  ...  
</Root>
```

The following example from the `Application.xml` file creates a new virtual directory mapping for server-side File objects:


```
<Application>
  <ScriptEnging>
    <FileObject override="no">
      <VirtualDirectory>
        /flashapps;C:\dev\Code\Flash\tincan\flashapps\
      </VirtualDirectory>
    </FileObject>
    ...
  </ScriptEnging>
  ...
</Application>
```

Performing general configuration tasks

Enable checkpoint logging events

Many companies use statistics from the access log to bill customers. If your programming includes live events or 24/7 programming, the events you use to calculate billing might not occur within a billing cycle. To solve this problem, you can enable *checkpoint* events. Checkpoint events log bytes periodically during an event. The following are available as checkpoint events: *connect-continue*, *play-continue*, and *publish-continue*.

You must enable checkpoint events at the server level in the `Server.xml` file. You can disable checkpoints at the vhost and application level in the `Vhost.xml` and `Application.xml` files. You can also override the logging interval at the vhost and application levels. For more information, see the comments in the XML files located in the `rootinstall/conf` folder.

- 1 Open the `rootinstall/conf/Server.xml` file in a text editor.
- 2 Locate the `Root/Server/Logging/Checkpoints` element:

```
<Checkpoints enable="false">
  <CheckInterval>60</CheckInterval>
  <LogInterval>3600</LogInterval>
</Checkpoints>
```

- 3 Set the `Checkpoints enable` attribute to `true`, like this: `<Checkpoints enable="true">`.
- 4 You can optionally set values for the `CheckInterval` and `LogInterval` elements to indicate how often the server should check for and log events.
- 5 Save and validate the file.
- 6 Restart the server.

Allow Administration API methods to be called over HTTP

You must specify each Administration API method that may be called over HTTP.

- 1 Open the `rootinstall/conf/ams.ini` file.
- 2 Set the `USERS.HTTPCOMMAND_ALLOW` parameter to a comma-delimited list of APIs, and restart the server. The default value is `ping`. For more information, see “[Allow \(HTTPCommands\)](#)” on page 235.

Allow application debugging connections

To play back streams and obtain data from shared objects, the Administration Console must make a special debugging connection to the server. By default, the server does not allow this connection.

- 1 Open the `Application.xml` file.

Note: You can set these values in an `Application.xml` file at the `VHost` level or at the application level. To set the value at the application level, copy an `Application.xml` file to the application's folder.

- 2 Locate the following XML:

```
<Debug>
  <MaxPendingDebugConnections>50</MaxPendingDebugConnections>
  <AllowDebugDefault>>false</AllowDebugDefault>
</Debug>
```

- 3 Set the `AllowDebugDefault` element to `true`.

Note: Debug connections count against license limits.

- 4 Save and validate the file.

- 5 Restart the server.

Configuring IPv6

IPv6 (Internet Protocol version 6) is a new version of Internet Protocol that supports 128-bit addresses. The current version of Internet Protocol, IPv4, supports 32-bit addresses. IPv6 alleviates the address shortage problem on the Internet. A system that only runs IPv6 can't communicate with a system that only runs IPv4.

By default, Adobe Media Server runs in IPv4 mode, even if the operating system is configured for IPv6. When Adobe Media Server runs in IPv6 mode, hostnames specified in `Adaptor.xml` are resolved to IPv4 and IPv6 addresses.

Important: In Red Hat® Linux systems, you must update the `NETWORKING_IPV6` value in `/etc/sysconfig/network` when installing or uninstalling IPv6.

Activate IPv6 on the network interface card.

IPv6 is embedded in all operating systems that the server supports. You may need to activate IPv6 on the interfaces. For more information, see the operating system's Help.

Allow the server to listen on IPv6 sockets.

- 1 Open the `rootinstall/conf/Server.xml` file in a text editor.

- 2 Locate the `NetworkingIPv6` tag and set `enable` to `"true"`:

```
<NetworkingIPv6 enable="true" />
```

- 3 Save the file and restart the server.


Enclose numeric IPv6 addresses in URLs in brackets.

Wherever a numeric IPv6 address is used in a client-side script, server-side script, or in the server configuration files, enclose it in brackets:

```
rtmp:// [fd5e:624b:4c18:ffff:0:5efe:10.133.128.108] :1935/streamtest
```

You must specify the interface zone index for a link-local address:

```
rtmp:// [fe80::204:23ff:fe14:da1c%4] :1935/streamtest
```

 *It's a good idea to register the RTMP, RTMPS, and RTMPE protocols with a network services database and use a service name (or decimal port number, if necessary) in the server configuration files.*

Check the logs

When the server starts, it logs available stack configuration, host name, and all available IP addresses for the host in the master.xx.log, edge.xx.log, and admin.xx.log files (located in the *rootinstall/logs/* directory). The following x-comment fields from a sample edge log file indicate that the IPv6 stack and the IPv4 stack are available, and that the server host has dual addresses and is listening on both interfaces:

```
AMS detected IPv6 protocol stack!
AMS config <NetworkingIPv6 enable=true>
AMS running in IPv6 protocol stack mode!
Host: amsqewin2k3-02 IPv4: 10.133.192.42 IPv6: fe80::204:23ff:fe14:dalc%4
Listener started ( _defaultRoot_? ) : 19350/v6
Listener started ( _defaultRoot_? ) : 19350/v4
Listener started ( _defaultRoot_? ) : 1935/v6
Listener started ( _defaultRoot_? ) : 1935/v4
```

On Red Hat Linux, the edge logs display only the highest IP version the socket listeners are using, even if the socket listeners accept connections over both IPv4 and IPv6. In the previous example, only the v6 entries would be displayed.

On IPv6-enabled Linux, if you are using an host name that resolves to IPv4 host name with an RTMP or RTMPE connection, configure the Adaptor.xml appropriately to resolve connections quickly. See “[HTTPIdent2](#)” on page 131.

Defining Application object properties

You can define properties for the server-side Application object in the server's Application.xml configuration files. If you define properties in the default Application.xml file, the properties are available for all applications on a virtual host. If you define properties in an Application.xml file in an application folder, the properties are available only for that application.

To define a property, create an XML tag in the `ScriptEngine` section of the Application.xml file. The property name corresponds to the tag's name, and the property value corresponds to the tag's contents.

For example, the following XML fragment defines the properties `user_name` and `dept_name`, with the values `jdoe` and `engineering`, respectively:

```
<Application>
  <ScriptEngine>
    <config>
      <user_name>jdoe</user_name>
      <dept_name>engineering</dept_name>
    </config>
  </ScriptEngine>
</Application>
```

To access the property in server-side code, use the syntax in either of these examples:

```
application.config.prop_name
application.config["prop_name"]
```

Note: *The properties you define are accessible from `application.config.property`, not from `application.property`.*

For example, given the previous XML fragment, the following `trace()` statements are valid:

```
trace("I am " + application.config.user_name + " and I work in the " +  
application.config.dept_name + " department.");  
trace("I am " + application.config["user_name"] + " and I work in the " +  
application.config["dept_name"] + " department.");
```

The output from either statement would be as follows:

I am jdoe and I work in the engineering department.

You can also use environment variables and symbols you define in the substitution.xml file. For example, assume that the environment variable `COMPUTERNAME` is equal to `jsmith01`, and you have defined a symbol named `DEPT` in the substitution.xml file:

```
<Root>  
  <Symbols>  
    <DEPT>Engineering</DEPT>  
  </Symbols>  
</Root>
```

In addition, the following XML appears in the Application.xml file:

```
<Application>  
  <ScriptEngine>  
    <config>  
      <compName>${%COMPUTERNAME%}</compName>  
      <dept>${DEPT}</dept>  
    </config>  
  </ScriptEngine>  
</Application>
```

In a server-side script, the following `trace` statements are valid:

```
trace("My computer's name is: " + application.config.compName);  
trace("My department is: " + application.config.dept);
```

The output is as follows:

```
My computer's name is: jsmith01  
My department is: Engineering
```

Note: Server-side ActionScript `trace()` statements display in the Live Log panel of the Administration Console.

More Help topics

[“Using symbols in configuration files”](#) on page 18

Configure or disable native bandwidth detection

The server can detect a client’s bandwidth in the core server code (called *native bandwidth detection*), or in a server-side script (called *script-based bandwidth detection*). Native bandwidth detection is faster than script-based because the core server code is written in C and C++. Also, with native bandwidth detection, if a client connects through edge servers, the outermost edge server detects the bandwidth, which results in more accurate data. Native bandwidth detection is enabled and configured by default.

The server detects bandwidth by sending a series of data chunks to the client, each larger than the last. If desired, you can configure the size of the data chunks, the rate at which they’re sent, and the amount of time the server sends data to the client. For more information about detecting bandwidth in an application, see the *Developer Guide*.

- 1 Open the Application.xml file.

Note: You can set these values in an *Application.xml* file at the *VHost* level or at the application level. To set the value at the application level, copy an *Application.xml* file to the application's folder.

2 Locate the following code:

```
...  
...  
    <BandwidthDetection enabled="true">  
        <MaxRate>-1</MaxRate>  
        <DataSize>16384</DataSize>  
        <MaxWait>2</MaxWait>  
    </BandwidthDetection>  
...  
...
```

Note: To disable native bandwidth detection, set the *enabled* attribute to *false* and restart the server.

3 Edit the following elements.

Element	Description
BandwidthDetection	Set the <i>enabled</i> attribute to "true" or "false" to turn this feature on or off.
MaxRate	The maximum rate in Kbps that the server sends data to the client. The default value is -1, which sends the data at whatever rate is necessary to measure bandwidth.
DataSize	The amount of data in bytes that the server sends to the client. To detect the client's bandwidth, the server attempts to send a series of random blocks of data to the client, each time sending this much more data. For example, x bytes are sent, followed by 2x bytes, followed by 3x bytes, and so on until <i>MaxWait</i> time has elapsed.
MaxWait	The number of seconds the server sends data to the client. Increasing this number provides a more accurate bandwidth figure, but it also forces the client to wait longer.

4 Save and validate the *Application.xml* file.

5 Restart the server.

Configuring content storage

Setting the location of application files

The presence of an application folder within the applications folder tells the server that the application exists. By default, the applications folder is located at *rootinstall/applications*.

Within the applications folder, create subfolders for your applications. Within each individual application folder, create subfolders to create instances of applications. For example, the following path is to an instance of an application called "my_application":

rootinstall/applications/my_application/first_instance

To change the location of the applications folder and the live and vod applications, edit the locations in the following parameters in the *ams.ini* file:

- Registered applications folder: *VHOST.APPSDIR*
- Live application: *LIVE_DIR*

- Vod application: `VOD_DIR`

Mapping directories to network drives

By default, the server runs as System Account with no access to network drives. You can change the service user to a user with network access privileges with a UNC path.

A Windows network-mapped drive is not valid when a user is logged out. If the server is running as a service and the user is logged out, the mapped drive is removed as well. To run with the mapped drive, lock the server instead of logging out. Using the UNC path is preferred when the server is running as a service.

- 1 Stop Adobe Media Server and Adobe Media Administration Server.
- 2 Make the changes to the configuration.
- 3 Check that the server user has appropriate access rights to map to the network drive (system account rights are usually not sufficient.)
- 4 Restart AdobeMedia Server and Adobe Media Administration Server.

Setting the location of recorded streams and shared objects

By default, all recorded streams for an application are stored in a streams folder in the application directory. Shared objects are stored in a shared objects folder in the application directory.

Note: Adobe strongly recommends that folders that store streams always contain only streams and no other application files.

Use the `StorageDir` tag in the `Application.xml` file to specify a different location to store streams or shared objects. You could do this for vod applications. For example, if you already have a collection of video files in a directory other than the application directory, you can set the storage directory to that other directory instead of copying content to the application directory.

When you specify a value for the `<storageDir>` element in the application-specific XML, that value is specific to the application. Otherwise, when you specify a value in the virtual host-level `Application.xml`, the scope is extended to all the applications on that virtual host.

Within the directory that you specify as the storage directory, you must create physical subdirectories for different application instances. Adobe Media Server sandboxes the content for each instance of an application.

Let's say, for example, you set the storage directory to `C:\Content` for the `chatApp` application:

```
<storageDir>C:\Content</storageDir>
```

When a user connects to the `firstRoom` instance of the `chatApp` application and tries to play a stream, the server looks for the stream in a subfolder `C:\Content\firstRoom`. Content for each instance is sandboxed from other instance of the same application; a user who connects to the `secondRoom` instance would not be able to access the content in `C:\Content\firstRoom`.

If you do not want resources to be sandboxed by application and application instance, use virtual directories.

Mapping virtual directories to physical directories

Adobe Media Server stores recorded streams and video and audio files in default locations in the application directory. In some scenarios, you might want to specify particular locations for these resources, but without restricting access by application or application instance. By mapping a virtual directory to a physical directory, you do not need to copy resources to Adobe Media Server's application directory, and you can retain your existing classification and categorization of resources.

To map a virtual directory for an application, you can use the `<VirtualDirectory>` element in the `Vhost.xml` or the `Application.xml` file. This element provides various options:

- You can specify a virtual directory name or not. When a name is specified, the server maps the name to the specified directory and first looks for the stream in the specified directory.
- You can specify multiple physical directories for a single virtual directory. Use multiple `<Streams>` tags.
- When specified in an application-specific `Application.xml` file, `<VirtualDirectory>` controls only the storage location of resources for that application. Any instance of the application can access video files in that location (unlike with `<storageDir>`), but other applications cannot.
- When specified in the virtual-host `Application.xml` file or the `Vhost.xml` file, `<VirtualDirectory>` controls the storage location of all applications on that virtual host. All applications on the virtual host can access video files in the specified location, although Adobe recommends that if you want control at the virtual host level, you configure the `<VirtualDirectory>` tag in `Vhost.xml` file instead of the virtual-host `Application.xml` file.

If you are creating a new virtual directory for a File object (as opposed to a stream), you must also enable the `VirtualDirectoryForFile` option in the `Server.xml` file. For more information, see [“Enable virtual directory mappings for server-side File objects”](#) on page 44.

The order in which the server determines the correct directory to use for streams is as follows:

- 1 Virtual directory (as specified in `<VirtualDirectory>`)
- 2 Storage directory (as specified in `<storageDir>`)
- 3 Default location (the streams folder in the application directory)

Note: Adobe strongly recommends that folders that store streams always contain only streams and no other application files.

Virtual directory example: vod

One usage scenario for this element is to specify a directory to use for a specific vod application and put video files in this directory to stream them instantly. You would use the `<VirtualDirectory>` element in the application-specific `Application.xml` file. To map a directory in this way, edit the application-specific `Application.xml` file to include the virtual directory, as shown in the following example:

```
<Application>
  <StreamManager>
    <VirtualDirectory>
      <!-- Specifies application specific virtual directory mapping for recorded
      streams. -->
      <Streams>/;C:\my_videos</Streams>
    </VirtualDirectory>
  </StreamManager>
</Application>
```

This code overrides the `VHost.xml` file's mapping of `/` (if it exists) for this application only. To play a file in the virtual directory, such as `C:\my_videos\sample.flv`, a client connects to the vod application and calls `NetStream.play()`:

```
ns.play("sample");
```

The `<VirtualDirectory>` element in the application-specific `Application.xml` file affects only that application, protecting your streams from being accessed by other applications on the same virtual host. It overrides settings in the `Vhost.xml` file.

Playing streams nested in subfolders

To play a stream, specify the codec before you specify the path to the stream. FLV streams don't require a codec prefix, but F4V/MP4 files, MP3 files, and RAW files do.

Suppose the virtual directory mapping is `<Streams>foo;C:\my_videos</Streams>`. The following examples are for the client-side `NetStream.play()` method:

```
ns.play("mp4:foo/sample.f4v",0,-1) // F4V/MP4 files
ns.play("raw:foo/sample",0,-1) // RAW files
ns.play("mp3:foo/sample",0,-1) //MP3 files
ns.play("foo/sample",0,-1). // FLV files
```

The following are examples for the Server-Side ActionScript `Stream.play()` method:

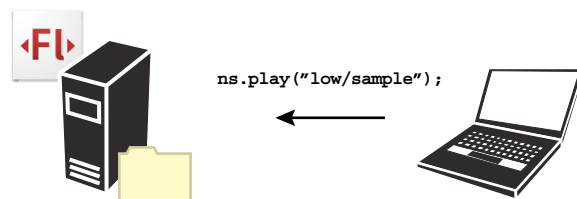
```
myStream.play("mp4:foo/sample.f4v",0,-1) // F4V/MP4 files
myStream.play("mp4:foo/sample",0,-1) // RAW files
myStream.play("mp3:foo/sample",0,-1) // MP3 files
myStream.play("foo/sample",0,-1) // FLV files
```

Virtual directory example: Separating high- and low-bandwidth video

One way you can use directory mapping is to separate storage of different kinds of resources. For example, your application could allow users to view either high-bandwidth video or low-bandwidth video, and you might want to store high-bandwidth and low-bandwidth video in separate folders. You can create a mapping wherein all streams that start with *low* are stored in specific directories, `C:\low_bandwidth` and `C:\low_bandwidth2`, and all streams that start with *high* are stored in a different directory:

```
<VirtualDirectory>
  <Streams>low;c:\low_bandwidth</Streams>
  <Streams>low;c:\low_bandwidth2</Streams>
  <Streams>high;c:\high_bandwidth</Streams>
</VirtualDirectory>
```

When the client wants to access low-bandwidth video, the client calls `ns.play("low/sample")`. This call tells the server to look for the `sample.flv` file in the `C:\low_bandwidth` and `C:\low_bandwidth2` folders.



`C:\low_bandwidth\sample.flv`

A client connects to the `sample.flv` file in the low-bandwidth storage area on the server, which is mapped in `Application.xml`.

Similarly, a call to `ns.play("high/sample")` tells the server to look for the `sample.flv` file in the `c:\high_bandwidth` folder.

Important: To play an F4V/MP4, MP3, or RAW file, specify the codec before the path, for example `ns.play("mp4:high/sample")`. The mp3 prefix is `mp3:`. The RAW prefix is `raw:`.

Note that if the client calls `ns.play("sample")`, the stream name does not match any virtual directory specified, so the server will then look for `sample.flv` inside the directory specified by the storage directory element (`<storageDir>`). If no storage directory is specified by `<storageDir>`, then the server looks in the default location (the streams folder) for the file; that is, the order in which the server looks for files is:

- 1 Virtual directory (as specified in `<VirtualDirectory>`)
- 2 Storage directory (as specified in `<storageDir>`)
- 3 Default location (the streams folder in the application directory)

Virtual directory example: Local and network file paths

The following table shows three examples of different virtual directory configurations, including mapping to a network drive, and how the configurations determine the directory to which a recorded stream is published. In the first case, because the URI specified (`"myStream"`) does not match the virtual directory name that is specified (`"low"`), the server publishes the stream to the default streams directory.

Mapping in Vhost.xml	URI in NetStream call	Location of published stream
<code><VirtualDirectory><Streams></code> tag		
<code>low:e:\amsstreams</code>	<code>"myStream"</code>	<code>c:\...\rootinstall\applications\yourApp\streams_definst_\myStream.flv</code>
<code>low:e:\amsstreams</code>	<code>"low/myStream"</code>	<code>e:\amsstreams\myStream.flv</code>
<code>low:\\mynetworkDrive\share\amsstreams</code>	<code>"low/myStream"</code>	<code>\\mynetworkDrive\share\amsstreams\myStream.flv</code>

Important: To play an F4V/MP4, MP3, or RAW file, specify the codec before the path, for example `ns.play("mp4:low/myStream")`. The mp3 prefix is `mp3:`. The RAW prefix is `raw:`.

Configuring Apache HTTP Server

About Apache HTTP Server

Adobe Media Server includes Apache HTTP Server. If you install and enable the web server, you can deliver client SWF files, container HTML pages, and media assets from Adobe Media Server. You can serve content over HTTP, as well as RTMP. In addition, you can serve video over HTTP progressive download as a fallback solution for web proxies that break RTMP or RTMPT.

The Adobe Media Server installation of Apache is like the standard installation. You can use Apache documentation for most configuration tasks. The Apache root installation folder is `rootinstall/Apache2.2`. The web root is `rootinstall/webroot`.

Note: The Adobe Media Server documentation uses “`rootinstall`” to indicate the Adobe Media Server root installation folder (C:\Program Files\Adobe\Adobe Media Server 4 by default on Windows). The Apache documentation and configuration files use “`ServerRoot`” to indicate the Apache root installation folder.

Installation locations

The Apache server that installs with Adobe Media Server differs from a standard Apache installation in the following ways:

- The `apachectl` files are in the `rootinstall/Apache2.2/manual/programs` folder.
- The `httpd.conf` file and the standard secondary configuration files are in the `rootinstall/Apache2.2/conf` folder.

- The log files are in the *rootinstall*/Apache2.2/logs folder. For more information, see “[Monitoring and Managing Log Files](#)” on page 97.

Alternate configurations

To do any of the following, edit the *rootinstall*/Apache2.2/conf/httpd.conf file:

- Handle multiple adaptors or virtual hosts
- Block sensitive file types, such as ASC
- Handle any non-standard MIME types required for progressive download
- Serve the default server-status and server-info pages
- Modify source directories

Specify the maximum HTTP header line length

In the Adobe Media Server Adaptor.xml file, the `MaxHeaderLineLength` element determines the size of the HTTP header the server can handle. The default value for `MaxHeaderLineLength` is 1024 bytes. Some browsers send a header larger than 1024 bytes. In this scenario, Apache sends back an empty response. To fix this issue, configure `MaxHeaderLineLength` to 8192.

Note: By default, the Apache HTTP header size limit is 8 KB (8190 bytes plus a carriage return).

- 1 Open *rootinstall*/conf/Adaptor.xml in a text editor.
- 2 Edit the following line:

```
<MaxHeaderLineLength>8192</MaxHeaderLineLength>
```
- 3 Save the file.
- 4 Restart the server.

Enable and disable Apache and HTTP proxying

When you install Adobe Media Server and choose to install Apache, it is enabled by default. Adobe Media Server listens on port 80 and proxies HTTP requests to Apache over port 8134.

Setting the parameter to no value disables HTTP proxying:

```
HTTPPROXY.HOST =
```

To proxy to a remote server, set the parameter to the remote server you want to use:

```
HTTPPROXY.HOST = webfarm.example.com:80
```

Enable Apache and HTTP proxying

- 1 Open the *ams.ini* file in a text editor.
- 2 Add port 80 to the `ADAPTOR.HOSTPORT` tag.

```
ADAPTOR.HOSTPORT = :1935, 80
```
- 3 To start and stop Apache when Adobe Media Server starts and stops, set the `SERVER.HTTPD_ENABLED` parameter to `true`:

```
SERVER.HTTPD_ENABLED = true
```

4 Do one of the following:

- Open *rootinstall/conf/ams.ini* and set the value of the `HTTPPROXY.HOST` variable to 8134:

```
HTTPPROXY.HOST = :8134
```

- Open the *Adaptor.xml* configuration file in a text editor.

If you have multiple adaptors, open the file for the adaptor you want to configure. The default *Adaptor.xml* file is located in the *rootinstall/conf_defaultRoot_* folder.

Set the `enable` attribute of the `HttpProxy` tag to "true", as in the following:

```
<HttpProxy enable="true" maxbuf="16384">  
  <Host port="80">${HTTPPROXY.HOST}</Host>  
</HttpProxy>
```

Save and validate the *Adaptor.xml* file.

5 Restart the server.

Disable Apache and HTTP proxying

1 Open the *ams.ini* file in a text editor.

2 Remove port 80 from the `ADAPTOR.HOSTPORT` tag.

```
ADAPTOR.HOSTPORT = :1935
```

3 Do one of the following:

- Open *rootinstall/conf/ams.ini* and set the value of the `HTTPPROXY.HOST` variable to no value:

```
HTTPPROXY.HOST =
```

- Open the *Adaptor.xml* configuration file in a text editor.

If you have multiple adaptors, open the file for the adaptor you want to configure. The default *Adaptor.xml* file is located in the *rootinstall/conf_defaultRoot_* folder.

Set the `enable` attribute of the `HttpProxy` tag to "false", as in the following:

```
<HttpProxy enable="false" maxbuf="16384">  
  <Host port="80">${HTTPPROXY.HOST}</Host>  
</HttpProxy>
```

Save and validate the *Adaptor.xml* file.

4 Restart the server.

Deliver SWF files and HTML files over HTTP

Apache can deliver SWF files, HTML files, JPG files, and many other standard file types over HTTP. Place files that you want to deliver over HTTP in the correct folders. By default, Apache is configured to use following paths:

Path	Location	Example URL
/	<i>rootinstall</i> /Apache2.2/webroot	http://myAMSServer.com/app.swf
/cgi-bin	<i>rootinstall</i> /Apache2.2/cgi-bin	http://myAMSServer.com/cgi-bin/someScript.pl

These folders are global, not application-specific. Any SWF files and HTML files you want to serve over HTTP must be in these folders or in application-specific folders. To create aliases for application-specific folders, edit the *httpd.conf* file.

Create web directories for applications

To have Apache serve content over HTTP, either use one of the predefined web directories or create a web directory for that application. To create a web directory for an application, add an alias to the `rootinstall/Apache2.2/conf/httpd.conf` file. For example, the following lines create an alias that points to the streams directory of an application called “hd”:

```
Alias /hd/ "applications/hd/streams/_definst/"
<Directory "applications/hd/streams/_definst/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Copy media files to the `rootinstall/applications/hd/streams/_definst/` folder to serve them over HTTP. For example, if a client requests `http://ams.example.com/hd/someVideo.flv`, Apache serves it the `someVideo.flv` file from the `rootinstall/applications/hd/streams/_definst/` folder.

For more information about editing the configuration file, see the Apache documentation at www.apache.org.

Using Apache as a WebDAV host

You can configure the web server to act as a WebDAV host. For information, see the Apache HTTP Server version 2.2.15 documentation at www.apache.org.

The following keywords are reserved words: open, close, send, idle, fcs, ams, crossdomain.xml, fpad. Reserved words cannot be used for directory names under the webroot directory or for anything else. If you have applications with these names, use custom aliases to give the HTTP virtual directories slightly different names from the RTMP applications, such as “open_”.

Running Apache for Windows as a service

To install Apache to run as a service on Windows:

- 1 Open a Command prompt.
- 2 Change directories to the `rootinstall/Apache2.2/bin` folder. For example, at the command prompt, enter the following:

```
cd ../../Program Files/Adobe/Adobe Media Server 4/Apache2.2/bin
```

- 3 To install Apache to run as a service, enter the following:

```
httpd.exe -f conf/httpd.conf -n AMSHtpd -k install
```

Note: To uninstall Apache, enter the following: `httpd.exe -n AMSHtpd -k uninstall`.

- 4 To verify that Apache is running, open the Windows Services panel and locate the AMSHtpd service.

To start and stop the Windows services on Windows XP:

- 1 Right click My Computer and select Manage. The Computer Management dialog box displays.
- 2 Select Services and Applications.
- 3 Double-click Services. The list of available services displays.
- 4 To start a service, select the service from the list and click the Start Service button. To stop a service, select the service from the list and click the Stop Service button.

Upgrading Apache HTTP server

At some time after installing Adobe Media Server, you might be required to update the included Apache web server. Most commonly, this would be due to Apache HTTP server security patches or functionality upgrades.

The version of the Apache HTTP server that ships with Flash Media Server 4.5 is 2.2.17. For the latest information about the Apache web server, see <http://httpd.apache.org>.

The recommended approach to upgrading the Apache web server is to install or build the Apache server in a directory that is separate from the existing server. Then copy the contents of newly-installed directories to the existing server.

In most cases, you do not replace the existing Apache configuration files with the newer versions of these files. This is because the Apache web server installation for Adobe Media Server is pre-configured with specific modules and settings. If there are changes to the configuration files, it is usually easier to update the existing files with changes.

Before upgrading:

- 1 Review any installation notes on apache.org about the differences between the current version and the new version. For example, look for renamed directives or renamed modules that would require you to merge configuration files.

- 2 Make sure the currently-installed version of Apache is working correctly:

- 1 Check the master log file for errors. The master.00.log file should have a line that launches the httpd process and returns 0; for example:

```
- C:\Program Files\Adobe\Adobe Media Server 5\Apache2.2\bin\httpd
-f ./conf/httpd.conf -d "C:\Program Files\Adobe\Adobe Media Server 5\Apache2.2"
-n AMSHttpd -k start returned 0:
```

- 2 Try playing videos using the “http” protocol.

- 3 Check the current version of Apache and make a note of it. From a command prompt, go to *AMS_Install_Dir*/Apache2.2/bin and issue the following command:

```
httpd -v
```

- 4 Back up following directories:

```
AMS_Install_Dir/Apache2.2/bin
AMS_Install_Dir/Apache2.2/lib
AMS_Install_Dir/Apache2.2/include
AMS_Install_Dir/Apache2.2/modules
AMS_Install_Dir/Apache2.2/manual
AMS_Install_Dir/Apache2.2/icons
```

To upgrade:

- 1 Stop Adobe Media Server.
- 2 Follow the directions on apache.org to download and run the Apache installer. If you are using Linux, you may be required to build the Apache server on your system.
- 3 Copy the following directories from the newly-installed version to the current version that was installed with Adobe Media Server:

```
Apache_Install_Dir/bin
Apache_Install_Dir/modules
Apache_Install_Dir/manual
Apache_Install_Dir/icons
```

Note that the /conf directory, which contains the Apache configuration files, is not included in this list. You should not replace the existing configuration files with the new configuration files.

- 4 Merge the configuration files, if necessary. In most cases, you will not need to merge the files. However, if the newer version of Apache includes new or renamed modules or directives, you might need to edit the configuration files to meet the new requirements.
- 5 Start Adobe Media Server.

After upgrading:

- 1 Ensure that the newer version of Apache is now available. From a command prompt, go to `AMS_Install_Dir/Apache2.2/bin` and issue the following command:

```
httpd -v
```
- 2 Recompile third-party modules, if necessary.
- 3 Make sure the newly-updated version of Apache is working correctly:
 - 1 Check the master log file for errors or warnings. The master.00.log file should have a line that launches the httpd process and returns 0; for example:

```
- C:\Program Files\Adobe\Adobe Media Server 4\Apache2.2\bin\httpd -f  
./conf/httpd.conf -d "C:\Program Files\Adobe\Adobe Media Server 4\Apache2.2" -n  
AMSHttpd -k start returned 0:
```
 - 2 Try playing videos using the “http” protocol.

Updating the OpenSSL version in the bundled Apache server

On a Windows Machine, OpenSSL version 0.9.8r is available with bundled Apache. To update the OpenSSL version, follow these steps:

- 1 Access the `bin` directory of your Apache installation.
- 2 Copy `libeay32.dll`, `ssleay32.dll`, and `openssl.exe` files from a separate, patched security release of OpenSSL from the same major release.
- 3 Replace the existing files with the new files in the `bin` directory.

On Linux machines, the bundled Apache does not include the OpenSSL libraries `libcrypto` and `libssl`. Adobe recommends the administrators keep the system updated with the appropriate version of OpenSSL binaries.

Troubleshooting Apache HTTP server

In some cases, the Apache server could be at fault for the Adobe Media Server not running properly. This is sometimes caused by syntax errors in the Apache configuration files.

Check the master.log file in the `rootinstall/logs` directory. If Apache could not start for some reason, it typically returns an error code of “1” and writes a line similar to the following in the master.log file:

```
2010-06-22 15:45:09 4276 (w)2581414 C:\Program Files\Adobe\Adobe Media Server 4  
\Apache2.2\bin\httpd -f ./conf/httpd.conf -d "C:\Program Files\Adobe\Adobe Media Server 4  
\Apache2.2" -n AMSHttpd -k start returned 1:
```

The status of (w) or (e) indicates a warning or error. For more information about status codes in the master.log file, see “[Diagnostic logs](#)” on page 106.

If you are running Adobe Media Server as a service, you can check the [Windows Event Viewer](#) for error messages. On Linux, check `syslog`’s log file for error messages.

You can also check the Apache log files for error messages. The default location of the Apache log files is *rootinstall*/Apache2.2/logs. The logs are in the default Apache error and combined access log formats. For information about these log file formats, see [Log Files](#).

To change the location of the log files, edit the *rootinstall*/Apache2.2/conf/httpd.conf file.

Configure Apache for Adobe HTTP Dynamic Streaming and Apple HTTP Live Streaming

Configure the HTTP modules

Apache HTTP Server that installs with Adobe Media Server is configured by default for Adobe HTTP Dynamic Streaming (to Flash Player and AIR) and Apple HTTP Live Streaming (to iOS and Mac OS). The Adobe Media Server Apache installation includes three custom modules that handle HTTP streaming:

- `jithttp_module`—just-in-time on-demand Adobe HTTP Dynamic Streaming (HDS).
- `f4fhttp_module`—live HDS; on-demand HDS packaged offline
- `hlshttp_module`—live and on-demand Apple HTTP Live Streaming

The configuration of the modules determines the format of the URL that the player requests from the server, the location of the content, the content fragmentation settings, and other settings. Configure the settings at the server-level in the Apache `httpd.conf` file located at *rootinstall*/Apache2.2/conf/httpd.conf. You can also configure some settings at the application level, event level, and stream level. For complete details, see [Configuring HTTP Dynamic Streaming and HTTP Live Streaming](#) in the *Adobe Media Server Developer's Guide*.

For information about configuring ports for HTTP Streaming, see “[Configure ports for HTTP streaming](#)” on page 6.

Logging for HTTP streaming

All HTTP access requests are logged in the Apache `access_log` file in the `\logs` directory. The HTTP Origin Module doesn't add any logging directives. All access requests are logged through the standard Apache HTTP Server log module.

All errors are logged in the Apache `error_log` file in the `\logs` directory. The following table shows the response codes that the HTTP Origin Module returns.

Response code	Error message	Description
200	None	No error occurs, and a response has been returned successfully.
304	None	The HTTP Origin Module supports "If-Modified-Since" in the request header and returns 304 if the file has not been modified."If-Modified-Since" is not support in requests for manifest files of live events.
400	<code>mod_f4fhttp [400]: Syntax error in %s</code> <code>mod_hlshttp [400]:Syntax error in %s</code>	Error occurs while parsing the URI request. The variable %s is the URI string. For example, the request URI <code>http://servername/media/sample-</code> is missing a fragment number. For example, in the request <code>http://example.com/hls-vod/sample.f4vNumX.ts</code> , the <code>ts</code> fragment name format is not correct.

Response code	Error message	Description
400	mod_hlshttp [400]: %1 is a invalid file	Error occurs while reading the information in a file, where %1 is the URI string. For example, the manifest file is wrong or the bootstrap is invalid or the .stream file is not valid.
403	mod_f4fhttp [403]: No access to %s mod_hlshttp[403]: No access to %s mod_f4fhttp [403]: Internal error %1 (%2) when processing %3 mod_hlshttp[403]: Internal error %1 (%2) when processing %3	<p>An error occurs when accessing the file. The variable %s is the file path. For example, the request URI is http://servername/media/sample.fmf and the module doesn't have permission to read the file sample.f4m.</p> <p>An error occurs when the library fails to access a file. The variable %1 is the error code from the library, %2 is error description, and %3 is the url string.</p>

Response code	Error message	Description
404	<p>mod_f4fhttp [404]: %s does not exist</p> <p>mod_hlshttp[404]: %1 does not exist</p> <p>mod_f4fhttp [404]: Internal error %1 (%2) when processing %3</p> <p>mod_hlshttp[404]: Internal error %1 (%2) when processing %3</p>	<p>An error occurs when the request file is not found. The variable %s is the file path. For example, the request URI is http://servername/media/sample.fmf but the file sample.f4m does not exist.</p> <p>Error occurs when the library fails to find a file. The variable %1 is the error code from the library, %2 is the error description, and %3 is the url string.</p>
500	<p>mod_f4fhttp [500]: Unknown exception when processing %1</p> <p>mod_hlshttp [500]: Unknown error when processing %1</p> <p>mod_f4fhttp [500]: Internal error %1 (%2) when processing %3</p> <p>mod_hlshttp [500]: Internal error %1 (%2) when processing %3</p> <p>mod_hlshttp[500]: Packaging error %1 (%2) when processing %3</p>	<p>An unknown error occurs when processing a request, where %1 is the URL string. For example, http://servername/media/sampleSeg1-Frag1, sampleSeg1.f4f is not a corrupted file.</p> <p>An unknown error occurs when processing a request, where %1 is the URI string. The module fails, where %1 is the error code, %2 is error description, and %3 is the URI string.</p> <p>Error occurs while packaging a ts segment, where %1 is the error code for diagnostic purposes, %2 is the error description and %3 is the URI being processed. An example of an error description is, "Audio format is not supported".</p> <p>A packaging error indicates that something went wrong while packaging the stream data into a ts segment. An internal error indicates that something went wrong while reading stream data from the recorded f4f fragments. An internal error is more likely to occur due to a recording failure than to a play back failure.</p>
503	None	<p>The HTTP Origin Module returns a 503 code so the client and the proxy servers don't cache the response.</p> <p>In a live event, the file can be growing and a fragment could be missing in one origin server but not missing on another origin server. The HTTP response header includes an "X-mod_f4fhttp" field to provide more details. The following is a sample response header:</p> <pre>Date Thu, 04 Mar 2010 06:20:20 GMT Server Apache/2.2.9 (Win32) Content-Length 432 Last-Modified Wed, 27 Jan 2010 11:43:25 GMT Keep-Alive timeout=5, max=100 Connection Keep-Alive Content-Type video/f4f X-mod_f4fhttp 1, error 71</pre> <p>There are two parameters in the X-mod_f4fhttp field delimited by a comma. The first parameter is a status code and the second parameter is the error code from the library.</p> <p>There are two status codes: 1 - The requested fragment number is larger than the last fragment in the file (the corresponding error is 71). 2 - The requested fragment number is a missing fragment (the corresponding error is 34).</p>

Use an external Apache HTTP Server for HTTP Dynamic Streaming and HTTP Live Streaming

To use an installation of Apache HTTP Server other than the one that installs with Adobe Media Server, do the following:

- 1 Install Adobe Media Server so you can extract the Apache modules.
- 2 Copy the following files from Adobe Media Server to your Apache modules folder:

- *rootinstall*/Apache2.2/modules/adbe_dme.dll
- *rootinstall*/Apache2.2/modules/adbe_license.dll
- *rootinstall*/Apache2.2/modules/asneu.dll
- *rootinstall*/Apache2.2/modules/hds.dll
- *rootinstall*/Apache2.2/modules/libeay32.dll
- *rootinstall*/Apache2.2/modules/libexpat.dll
- *rootinstall*/Apache2.2/modules/mod_f4fhttp.so
- *rootinstall*/Apache2.2/modules/mod_hlshttp.so
- *rootinstall*/Apache2.2/modules/mod_jithttp.so

- 3 Edit your *httpd.conf* file to load the following modules:

- For live Adobe HTTP Dynamic Streaming, add the following:

```
LoadModule f4fhttp_module modules/mod_f4fhttp.so
<IfModule f4fhttp_module>
<Location /hds-live>
    HttpStreamingEnabled true
    HttpStreamingLiveEventPath "../applications"
    HttpStreamingContentPath "../applications"
    HttpStreamingF4MMMaxAge 2
    HttpStreamingBootstrapMaxAge 2
    HttpStreamingFragMaxAge -1
    HttpStreamingDrmmetaMaxAge 3600
    Options -Indexes FollowSymLinks
</Location>
</IfModule>
```

- For live Apple HTTP Live Streaming, add the following:

```
LoadModule hlshttp_module modules/mod_hlshttp.so
<IfModule hlshttp_module>
<Location /hls-live>
    HLSHttpStreamingEnabled true
    HttpStreamingLiveEventPath "../applications"
    HttpStreamingContentPath "../applications"
    HLSMediaFileDuration 8000
    HLSSlidingWindowLength 6
    HLSAmsDirPath ".."
    HLSM3U8MaxAge 2
    HLSTSSegmentMaxAge -1
    Options -Indexes FollowSymLinks
</Location>
```

- For on-demand Adobe HTTP Dynamic Streaming, packaged in just-in-time, add the following:

```
LoadModule jithttp_module modules/mod_jithttp.so
<IfModule jithttp_module>
<Location /hds-vod>
    HttpStreamingJITPEnabled true
    HttpStreamingContentPath "../webroot/vod"
    JitAmsDirPath ".."
    Options -Indexes FollowSymLinks
</Location>
</IfModule>
```

- For on-demand Apple HTTP Live Streaming, add the following:

```
LoadModule hlshttp_module modules/mod_hlshttp.so
<IfModule hlshttp_module>
<Location /hls-vod>
    HLSHttpStreamingEnabled true
    HLSMediaFileDuration 8000
    HttpStreamingContentPath "../webroot/vod"
    HLSAmsDirPath ".."
    Options -Indexes FollowSymLinks
</Location>
</IfModule>
```

- 4 In the directives that you entered in your httpd.conf file, edit the following:

- `HttpStreamingLiveEventPath` and `HttpStreamingContentPath`—set to the Adobe Media Server applications folder, for example:

```
C:\Program Files\Adobe\Adobe Media Server 5\applications
```

- `JitAmsDirPath` and `HLSAmsDirPath`—set to the location of your Adobe Media Server installation, for example:

```
C:\Program Files\Adobe\Adobe Media Server 5
```

- 5 For information about ports, see [“Using a third-party web server”](#) on page 76.

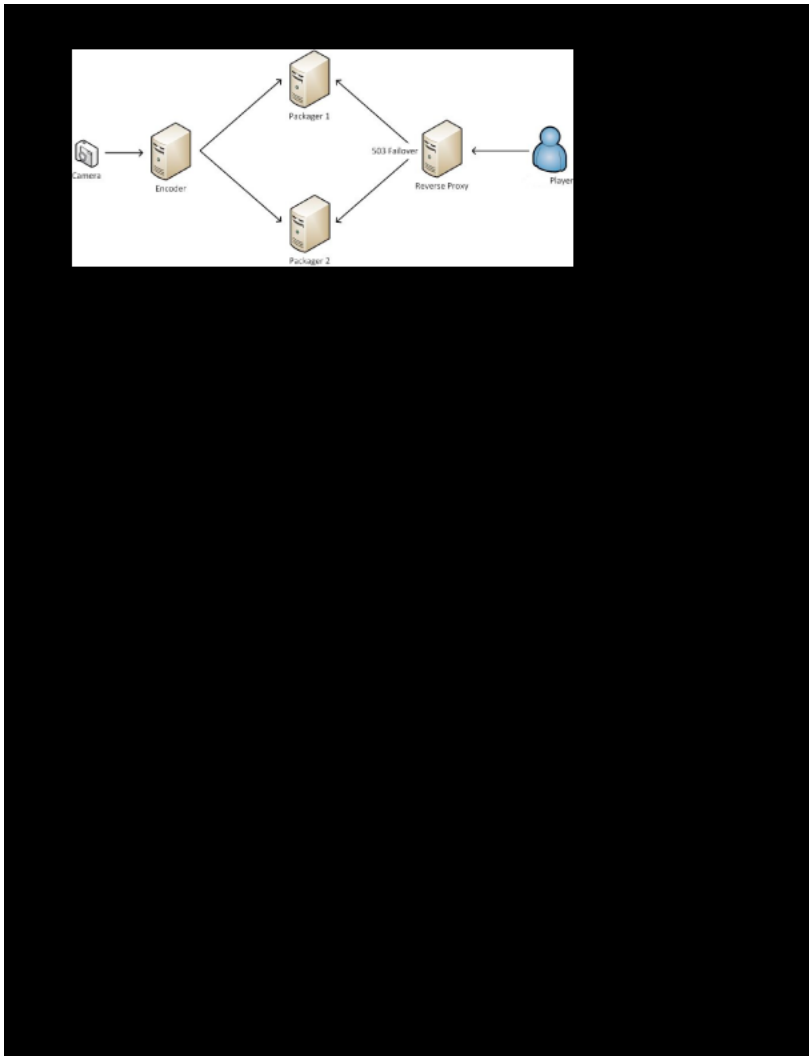
Configure HTTP Streaming failover

In a real-world clustered HTTP Streaming scenario, you want your site to offer robust playback, even in the face of server-side streaming problems. HTTP Streaming failover, new in FMS 4.5.2, lets you control how AMS handles certain streaming problems.

Overview of HTTP Streaming failover

HTTP Streaming failover features primarily address the issues of liveness and dropout.

The following figure illustrates a simple two-server cluster of packagers with a Reverse Proxy controlling access. The explanations that follow use this figure as a reference.



Packagers in a two-server cluster

- **Liveness** is a server-side situation in which a packager advertises a stale bootstrap (that is, a stale view of a live stream). Although there are other potential scenarios, the following describes one liveness situation:
 - 1 The stream has been running normally for some time on the server side.
 - 2 Packager 2's connection to the encoder fails. The connection between Packager 2 and the proxy continues running. Suppose that at the time of this failure, the last successfully written fragment on Packager 2 was still #700 with current media time as X.
 - 3 A short amount of time elapses, in this case, 10 fragment intervals.
 - 4 The server-side now suffers from liveness. Packager 2 is serving stale bootstrap information. Packager 2 indicates that there has been no update in the bootstrap after time X, even though more recent fragments are available on Packager 1 (that is, the current media time is greater than X.)
 - 5 Playback stalls at the player end.

- **Dropout** - is a server-side situation in which a packager has gaps in its bootstrap (that is, gaps in its fragment list). The following scenario describes a dropout situation:
 - 1 The stream has been running normally for some time on the server side.
 - 2 Packager 2's connection to the encoder fails. The connection between Packager 2 and the proxy continues running. Suppose that at the time of this failure, the last successfully written fragment on Packager 2 was still #700.
 - 3 A short amount of time elapses, in this case, 10 fragment intervals.
 - 4 Packager 2's encoder connection is restored and remains restored for a long time.
 - 5 The server-side now suffers from dropout. Packager 2's bootstrap is missing entries for #701-710, even though those fragments are still available on Packager 1.
 - 6 Playback stalls at the player end.

HTTP Streaming failover provides a multi-faceted solution:

- Best-effort fetch
- Server-side configuration
- Control plane management

Enable best-effort fetch

Best-effort fetch enables the OSMF and iOS video players to continue playback as normally as possible in the presence of short-term liveness and dropout problems on the server-side.

OSMF

The OSMF 2.0 player includes HDS functionality that integrates with server-side configuration and run-time settings. For more information on OSMF 2.0 client side functionality, see the [OSMF documentation](#) on SourceForge.

HLS

FMS 4.5.2 includes server-side settings and functionality that enable best-effort fetch.

Configure HTTP Streaming failover server settings

You configure server settings for HTTP Streaming failover in the `httpd.conf` and `manifest.xml` files.

Configure HDS and HLS settings in `httpd.conf`. For more information on server-side configuration in the `httpd.conf` file, see [Configure live and on-demand HTTP Streaming at the server level \(httpd.conf\)](#). Configure the `bestEffortFetchInfo` element in the `manifest.xml` file, as described in [Manifest.xml](#).

Note: *Best-effort fetch relies on configuring the back-end reverse-proxy with HTTP 503 failover, meaning that when the reverse proxy receives a 503 response from a packager, it retries the request with the next packager in the pool. If all packagers generate a 503, the proxy generates an error that it caches for less than a fragment interval. You can configure the HTTP error code, as described in [Configure live and on-demand HTTP Streaming at the server level \(httpd.conf\)](#).*

Code the control plane application

To implement HTTP Streaming failover, you write a client application that manages the state of events and streams by using a set of REST-based control plane APIs. Control plane is a router term and in effect, that is what your client application does through these APIs.

The basic workflow is as follows:

- Begin repeatedly checking stream status at a set interval. The exact interval depends on your configuration.
- Disable any stream with a bad status code.
- You could also compare the age of the bootstrap file and disable the stream for which the bootstrap is older than some predetermined interval.

AMS control plane REST interface request path

The AMS control plane REST interface request path differs, depending on whether you are using HDS, HLS with adaptive bit-rate (ABR), or HLS with a single bit-rate stream:

- HDS: `http://server/ctrlplane/application-name/instance-name/event-name.f4m/`
- HLS with ABR: `http://server/ctrlplane/application-name/instance-name/event-name.m3u8/`
- HLS with single bit-rate: `http://server/ctrlplane/application-name/instance-name/event-name/stream-name.m3u8/`

For example:

- HDS: `http://www.example-server.com/ctrlplane/livepkgr/_definst_/liveevent.f4m/`
- HLS with ABR: `http://www.example-server.com/ctrlplane/livepkgr/_definst_/liveevent.m3u8/`
- HLS with single bit-rate: `http://www.example-server.com/ctrlplane/livepkgr/_definst_/liveevent/livestream.m3u8/`

You use HTTP GET or POST, depending on the endpoint and parameters passed.

Query stream status

Description

Returns stream status.

Endpoint

`status`

Result

If the `.disabled` file exists in the streams directory for the respective associated stream, the control plane module returns "disabled" in the status element for the particular stream. This endpoint returns stream age for each stream (multiple streams in case of MBR) calculated as `current time in minutes - last modified time for the bootstrap file`. The `up` element indicates whether the stream is up. It is set to `false` if the bootstrap has not been updated in specified number of seconds (configurable as `MaxBootstrapAge`). The global `up` element is set to `true` if all bootstrap `up` elements are true, otherwise it is set to `false`. The endpoint sets the upper level `status` to `disabled` if all streams are disabled, otherwise it is set to `enabled`.

The `done` element indicates whether the stream is finished.

Sample request

```
[GET] http://www.example-server.com:8134/ctrlplane/livepkgr/_definst_/livestream.f4m/status
```

Sample response

This is an example response for an MBR stream in which all streams are disabled:

200 OK

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-event-status xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkgr/_definst_/livestream.f4m
  </manifest-file>
  <status>
    disabled
  </status>
  <up>
    false
  </up>
  <done>
    false
  </done>
  <bootstrap name="livestream2 ">
    <up>
      false
    </up >
    <age>
      75
    </age>
    <status>
      disabled
    </status>
    <done>
      false
    </done>
  </ bootstrap>
  <bootstrap name="livestream 3">
    <up>
      false
    </up >
    <age>
      75
    </age>
    <status>
      disabled
    </status>
  </bootstrap>
</controlplane-event-status>
```

```
        </status>
        <done>
            false
        </done>
    </ bootstrap>
    <bootstrap name="livestream 1">
        <up >
            false
        </up>
        < age>
            75
        </age>
        <status>
            disabled
        </status>
        <done>
            false
        </done>
    </ bootstrap>
</controlplane-event-status>
```

Disable an event

Description

Disables all streams for an event (works for both HDS and HLS streams). You send this as an HTTP POST request with no body.

Endpoint

disable

Result

Upon receiving this request, the control plane module creates a `.disabled` file in all streams directories (multiple streams directories for an MBR stream) and returns an XML formatted response indicating success or failure for each stream.

If a client subsequently requests streams for the event, the Apache module (`mod_f4fhttp` or `mod_hlshttp`) checks for the existence of a `.disabled` file before processing the request. If a `.disabled` file exists, the module returns a 503 error (configurable in `httpd.conf`).

Sample request

```
[POST] http://www.example-
server.com:8134/ctrlplane/livepkgr/_definst_/livestream.f4m/disable
```


Sample response

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkgr/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream2">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream3">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream1">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Disable a stream

Description

Disables a stream (works for both HDS and HLS streams). You send this as an HTTP POST request with the stream name specified in the body.

Disable a stream if you detect a liveness problems or to take a faulty packager out of production for some other reason. As mentioned below, the Apache module returns a 503 error, making the faulty packager a non-entity.

Endpoint

disable

Result

Upon receiving this request, the control plane module creates a `.disabled` file in the streams directory and returns an XML formatted response indicating success or failure.

If a client subsequently requests the stream, the Apache module (`mod_f4fhttp` or `mod_hlshttp`) checks for the existence of a `.disabled` file before processing the request. If a `.disabled` file exists, the module returns a 503 error (configurable in `httpd.conf`).

Sample request

```
[POST] http://www.example-server.com:8134/ctrlplane/livepkg/_definst_/livestream.f4m/disable

<controlplane-request>
  <stream> livestream2</stream>
</controlplane-request>
```

Sample response

```
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkg/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream2">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Enable an event

Description

Enables all streams for an event (works for both HDS and HLS streams). You send this as an HTTP POST request with no body.

Endpoint

enable

Result

Upon receiving this request, the control plane module deletes any existing `.disabled` files in all streams directories (multiple streams directories for an MBR stream) and returns an XML formatted response indicating success or failure for each stream.

If a client subsequently requests streams for the event, the Apache module (`mod_f4fhttp` or `mod_hlshttp`) processes the request.

Sample request

```
[POST] http://www.example-server.com:8134/ctrlplane/livepkg/_definst_/livestream.f4m/enable
```

Sample response

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkg/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream2">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream3">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream1">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Enable a stream

Description

Enables a stream (works for both HDS and HLS streams). You send this as an HTTP POST request with the stream name specified in the body. This request enables a stream and removes the `.disabled` file.

Note: Streams are enabled by default.

Endpoint

enable

Result

Upon receiving this request, the control plane module deletes any existing `.disabled` file in the streams directories and returns an XML formatted response indicating success or failure.

If a client subsequently requests the stream, the Apache module (`mod_f4fhttp` or `mod_hlshttp`) processes the request.

Sample request

```
[POST] http://www.example-server.com:8134/ctrlplane/livepkg/_definst_/livestream.f4m/enable
```

```
<controlplane-request>
  <stream>livestream2</stream>
</controlplane-request>
```

Sample response

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkggr/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream2">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Mark an event as done

Description

Signals that an event is complete. You send this as an HTTP POST request with no body.

You signal an event as done so that the client is notified when content ends. This achieves two things: the control plane does not activate failover, and client applications can take appropriate action to signal the end of content to the user.

Endpoint

done

Result

Upon receiving this request, the control plane module creates a `.done` file in all streams directories (multiple streams directories for an MBR stream) and returns an XML formatted response indicating success or failure for each stream.

Sample request

```
[POST] http://www.example-server.com:8134/ctrlplane/livepkggr/_definst_/livestream.f4m/done
```

Sample response

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkggr/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream1">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream2">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream3">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Mark a stream as done

Description

Signals that a stream complete. You send this as an HTTP POST request with the stream name specified in the body.

You signal a stream as done so that the client is notified when content ends. This achieves two things: the control plane does not activate failover, and client applications can take appropriate action to signal the end of content to the user.

Endpoint

done

Result

Upon receiving this request, the control plane module creates a `.done` file in the associated streams directory and returns an XML formatted response indicating success or failure.

Sample request

```
[POST] http://www.example-server.com:8134/ctrlplane/livepkggr/_definst_/livestream.f4m/done
<controlplane-request>
  <stream name="livestream1"></stream>
</controlplane-request>
```

Sample response

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkgr/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream1">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Mark an event as in-progress

Description

Signals that an event is still in progress. You send this as an HTTP POST request with no body. This request undoes a /done request.

Endpoint

inProgress

Result

Upon receiving this request, the control plane module deletes any existing .done files in all streams directories (multiple streams directories for an MBR stream) and returns an XML formatted response indicating success or failure for each stream.

Sample request

```
[POST] http://www.example-
server.com:8134/ctrlplane/livepkgr/_definst_/livestream.f4m/inProgress
```

Sample response

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkgr/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream1">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream2">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
  <stream name="livestream3">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Mark a stream as in-progress

Description

Signals that a stream is still in progress. You send this as an HTTP POST request with the stream name specified in the body. This request undoes a /done request.

Endpoint

inProgress

Result

Upon receiving this request, the control plane module deletes any existing .done file in the associated streams directory and returns an XML formatted response indicating success or failure for each stream.

Sample request

```
[POST] http://www.example-
server.com:8134/ctrlplane/livepkgr/_definst_/livestream.f4m/inProgress

<controlplane-request>
  <stream name="livestream1"></stream>
</controlplane-request>
```

Sample response

```
<?xml version="1.0" encoding="UTF-8"?>
<controlplane-response xmlns="http://ns.adobe.com/hds/controlplane/status/1.0">
  <manifest-file>
    /livepkgr/_definst_/livestream.f4m
  </manifest-file>
  <stream name="livestream1">
    <status-code>
      1
    </status-code>
    <status-desc>
      Command executed successfully.
    </status-desc>
  </stream>
</controlplane-response>
```

Using a third-party web server

You can use web servers other than the included version of Apache to serve Adobe Media Server files. Note, however, that the HTTP Module for HTTPDynamic streaming is currently supported only on Apache web servers. To use a third-party web server, use the following guidelines:

- Configure Adobe Media Server and the web server so that they do not share a port. (You must decide which server can use ports 80 and 443, or you must multi-home the servers (by using separate IP addresses/physical NICs).
- If you want Adobe Media Server and your web server to share a port, set up Adobe Media Server as the proxy, put your web server on an unused port, and change the `ams.ini` `HTTPPROXY_HOST` setting to point to that port.
- If you want your web server to serve the Adobe Media Server standard files, either put the files in your web server's document root, or add the Adobe Media Server document root (`rootinstall/webroot`) as a virtual directory for your web server.
- If you want your web server to serve Progressive Download (PD) files from the same location as Adobe Media Server, add an alias or secondary document root to point to `rootinstall/webroot/vod`. You can also add a virtual directory to the vod application that points to your web server's document root.
- If you want your web server to use the exact same settings as the Adobe Media Server pre-installed web server (or you want to start with that and then customize it), start with the default Apache `httpd.conf` file that was included with Adobe Media Server.
- Configure your web server so that it handles the correct MIME types. For example, add the `video/x-flv` MIME type for `*.flv` files. For other MIME types, view the configuration file for the `mod_mime_magic` Apache module and the `installroot/Apache2.2/conf/mime.types` file.

Configuring Differentiated Services (DiffServ)

Flash Media Server 3.5.2

Understanding DiffServ

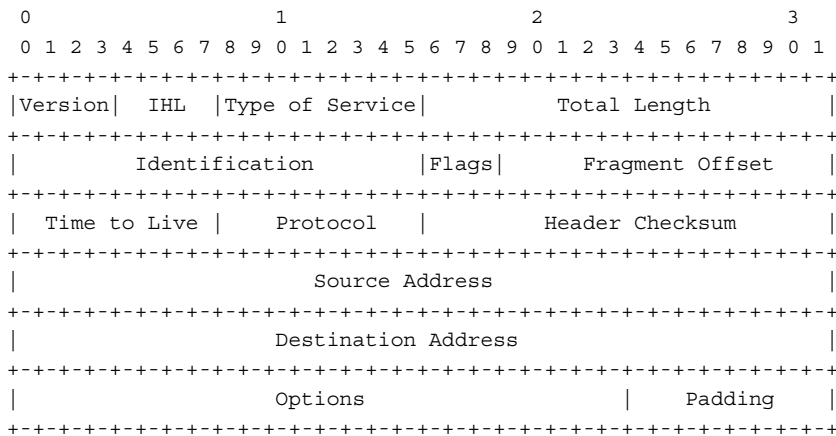
Use Differentiated Services to handle different types of traffic differently within a network. For example, you could give RTMP traffic a higher priority than FTP traffic. Or, you could give higher bandwidth and latency to RTMP than to HTTP. A publisher could choose to give paying customers' streams higher priority. A CDN or other hosting company could choose to give higher-paying publishers' streams higher priority. Historically, IP contained a feature called Type of Service (TOS) (see RFC 1349). Each packet's header contained a 3-bit Precedence value and three 1-bit flags. You could use these bits to prioritize packets in an operating system kernel or router. IPv6 had an experimental (and now deprecated) feature that provided for a 3-bit Priority, sent in the same header position as TOS precedence. This feature was replaced by a TCLASS (later renamed TRAFFIC_CLASS) byte that works the same as IP_TOS in IPv4. Differentiated Services (see RFC 2474) has replaced TOS with a single 6-bit value called DSCP (Differentiated Services Code Point). In IPv6, the TCLASS octet has been renamed the DS octet, and the other 2 bits are reserved. In IPv4, the TOS byte no longer has a name, and the other 2 bits are used for ECN (Explicit Congestion Notification). Edge routers and endpoints can make decisions on a per-flow basis. They mark the packets within the flow so that internal routers can look at the headers (as with TOS). DiffServ is defined such that edge routers can work entirely based on service-level agreements (SLAs) between two networks, but internal sources can also mark packets based on the type of service required for internal traffic.

DiffServ system requirements

Differentiated Services is supported on Linux only. There is no standard way of setting the bits on Windows platform that would work on all versions of the operating system that Adobe Media Server supports. For more information on setting Differentiated Services bits on Windows, see the following, <http://support.microsoft.com/kb/248611>. If you try to set the bits on Windows, the following message prints to the Authorization log, "Setting Diff Serv bits is not supported for Windows Platforms".

DiffServ Internet Protocol versions

The Adobe Media Server Differentiated Services feature supports the IPv4 protocol only. Setting the bits for IPv4 protocol means setting the Differentiated Services field (previously called the Type of Service field) of the IP packet header. The following is an IPv4 packet header:



For IPv6, Linux provides no standard way to set the Differentiated Services field (which has been renamed TCLASS field). If Adobe Media Server has IPv6 enabled and clients connect using an IPv6 address, setting bits on IPv6 clients does not result in an error. However bits are not set.

Note: Although bits are not set, the server reports in the logs that bits were set successfully.

Unsupported operations

The following operations are not supported:

- Setting bits between an edge server and an origin server.

The connections made between an edge server and an origin server are not “normal” connections. These connections are identified as “group” or “virtual”. To determine the connection type, call the Administration API `getNetStreamStats()` command and look at the `client_type` property. Trying to set DiffServ bits on these connections results in the following error message in the core log file, “Setting Diff Serv bits is not supported for group, service or virtual connections”.

- Setting the ECN bits of a Differentiated Services field.

For example, if you try to set DiffServ bits to 50 with mask 255, the bits set are 48 with the following warning in the AuthMessage log file, “ECN part of diffserv will not get set for diffserv bits 50 and mask 255”.

Configure DiffServ in the Server.xml file

Use the `DiffServ` and `DiffServMask` elements in the `Server.xml` configuration file to set DiffServ bits for all sockets that connect to the RTMP listener. Bits are set for all sockets, including tunneling and HTTP proxy sockets. All the outgoing packets on a socket are set, including the packets from the server that are exchanged during handshaking.

The following example sets `DiffServ` to 128 which sets the top 3 bits to 4. It sets the `DiffServMask` to 224 which turns on the top 3 bits.

- 1 Open the `rootinstall/conf/Server.xml` file in a text editor.
- 2 Edit the `Root/Server/ResourceLimits/Protocol/RTMP/DiffServ` element:

```
<DiffServ>128</DiffServ>
```
- 3 Edit the `Root/Server/ResourceLimits/Protocol/RTMP/DiffServMask` element:

```
<DiffServMask>224</DiffServMask>
```
- 4 Save and validate the file.
- 5 Restart the server.

Note: The server doesn't log whether the bits are set successfully or not. To verify that the bits are set, use the `getNetStreamStats()` Administration API.

Configuring DiffServ in an Authorization plug-in

Use the `F_CLIENT_DIFFSERV_BITS` field and the `F_CLIENT_DIFFSERV_MASK` field of the Authorization plug-in to set Differentiated Services fields. The sample Authorization plug-in shipped with Adobe Media Server has got the header file updated. The following example sets the bits in the Authorization plug-in:

```

U8 dscpVal;
if (getU8Field(m_pAev, IAmsAuthEvent::F_CLIENT_DIFFSERV_BITS, dscpVal))
{
    dscpVal = 0xFC;
    bool res = setU8Field(m_pAev,
        IAmsAuthEvent::F_CLIENT_DIFFSERV_BITS,
        dscpVal);
}
U8 dscpMask;
dscpMask = 0xF0;
bool res = setU8Field(m_pAev,
    IAmsAuthEvent::F_CLIENT_DIFFSERV_MASK,
    dscpMask);

```

The following table shows how to use the mask and bits fields together to get the desired bit settings on the socket:

No.	Current value on socket	DiffServ bits	DiffServ mask	New value on successful set	Comment
1	0	50	252	48	ECN bits were masked.
2	0	100	252	100	
3	0	255	255	252	Although ECN bits were not masked, setting them is not allowed.
4	100	200	128	228	Mask allowed only MSB to be modified.
5	100	200	252	200	Mask allowed DSCP bits to be modified.

Set operations

Set bits in the following Authorization events: E_CONNECT and E_PLAY. You can set bits in the `authorize()` method of the Authorization plug-in. You cannot set bits in the `notify()` method of the Authorization plug-in.

Get operations

Get bits in the following events: E_CONNECT, E_PLAY, and E_STOP. The get result on the `F_CLIENT_DIFFSERV_MASK` field always returns the value `0xFC`.

Logging

The following table lists error messages for normal connections:

Condition	Message	Log file
Unable to set DSCP bits from the Authorization plug-in.	While setting diff serv bits for uri %s: Unable to set DiffServ field.	Core
Unable to query DSCP bits from the socket.	While setting diff serv bits for uri %s: Unable to get DiffServ field.	Core

Condition	Message	Log file
Setting DSCP bits for group/service/virtual connections.	Setting Diff Serv bits is not supported for group, service or virtual connections.	Core
Attempting to set DSCP values on Windows.	Setting Diff Serv bits is not supported for Windows Platforms.	Auth
Setting DSCP bits for an unknown type of connection.	Setting Diff Serv bits (B: %d, M: %d) for this connection (from ip: %S) is not supported.	Core

The following table lists error messages for tunneled connections:

Condition	Message	Log file
Unable to set DSCP bits from the Authorization plug-in.	While setting diff serv bits for uri %s: Unable to set DiffServ field.	Edge
Unable to query DSCP bits from the socket.	While setting diff serv bits for uri %s: Unable to get DiffServ field.	Edge
A socket is being used for multiple sessions and a session is trying to set the DSCP bits on a socket that already has DSCP bits set by another session.	Destination address %ls is being shared among multiple net connections. Changing the DSCP settings from the one for uri %ls to the uri %ls.	Edge
Setting DSCP bits for group/service/virtual connections.	Setting Diff Serv bits is not supported for group, service or virtual connections.	Core
Attempting to set DSCP values on Windows.	Setting Diff Serv bits is not supported for Windows Platforms.	Auth
Setting DSCP bits for an unknown type of connection.	Setting Diff Serv bits (B: %d, M: %d) for this connection (from ip: %S) is not supported.	Core

The following table lists warning messages:

Condition	Message	Log file
If ECN part of mask is ON for the bits informed by the Authorization plug-in.	ECN part of diffserv will not get set for diffserv bits %d and mask %d.	AuthMessage

The following table lists info messages:

Condition	Message
DSCP bits are successfully set by the Authorization plug-in.	While setting diff serv bits for uri %s: Successfully set the diff serv bits (bits: %d, mask: %d) for addr:port %S:%d and nc uri %s. Current diff serv value is %d.
DSCP bits informed from the Authorization plug-in are already set for RTMP.	Requested diff serv bits (bits: %d, mask: %d) for addr:port %S:%d and nc uri %s already set. Current diff serv value is %d.

Verifying bits

To verify that bits are set on a socket, call the `getNetStreamStats()` Administration API in a browser:

```
http://<server>:<admin_port>/admin/getNetStreamStats?user=<admin_user>&passwd=<admin_password>&appInst=<app_instance>&streamIds=-1
```

The command results in output similar to following:

```
<result>
  <level>status</level>
  <code>NetConnection.Call.Success</code>
  <timestamp>Mon 25 Jan 2010 08:00:00 PM PST</timestamp>
  <data>
    <_0>
      <name>sample</name>
      <time>Sun 24 Jan 2010 06:00:00 PM PST</time>
      <type>playing recorded</type>
      <client>BCAgYLPK</client>
      <stream_id>BBAgYLPK</stream_id>
      <client_type>normal</client_type>
      <diffserv_bits>0xfc</diffserv_bits>
    </_0>
  </data>
</result>
```

The `diffserv_bits` element updates only when bits are set successfully.

Verifying bits in an RTMP stream

Suppose a client connects to the server and plays a stream. Suppose that the Authorization plug-in sets the bits twice, first when the `E_CONNECT` event is received, and then when the `E_PLAY` event is received. When the stream is playing, use Wireshark or a similar tool to capture the packets. When streaming stops, stop the capture and filter the packets so that the source port is 1935 or whichever port is used for RTMP. After you capture at least one packet, right click and select “Follow TCP stream” to filter all packets for that particular stream. Sort the packet based on time. If you observe the packets from the beginning of the stream, the DiffServ field is 0 until the `NetConnection` response is sent from the server. From the `NetConnection` response packet until the `NetStream.Play.Reset` or `NetStream.Play.Start` packet, whichever is sent first, the bits are set to what was set in the `E_CONNECT` event. From this point, the bits are set to what was set in the `E_PLAY` event.

Note: When you use the `Server.xml` file to set bits, even the packets sent from the server as part of handshaking have the bits set.

Verifying bits in an RTMPT stream

RTMPT connections behave a bit differently than RTMP connections. When creating an RTMPT connection, multiple sockets can open from the client-side. If you open multiple `NetConnections` from the same client, these sockets are probably shared among the `NetConnections`. In such cases, the DSCP setting for a socket is the DSCP value from the session that most recently had its DSCP bits set. The following is an example: Time `t1`: `NetConnection nc1` connects and play starts. Its DSCP bits are set to `0x64`. The `NetConnection` uses sockets `s1`, `s2`, and `s3`. All the sockets have the DSCP bits set to `0x64`. `t2`: `NetConnection nc2` connects. The Authorization plug-in doesn't set any DSCP bits. The `NetConnection` uses sockets `s2` and `s4`. Socket `s2` has the DSCP value `0x64`. Socket `s4` has the default value (if specified in the `Server.xml` file). `t3`: `NetConnection nc2` starts playing. The Authorization plug-in sets the DSCP bits to `0x88`. From now on, sockets `s2` and `s4` have the DSCP values `0x88`, whether they are carrying packets for `nc1` or `nc2`. `t4`: `NetConnection nc2` play stops. Nothing changes.

Chapter 3: Using the Administration Console

Use the Adobe Media Server Administration Console to maintain the server, monitor applications, and manage users. Application developers can also use the Administration Console to debug applications.

Connecting to the Administration Console

About the Administration Console

The Administration Console is an Adobe Flash Player application (`ams_adminConsole.swf`) that lets you manage the server and view information about applications running on the server.

The Administration Console connects to Adobe Media Administration Server, which connects to Adobe Media Server. To log in to the Administration Console, the Administration Server must be running.

By default, Adobe Media Administration Server is installed on port 1111. To change the port number of the Administration Server after installation, edit the `ams.ini` file. Clients cannot access the Administration Server securely (over RTMPS). As a result, Adobe recommends blocking port 1111 from external requests so that only clients within your firewall can access the Administration Server.

***Note:** The Administration Console calls Administration APIs to inspect and manage the server. Use the Adobe Media Server Administration API Reference to build your own administrative applications.*

Connect to the Administration Console

There are two types of administrators: server administrators and virtual host (vhost) administrators. When you log in to the Administration Console as a virtual host administrator, your session is specific to a virtual host, and you can only manage applications running on that virtual host. Server administrators have access to all applications running on the server.

1 To open the Administration Console, do one of the following:

- On Windows, select Start > Programs > Adobe > Adobe Media Server 5.0 > Adobe Media Server Administration Console.
- On Linux, open the `ams_adminConsole.htm` and `ams_adminConsole.swf` files in a browser with Flash Player.
- On Mac® OS, copy the `ams_adminConsole.htm` and `ams_adminConsole.swf` files to the Mac. Open the `ams_adminConsole.htm` file in a web browser that has Flash Player installed.

If you installed Apache with Adobe Media Server, the `ams_adminConsole.swf` and `ams_adminConsole.htm` files are located in the `rootinstall/webroot` folder. Otherwise, the `ams_adminConsole.swf` and `ams_adminConsole.htm` files are located in the root installation folder.

2 (Optional) Specify a server name. This name is an alias you can use to connect to a server quickly. The Administration Console remembers the server address for this server name the next time the console is opened.

3 In the Server Address box, do one of the following:

- Type **localhost** if the server and the Administration Console are running on the same computer. If the Administration Server is installed on a port other than 1111 (the default), you must enter the port number as well; for example, **localhost:1234**. Entering localhost connects you to the default virtual host on this computer.
- To connect to a virtual host other than the default virtual host, enter the fully qualified host name (for example, www.example.com). The host name must be mapped to a valid network IP address. To test this behavior on Windows, you can edit your local hosts file at windows\system32\drivers\etc\hosts.
- If you are connecting remotely by running the Administration Console on another computer, enter the server's name (AdobeMediaServer.myCompany.com) or the IP address and port number (12.35.06.78:1112) of the Administration Server to which you want to connect. Ensure your computer has permission to connect to the specified port on the other computer. Also, check that the Administration Server has not been configured to prohibit connections from the specific domain or IP address you are using.

4 Enter your username and password.

If you are the server administrator who installed the server, enter the administrator user name and password you entered during installation. Otherwise, enter the username and password you received from the server administrator.

Note: To reset a password, see “Delete administrator accounts and reset passwords” on page 92

For security reasons, the Administration Console does not save your password between sessions.

To log in to a virtual host that is not on the default adaptor, you must specify the name of the adaptor. For example, when logging in to a virtual host on the adaptor `_secondAdaptor_`, the vhost administrator `JLee` would enter the following information in the Username box: `_secondAdaptor_/JLee`.

5 (Optional) Select the Remember My Password option.

6 (Optional) Select the Automatically Connect Me option.

7 (Optional) Click Revert to return the Administration Console to its default settings.

Reverting deletes all saved servers, user names, and passwords from the Administration Console. All custom resizing within the Administration Console is restored to the original state. (The Revert button, however, does not affect the server.)

8 Click Login.

You can disconnect at any time by clicking Logoff.

Note: The color of the vertical bar in the upper-right corner (next to the question mark icon) indicates whether the Administration Console is connected (green) or not connected (red) to a server.

Near the top of every screen of the Administration Console are two icons. Click the folder icon to display links to the Adobe Media Server website and related resources. Click the question mark icon to display links to Adobe Media Server Help.

To run the Administration Console from a computer other than the one in which the server is installed, copy `ams_adminConsole.htm` and `ams_adminConsole.swf` to the other computer, or make sure that this file is in the webroot directory so it can be accessed remotely. In both cases, verify that the `Allow` and `Deny` tags in the `Users.xml` file allow the connection from the other computer's IP address.

Change or pause the refresh rate

The information in the Administration Console panels refreshes every 5 seconds by default. You can change the refresh rate to any time interval between 1 and 60 seconds, or pause refreshing at any time.

Change the refresh rate of the Administration Console

- ❖ Click the pop-up menu next to Refresh Rate (upper-right corner) and select a new time duration, such as 10 seconds.

Pause refreshing the Administration Console

- 1 Click the pop-up menu next to Refresh Rate (upper-right corner), scroll down, and select Pause.
- 2 Click Pause Refresh to continue.

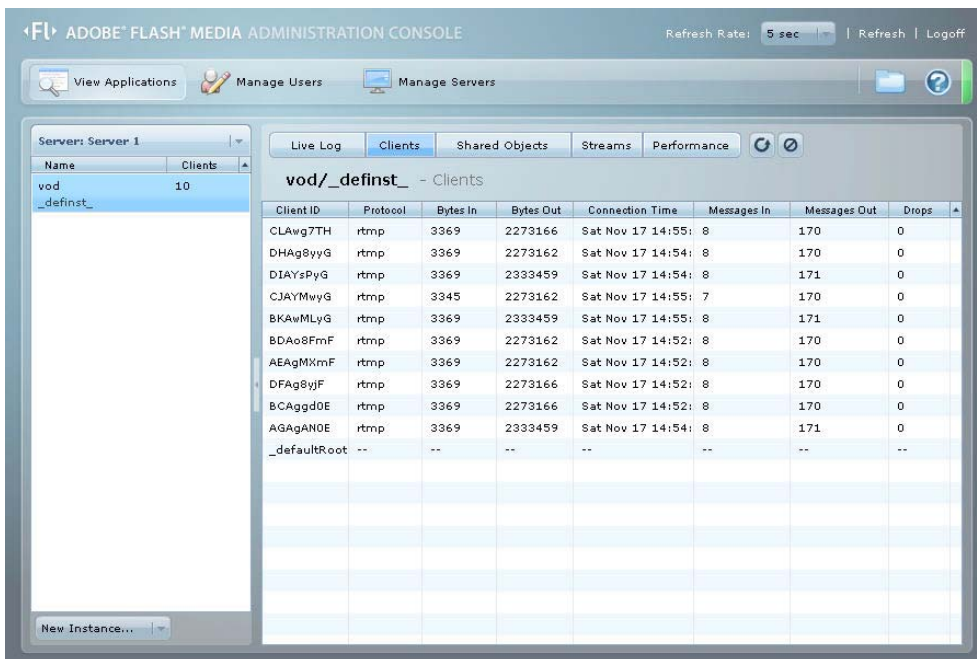
A red border appears around the panels of the Administration Console to show that the refresh feature is paused.

- 3 To start refreshing information again, click the pop-up menu and select a time duration.

Inspecting applications

View applications

After connecting to a server or virtual host, the Administration Console displays a panel that lists the currently running application instances. From here, the state of an application can be monitored.



Use the View Applications panel to view, load, or unload application instances.

Note: If you add an application while the Administration service is running and the new application doesn't appear in the Administration Console, move to another panel and then back to refresh the console.

Manually load an application instance in the Administration Console

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Enter the name and address of the server or virtual host to which you want to connect.
- 3 Enter the administrator user name and password.

- 4 Click View Applications.
- 5 Click New Instance.
- 6 Select the application from the pop-up menu. (The application must already be configured on the server.)
- 7 The Administration Console adds a default instance `_definst_`, which can be edited. Press Enter to submit the name and start the application instance. To cancel, press Shift+Escape.

Reload an application instance in the Administration Console

Reload an application instance to reload the server-side scripts for the instance or to disconnect all of its users while immediately allowing new connections.

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Select an application from the list.
- 4 Click Reload (circular arrow icon to the right of the Performance tab).

View information about an application

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Select the application from the list. The following information is listed for the application on the different tabs:
 - Log messages generated by the application instance on the server
 - A list of clients connected to the application instance
 - A list of active shared objects for the application instance
 - A list of active streams for the application instance
 - Information about the overall state of the selected application instance, such as total uptime or number of users

Sort applications list

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 In the applications list, do one of the following:
 - Click the Name column header to sort the applications list by name.
 - Click the Clients column header to sort by client.

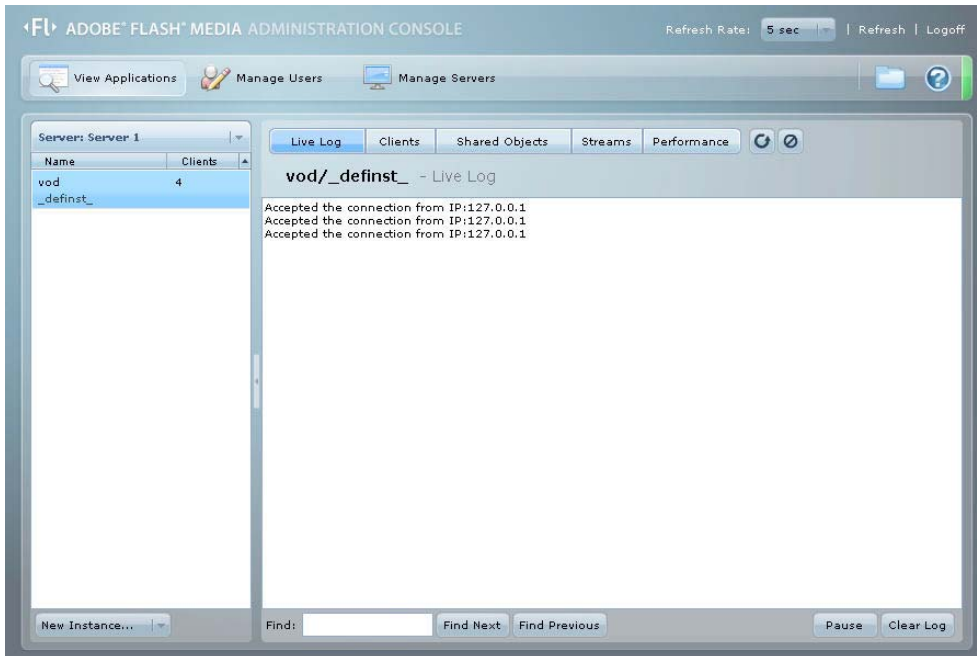
End an application instance

When an application instance is ended, all users are disconnected and all instance resources are released.

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Select an application from the list.
- 4 Click Unload (stop icon to the right of the Performance tab)

Viewing log messages for an application

The Administration Console Live Log panel displays log messages and `trace()` statements from server-side scripts for the selected application instance. The information in this panel is updated whenever the application instance generates a log message. (If the console refresh feature is paused, log messages are still received.)

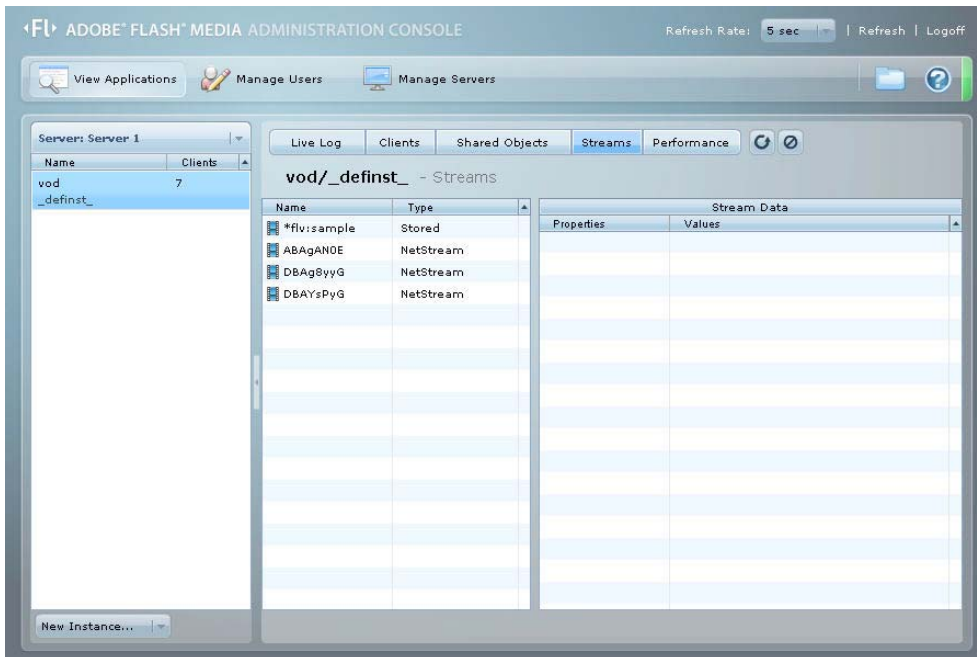


Live Log panel

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Click Live Log.
- 4 Select an application from the list.
- 5 Type text in the Find text box and click Find Next. Use the Find Previous and Clear Log buttons as necessary.

Viewing active streams

Use the Administration Console Streams panel to view information about streams and to play streams.



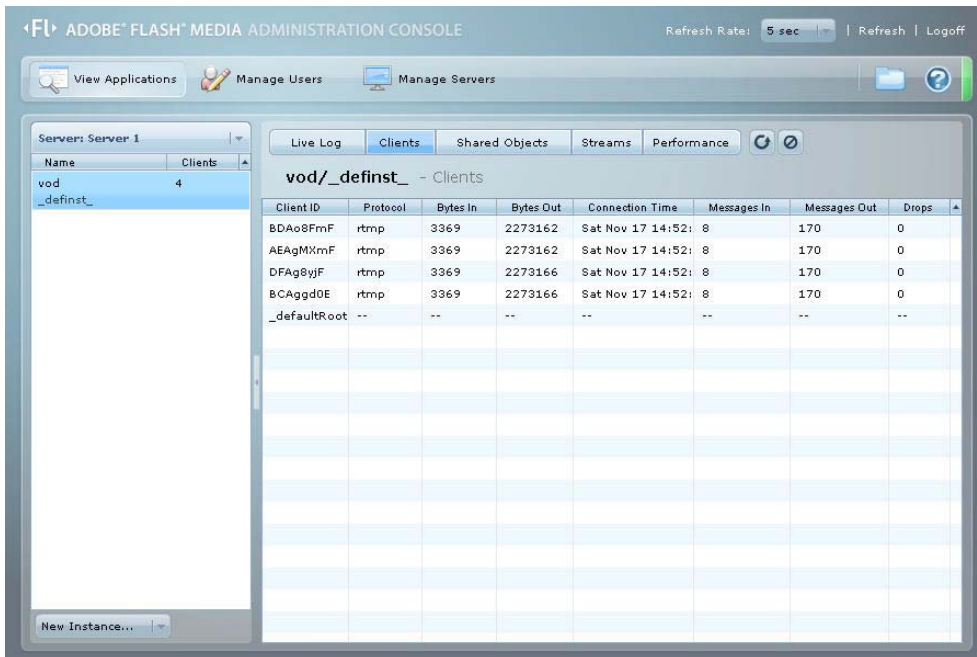
Streams panel

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Click Streams.
- 4 Select an application from the list. The Streams panel displays the following information:
 - Name: For NetStream streams, the name is the NetStream ID (a server-generated number). For a live stream being published, the entry displays the live stream name. For a recorded steam, the entry displays the FLV or MP3 filename; for example, flv:stream2.flv or mp3:sound.mp3. If a client requests the stream2.flv, there will be two entries: one for flv:stream2.flv (stored) and one for the actual network stream going to the client.
 - Type: A string that describes the type of stream, either stored, live, or NetStream.
- 5 Select a stream to view its properties. The values of the properties are as follows:
 - Name: The actual stream name, not streamID.
 - Status: States if the stream is publishing, playing live, or playing recorded.
 - Client: The client ID playing the stream.
 - Time: The time that the client started playing the stream.

Note: If the stream type is available for debugging, the Administration Console displays its properties in the adjoining panel. If the type is not available for debugging, an error message is displayed.
- 6 Click Play Stream to start playing the selected stream in a separate window that is the size of the selected stream. (The Play Stream button appears only if a debug connection is possible. Only named streams can be played.)

Viewing active clients

The Administration Console Clients panel lists detailed information about all clients connected to an application.

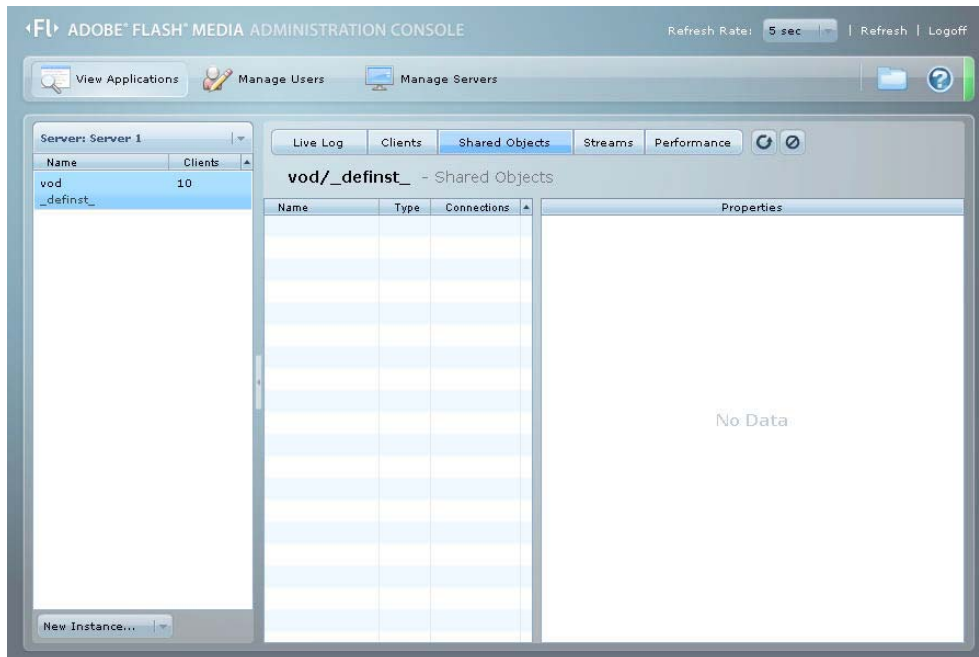


Clients panel

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Click Clients.
- 4 Select an application from the list. The following information is displayed:
 - Client ID: The internal ID of the client; this represents a server-generated number that Adobe Media Server uses to identify each client.
 - Protocol: The connection protocol that the client uses, such as RTMP.
 - Bytes In and Bytes Out: The average bytes per second being sent to and from the server. The Administration Console calculates this ratio by dividing the total number of bytes received in the most recent 15 seconds by 15. When the panel first appears, these figures appear as pending because there is only one data point to start with; figures appear after the panel is open for 15 seconds.
 - Connection Time: The date and time the client connected.
 - Messages In and Messages Out: The number of messages sent to or from the client. Messages In reflects update requests sent from clients to the server; Messages Out reflects notification of successful updates sent from the server to connected clients.
 - Drops: The number of messages dropped since the client connected. For live streams, audio, and video, messages may be dropped; for recorded streams, only video messages are dropped. Command messages are never dropped.

Viewing active shared objects

The Administration Console Shared Objects panel lists the active shared objects for an application and can be useful when debugging an application. The information is automatically refreshed every 5 seconds, or click Refresh to refresh at any time. The Administration Console displays the name, type (persistent or temporary), and connections (number of users subscribed) of each shared object.



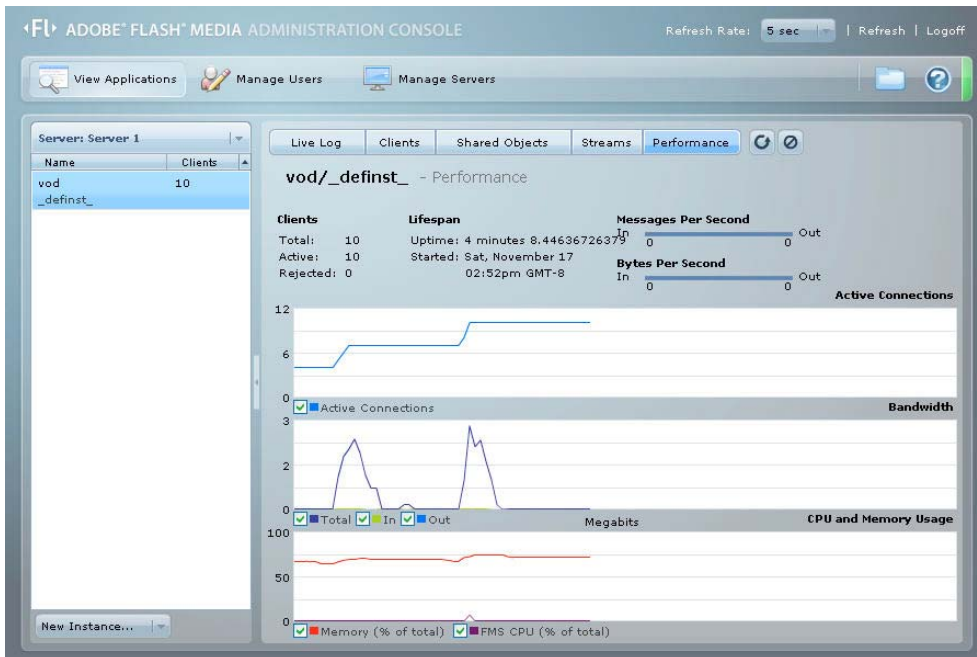
Shared Objects panel

Note: The Administration Console was built using ActionScript 2.0 and cannot understand AMF3 data. Therefore, the Administration Console cannot display data in remote shared objects in applications built in ActionScript 3.0 unless the application sets the `NetConnection.objectEncoding` property to `AMF0`.

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Click Shared Objects.
- 4 Select an application from the list.
- 5 To display information about a shared object, click the object. The number of users currently connected to and using the shared object is displayed, along with the data properties assigned to the shared object.

View performance information

The Administration Console Performance panel shows information about the overall state of the application instance. The information is automatically refreshed every 5 seconds, or click Refresh to refresh at any time.



Performance panel

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click View Applications.
- 3 Click Performance.
- 4 Select an application from the list. The following information is displayed:
 - Clients: Information about clients connected to this application instance, including the total number of clients who connected to the application instance since it started, active clients, and the number of users whose attempts to connect to the application instance were rejected. (To determine why connections may have failed, look at the Live Log panel under View Applications.)
 - Lifespan: The length of time the application instance has been running and the date and time it began to run.
 - Messages Per Second: The average number of messages (video frames, audio packets, and command messages) sent per second.
 - Bytes Per Second: The average number of bytes sent per second for this application instance. The Administration Console calculates this ratio by determining the total number of bytes received in the most recent 15 seconds and dividing that value by 15. When the panel first appears, these figures appear as pending because there is only one data point to start with; figures appear after the panel is open for 15 seconds.
 - Active Connections: The number of users currently connected to the application instance.
 - Bandwidth: The amount of data that the application instance manages, including data sent, data received, and the combined amount of data traffic.
 - CPU and Memory Usage: The percentage of CPU and memory used by Adobe Media Server.
- 5 Select and deselect checkboxes to customize the information displayed on the graphs. For example, in the Bandwidth graph, select Total and deselect In and Out to show only the total amount of bandwidth used.

Manage administrators

About administrator roles

Administrators are users who are allowed to log in to the Administration Console. There are two types of administrators: server administrators and virtual host administrators.

Server administrators can control all virtual hosts and perform server-level tasks, such as restarting or shutting down the server. Server administrators can access and perform all operations on all tabs.

Virtual host administrators can manage the applications on their virtual host—for example, they can reload or disconnect applications. Virtual host administrators can access and perform operations on the View Applications tabs. They cannot manage servers or administrative users.

Add administrators

Use the Administration Console to add server administrators

Note: You must be a server administrator (not a virtual host administrator) to access the Manage Users tab.

- 1 Log in to the Administration Console as a server administrator.
- 2 Select the Manage Users tab.
- 3 Click New User.
- 4 Enter the user name and password and click Save.

The Administration Console adds the new user and password to the Users.xml file. You do not need to restart either server.

Edit the Users.xml file to add server administrators

- 1 Open the `rootinstall/conf/Users.xml` file.
- 2 Locate the `UserList` section.
- 3 Add a new `<User></User>` section for each server administrator you want to add.

The `User` name attribute specifies the user name. The `Password` element specifies the password. The `Allow`, `Deny`, and `Order` elements specify the hosts from which the administrator can connect to the Administration Console. The following sample XML adds a user who can connect from any domain:

```
<UserList>
  <User name="{SERVER.ADMIN_USERNAME}">
    <Password encrypt="false">{SERVER.ADMIN_PASSWORD}</Password>
    <Allow></Allow>
    <Deny></Deny>
    <Order>Allow,Deny</Order>
  </User>
  <User name="janedoe">
    <Password encrypt="false">S4mpl3P4ss</Password>
    <Allow></Allow>
    <Deny></Deny>
    <Order>Allow,Deny</Order>
  </User>
</UserList>
```

For more information, see the comments in the Users.xml file.

- 4 Validate the XML and save the Users.xml file.

Add virtual host administrators

- 1 Open the Users.xml file in the root folder of the virtual host; for example, *rootinstall/conf/_defaultRoot_/www.sampleVhost.com/Users.xml*. If the file doesn't exist, copy the Users.xml file from the *rootinstall/conf* folder.
- 2 Locate the `UserList` section.
- 3 Add a new `<User></User>` section for each virtual host administrator you want to add.

The `User` name attribute specifies the user name. The `Password` element specifies the password. The `Allow`, `Deny`, and `Order` elements specify the hosts from which the administrator can connect to the Administration Console. The following sample XML adds a user who can connect from any domain:

```
<UserList>
  <User name="{SERVER.ADMIN_USERNAME}">
    <Password encrypt="false">{SERVER.ADMIN_PASSWORD}</Password>
    <Allow></Allow>
    <Deny></Deny>
    <Order>Allow,Deny</Order>
  </User>
  <User name="vHostAdmin">
    <Password encrypt="false">Ex4mpl3P4ss</Password>
    <Allow></Allow>
    <Deny></Deny>
    <Order>Allow,Deny</Order>
  </User>
</UserList>
```

For more information, see the comments in the Users.xml file.

- 4 Validate the XML and save the Users.xml file.

Delete administrator accounts and reset passwords

Only server administrators (not virtual host administrators) can perform operations on the Manage Users tab.

Passwords are stored as salted hashes in the *rootinstall/conf/Users.xml* configuration file.

Delete user account

- 1 Log in to the Administration Console as a server administrator.
- 2 Click Manage Users.
- 3 Select a user.
- 4 Click Delete This User Account On The Server.
- 5 Confirm the action.

Reset user password

- 1 Log in to the Administration Console as a server administrator.
- 2 Click Manage Users.
- 3 Select a user.
- 4 Click Reset The Password For This User.

- 5 Enter a new password.

Use the command line to reset a user password

- 1 Open a shell window.
- 2 Go to the Adobe Media Server installation directory:
(Windows) C:\Program Files\Adobe\Adobe Media Server 5.0
(Linux) /opt/adobe/fms
- 3 Stop Adobe Media Server and Adobe Media Administration Server. See [Start and stop the server](#).
- 4 Enter one of the following:
 - (Windows) **amsadmin -console -user <username>**
 - (Linux) **./amsadmin -console -user <username>**Enter an eight character password.

Note: The password is not exposed because terminal echo is off.

Use the Administration API to reset a user password

- ❖ Call the [changePswd\(\)](#) API.

For more information about using the Administration API, see “[Working with the Administration API](#)” on page 118.

Managing the server

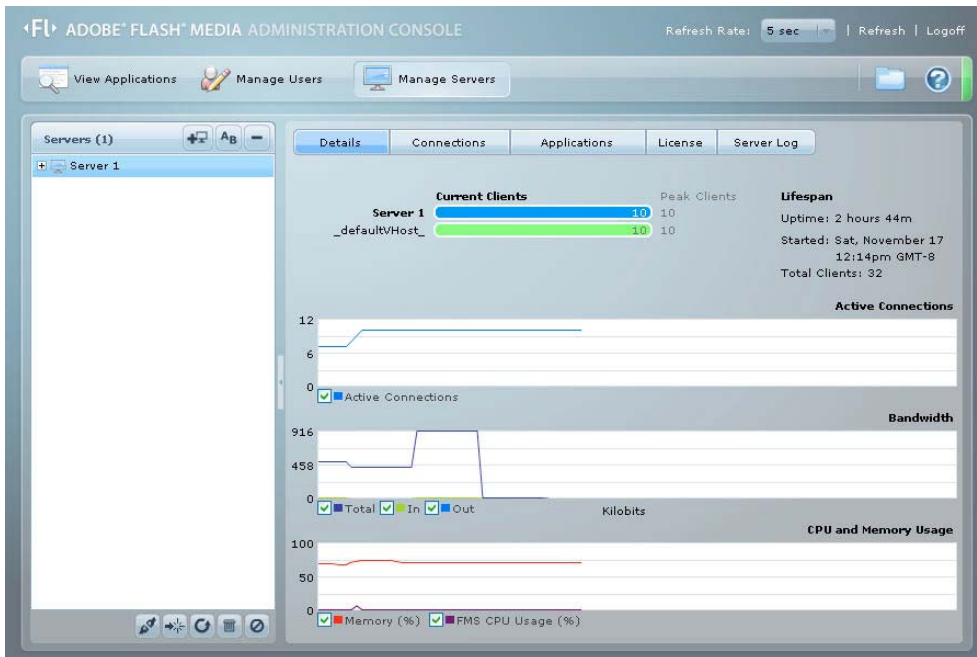
Monitoring server performance

You can review the performance of individual servers or a group of servers using the Administration Console. The servers are arranged in a tree structure.

A series of tabs is displayed along the top of the Manage Servers panel. From here, you can perform the following actions:

- Review the performance statistics for the computer where the applications are running.
- Review information about connections to the server.
- Review information about the applications located on the server or virtual hosts.
- Review server licenses and, if necessary, add serial keys.
- Review the access log and server log.

The Servers panel occupies the left side of the screen in this section of the Administration Console. The panel lists the servers and virtual hosts that you can access and manage.



Manage Servers panel

Use the small buttons at the top and bottom of the panel to perform the following tasks:

- Add a new server to the list.
- Edit server login information (user name and password) and select options such as remembering the password and automatically connecting when logging in to the server.
- Delete a server from the list.
- Connect to the selected server.
- Ping the server to verify that it is running.
- Restart the server or a virtual host.
- Run garbage collection to clear unused server resources, such as streams and application instances, from memory. (Automatic garbage collection intervals can be set in the Server.xml and VHost.xml configuration files.)
- Stop a server or virtual host.

Viewing server details

The Administration Console Details panel displays live information for the server.

- 1 Follow the steps in “[Connect to the Administration Console](#)” on page 82.
- 2 Click Manage Servers.
- 3 Click Details.
- 4 Select a server from the list. The following information is displayed:
 - Total number of current clients
 - Life span of the server

- Graphical displays of active connections, bandwidth resources consumed, and CPU and memory resources consumed

Viewing connection details

The Administration Console Connections panel lists all client connections to the selected server.

- 1 Follow the steps in [“Connect to the Administration Console”](#) on page 82.
- 2 Click Manage Servers.
- 3 Click Connections.
- 4 Select a server from the list. The following information is displayed for each client accessing the server or virtual host:
 - Server name
 - If connection has been made
 - Number of connections
 - Number of disconnections
 - Number of bytes in and out
 - Number of messages dropped

Viewing application details

The Administration Console Applications panel displays detailed information for all the applications running on the selected server or virtual host.

- 1 Follow the steps in [“Connect to the Administration Console”](#) on page 82.
- 2 Click Manage Servers.
- 3 Click Applications.
- 4 Select a server from the list. The name of each application, along with the following information, is displayed:
 - Server name
 - Application name
 - Number of instances of the application that have been loaded on and unloaded from the server
 - Number of users that are connected
 - Number of users that have connected and disconnected
 - Number of instances currently active
 - Number of instances that have been unloaded from the server
 - Total number of connections that have been accepted or rejected for each application

Viewing license files

The Administration Console License panel is where you add serial keys. The panel also displays detailed information for all serial keys authorizing you to run Adobe Media Server on the selected server. The lower frame displays information about custom licenses.

- 1 Follow the steps in [“Connect to the Administration Console”](#) on page 82.
- 2 Click Manage Servers.

- 3 Click License.
- 4 Select a server from the list. For each license, the following information is displayed:
 - The individual serial key number
 - Authorized peak number of client connections
 - Bandwidth cap
 - Whether the license is valid (true) or not (false)

Note: Your organization may have more than one license, so note the capacity totals listed near the bottom of the Administration Console.

Add a serial key

Add serial keys in the Administration Console License panel.

- 1 Follow the steps in [“Connect to the Administration Console”](#) on page 82.
- 2 Click Manage Servers.
- 3 Click License.
- 4 Enter the serial key number in the text boxes at the bottom of the Administration Console.
- 5 Click Add Serial Key.
- 6 Restart the server.

Note: Serial numbers that are added manually (that is, added by editing configuration files directly) to either `ams.ini` or the `<LicenseInfo>` tag of the `Server.xml` file cannot be removed using the Administration Console. Only serial numbers that are added using the Administration Console can be deleted using the Administration Console.

View the access log file

The Administration Console Server Log panel displays messages that are written to the access log.

- 1 Follow the steps in [“Connect to the Administration Console”](#) on page 82.
- 2 Click Manage Servers.
- 3 Click Server Log.
- 4 Select a server from the list.
- 5 To locate a specific string, type it in the Find text box and click Find Next. Use the Find Previous and Clear Log buttons as necessary.

Chapter 4: Monitoring and Managing Log Files

Adobe Media Server has a variety of log files to help you manage and troubleshoot the server. The log files track server activity, such as who is accessing the server, how users are working with applications, and general diagnostics. Logs are in W3C format. Administrators can use standard parsing tools to parse the log files.

Working with log files

About log files

The server maintains several different types of logs. The server outputs statistics about client connections and stream activity to access logs. The server also maintains diagnostic logs and application logs for application activities.

access.XX.log Tracks information about users accessing the server.

application.XX.log Tracks information about activities in application instances.

diagnostic logs Track information about server operations.

Note: In Adobe Flash Player 9 Update 3, Flash Player no longer notifies the server about pause events.

Configuration files for logging

Use the `Server.xml` and `Logger.xml` configuration files to configure logging. The `Server.xml` file contains a `Logging` section that controls logging behavior for the entire server. The `Logging` section includes an `Enable` tag that determines whether the server logs events. The `Logging` section also contains a `Scope` tag that determines whether the server writes separate log files for each virtual host or one file for the entire server. The location of each log file is determined by the `Directory` and `FileName` tags in the `Logger.xml` files.

If the `Scope` tag is set to `server`, the `Logger.xml` file in the `rootinstall/conf` folder determines the logging configuration for the whole server.

If the `Scope` tag is set to `vhost`, you can place `Logger.xml` files in virtual host root folders to control the behavior of each virtual host. If the `Scope` tag is set to `vhost` and a virtual host `Logger.xml` file doesn't exist, the root `Logger.xml` file controls the logging behavior. If the `Scope` tag is set to `server`, virtual host `Logger.xml` files are ignored.

For more information, see comments in the `Server.xml` and `Logger.xml` files installed in the `rootinstall/conf` directory.

Working with web server log files

Adobe Media Server installs with the Apache HTTP Server.

The default location of the Apache/Adobe Mediae log files is `RootInstall/Adobe2.2/logs`. The logs are in the default Apache error and combined access log formats. To change the location of the log files, edit the `RootInstall/Adobe2.2/conf/httpd.conf` file.

The Apache logs are named `access_log` and `error_log`. Adobe Media Server handles log rotation for the Apache logs.

For more information about Apache log files, see the Apache documentation at www.apache.org.

Rotating and backing up log files

Log files grow larger over time, but there are methods for managing log file size.

One option is to rotate log files, moving or deleting the oldest files. Use the `rotation` element in the `Logger.xml` file to specify a rotation schedule for log files. Two types of rotation schedules can be established. The first option is to set a daily rotation at a certain time. For example, setting daily at 00:00 rotates files every 24 hours at midnight.

Alternatively, set a rotation that occurs when the log exceeds a specified length. Name, maximum file size in kilobytes, and maximum number of log files to keep can also be customized using the `rotation` element. For an example, see the `Logger.xml` file installed in the `/conf` directory.

***Note:** Log file rotation cannot be disabled. However, you can set values in the `Logger.xml` configuration files that effectively turn off rotation. Choose a large value for the `MaxSize` tag. Set the `Schedule` type to "duration" and choose a long maximum duration. For more information about the `Logger.xml` configuration files, see ["Configuration files for logging"](#) on page 97.*

You can write an operating system script to delete or back up the log regularly. For important log files, move the log directory to a backup location. You can move the current active file; the server creates a new file on the next log event.

Access logs

Reading access logs

The access log records information about requests by Flash Player and Adobe Media Server application instances. Using these logs, you can find out about various events, such as when a user connected to the server, how much total bandwidth was consumed during the session, and which streams were accessed by the connection (and similar resource information). You can use the status codes associated with specific events to troubleshoot event failures. You can also use this information to determine which applications are used most.

The default access log is `access.xx.log`, which is located in the `RootInstall/logs` directory. The default configuration creates a single access log per server. You can also configure the server to create a separate file per virtual host. When logging is configured on a per-virtual-host basis, all logs for a particular virtual host are found in a subdirectory within the `logs` directory. The name of the subdirectory matches the virtual host name. Substitution strings can be found in the `[]` brackets, with `YYYY`, `MM`, `DD`, and `NN` representing year, month, date, and version, respectively. You can use the substitution string to customize the filename of the access log. (For example, `access.[YYYYMMDDNN].log` could be named `access.2007052401.log`.) To configure the server to create separate log files for each virtual host, set the value of the `scope` tag in the `Server.xml` file to "vhost." (This is a separate scope tag just for logging.)

Adobe Media Server defines event categories, and for each category, it defines events that can be recorded. Logging can be customized to record all events or only specific events by editing the `<Events>` and `<Fields>` elements in the `Logger.xml` file.

Logging events periodically for live and 24/7 applications

Many companies use statistics from the access log to bill customers. If your programming includes live events or 24/7 programming, the events you need to calculate billing might not occur within a billing cycle. To solve this problem, you can enable *checkpoint* events. Checkpoint events log bytes periodically from the start to the end of an event. You can configure how often the server logs checkpoint events. The following are available as checkpoint events: `connect-continue`, `play-continue`, and `publish-continue`. For information about configuring checkpoint events, see ["Enable checkpoint logging events"](#) on page 45.

Access events defined in access logs

Event	Category	Description
connect-pending	session	Client connects to the server, waiting for the client to be authenticated.
connect	session	Client connects to the server.
disconnect	session	Client disconnects.
connect-continue	session	A checkpoint event that provides updates of a corresponding connect event at intervals. Use the c-client-id field to find the corresponding connect event.
publish	stream	Client publishes a live stream.
unpublish	stream	Client unpublishes a live stream.
publish-continue	stream	A checkpoint event that provides updates of a corresponding publish event at intervals. Use the x-sid field (stream id) and the c-client-id field to find the corresponding publish event.
play	stream	Client plays a stream.
play-continue	stream	A checkpoint event that provides updates of a corresponding play event at intervals. Use the x-sid field (stream id) with the c-client-id field to find the corresponding play event.
pause	stream	Client pauses stream.
unpause	stream	Client resumes playing stream.
client-pause	stream	Client pauses a stream but the server still sends data to the client so the player has enough data to play when the client unpauses. This type of pause is called a "smart pause". Smart pause requires Flash Player 9,0,115,0 and Adobe Media Server 3 and later. The stream position is logged in the "c-spos" field.
client-unpause	stream	Client smart unpauses a stream. The stream position is logged in the "c-spos" field.
seek	stream	Client seeks in a stream and <code>NetStream.inBufferSeek=false</code> . Also fires when <code>NetStream.inBufferSeek=true</code> and the client seeks outside the buffer.
client-seek	stream	The seek position when the client seeks within the buffer when <code>NetStream.inBufferSeek=true</code> (called a "smart seek"). If a client seeks outside the buffer, the client sends a "seek" event. The stream position is logged in the "c-spos" field.
stop	stream	Client stops playing or publishing a stream.
record	stream	Client begins recording a stream.
recordstop	stream	Client stops recording a stream.
server-start	server	The server has started.
server-stop	server	The server has stopped.
vhost-start	server	A virtual host has started.
vhost-stop	server	A virtual host has stopped.
registry.start	registry	The registry has started.
registry.stop	registry	The registry has stopped.
service.start	registry	A service has started.
service.stop	registry	A service has stopped.
app-start	application	An application instance has started.

Event	Category	Description
app-stop	application	An application instance has stopped.
service.request	registry	A service request has been made.
debug-pending	session	A debug request is pending.
debug-approved	session	A debug request has been approved.
filenametransform	authorization	A virtual stream path has been mapped to a physical location. This event occurs if an Authorization plug-in is present to handle the event.
loadsegment	stream	A stream segment has loaded.
action	application	An action has occurred.
codec	stream	A codec has changed.
Authconnect	authorization	Client connects to server. This event occurs if an Authorization plug-in is present to handle the event.
Authplay	authorization	Client plays a stream. This event occurs if an Authorization plug-in is present to handle the event.
Authpublish	authorization	Client publishes a live stream. This event occurs if an Authorization plug-in is present to handle the event.
Authseek	authorization	Client jumps to a new location within a recorded stream. This event occurs if an Authorization plug-in is present to handle the event.
AuthRecord	authorization	Client begins recording a stream. This event occurs if an Authorization plug-in is present to handle the event.
start-transmit	stream	The server received a "startTransmit" command. This command asks the server to transmit more data because the buffer is running low.
stop-transmit	stream	The server received a "stopTransmit" command. This command asks the server to suspend transmission until the client sends a "startTransmit" event because there is enough data in the buffer.

Fields in access logs

Note: When the data for this field contains a space or delimiter, the data is wrapped in double quotation marks. The double quotation marks surrounding the data are not part of the data, but are present for better parsing of the data. This applies to the `tz`, `x-ctx`, `x-adaptor`, `x-vhost`, `s-uri`, `c-referrer`, `c-user-agent`, `cs-bytes`, `sc-bytes`, and `x-sname` fields.

The following formats apply to the fields in the table below:

For date: YYYY-MM-DD

For time: hh:mm:ss

For time zone: string such as "UTC," "Pacific Daylight Time," or "Pacific Standard Time"

Field	Description
x-event	Type of event.
x-category	Event category.
date	Date of the event.
time	Time the event occurred.

Field	Description
tz	Time zone information.
x-ctx	Event-dependent context information.
s-ip	IP address or addresses of the server.
x-pid	Server process ID.
x-cpu-load	CPU load.
x-mem-load	Memory usage (as reported by the <code>getServerStats()</code> method).
x-adaptor	Adaptor name.
x-vhost	Virtual host name.
x-app	Application names.
x-appinst	Application instance name.
x-duration	Duration of a stream or session event. For a stream, this value is the number of seconds the stream has played. For a session, this value is the number of seconds the client has been connected.
x-status	For a complete description of the x-status codes and descriptions, see "Event status codes in access logs" on page 103.
c-ip	Client IP address.
c-proto	Connection protocol.
c-proto-ver	Connection protocol version.
s-uri	URI of the Adobe Media Server application.
cs-uri-stem	The stem portion of the s-uri field.
cs-uri-query	The query portion of the s-uri field.
c-referrer	URI of the referrer.
c-user-agent	User agent.
c-client-id	Client ID.
cs-bytes	This field shows the number of bytes transferred from the client to the server. This information can be used to bill customers per session. To calculate the bandwidth usage per session, subtract cs-bytes in the "connect" event from cs-bytes in the "disconnect" event.
sc-bytes	This field shows the number of bytes transferred from the server to the client. This information can be used to bill customers per session. To calculate the bandwidth usage per session, subtract sc-bytes in the "connect" event by sc-bytes in the "disconnect" event
c-connect-type	Type of connection received by the server: Normal: Connection from a client, such as Flash Player Group: Connection between an edge and an origin server Virtual: Client connection that goes through an edge server, using the group connection between the servers for transmission
x-sname	Stream name.
x-sname-query	Query portion of the stream URI specified in the call to play or publish.
x-suri-query	Same as x-sname-query.

Field	Description
x-suri-stem	This is a composite field: cs-uri-stem + x-sname + x-file-ext.
x-suri	This is a composite field: cs-uri-stem + x-sname + x-file-ext + x-sname-query.
x-file-name	Full path of the file representing the x-sname stream.
x-file-ext	Stream type (FLV, MP3, or MP4).
x-file-size	Stream size, in bytes.
x-file-length	Stream length, in seconds.
x-spos	Stream position, in milliseconds.
cs-stream-bytes	This field shows the number of bytes transferred from the client to the server per stream. To calculate the bandwidth usage per stream, subtract cs-stream-bytes in the "publish" event from cs-stream-bytes in the "unpublish" event.
sc-stream-bytes	This field shows the number of bytes transferred from the server to the client per stream. This value can be less than sc-bytes which is the total number of bytes transferred from the server to the client. When possible, use sc-bytes for billing. To calculate the bandwidth usage per stream, subtract sc-stream-bytes in the "play" event from sc-stream-bytes in the "stop" event. The value of sc-stream-bytes can be greater than x-file-size after streaming files not encoded in FLV format, such as MP3 files. Note: The value of sc-stream-bytes is not necessarily the same as the value of the QoS ByteCount property.
x-service-name	Name of the service providing the connection (only applicable to certain connection types).
x-sc-qos-bytes	Number of bytes sent to the client for quality of service.
x-comment	Comments.
x-sid	The ID of a stream. This ID is unique for the client session but not across sessions.
x-trans-sname	The name of the stream that the server transitions from (the original stream).
x-trans-sname-query	The query stream portion of the stream name for the stream that the server transitions from.
x-trans-file-ext	The file extension portion of the stream name for the stream that the server transitions from.
x-plugin	Name of the plug-in. This field is only available in authorization (auth-) events.
x-page-url	The URL of the web page in which the client SWF file is embedded.
x-smax-rec-size	The maximum file size of a recorded stream.
x-smax-rec-duration	The maximum duration of a recorded stream.
x-trans-mode	The transition mode sent by the client in the <code>NetStream.play2()</code> call.
x-soffset	When a stream is reconnected, the offset value indicates where to resume streaming.
c-spos	The client stream position when a "client-pause" or "client-seek" event is logged.
c-proto-ver	The RTMPE version. For all other types of RTMP, this field is empty.
x-eid	An event ID received by Authorization plug-in. This event is visible only in the auth.log file. This field is empty in the access.log file.

Field	Description
x-codec-type	Codec type of the frame retrieved in the Authorization plug-in's E_CODEC event. This event is visible only in the auth.log file. This field is empty in the access.log file.
x-codec-value	Codec value of the "x-codec-type" retrieved in the Authorization plug-in's E_CODEC_CHANGE event. This event is visible only in the auth.log file. This field is empty in the access.log file.
x-forwarded-for	A string inserted by an HTTP proxy that usually contains the IP address of the originating client. This string can contain several IP address or other values. Adobe Media Server copies the string and reports it unchanged. For more information, see en.wikipedia.org/wiki/X-Forwarded-For .

Event status codes in access logs

The Event status codes are based on HTTP response codes.

Field	Symbol	Status Code	Description
connect pending	status_continue	100	Waiting for the application to authenticate.
disconnect	status_admin_command	102	Client disconnected due to admin command.
disconnect	status_shutdown	103	Client disconnected due to server shutdown (or application unloaded).
connect, publish, unpublish, play, record, record stop, stop	status_OK	200	Successful.
play, stop	status_transition	210	A transition between streams has occurred.
connect	status_unavailable	302	Application currently unavailable.
connect, publish, play	status_bad_request	400	Bad request; invalid parameter or client connected to server using an unknown protocol.
connect, play, publish	status_unauthorized	401	Connection rejected by application script or access denied by application.
connect	status_forbidden	403	Connection rejected by Authorization plug-in or connection rejected due to invalid URI.
connect, play	object_not_found	404	Application or stream not found.
play	client_disconnect	408	Stream stopped because client disconnected.
connect, publish	status_conflict	409	Resource limit exceeded or stream is already being published. Can also mean that a change has been made by the Authorization plug-in.
connect	status_lic_limit_exceeded	413	License limit exceeded.
play, publish	unsupported_type	415	Unsupported media type.
disconnect	data_exceeded	416	Message queue too large; disconnect the client.
connect	chunkstream_error	417	Unable to process unknown data type.
disconnect	cannot_broadcast	418	Client does not have privilege to broadcast.
disconnect	cannot_screenshare	419	License to receive screen sharing video failed.
disconnect	remote_link_closed	420	Close downstream connection.

Field	Symbol	Status Code	Description
connect	process_msg_failed	422	Unable to process message received when client connection was in pending or closed state.
disconnect	process_msg_exception	423	Error handling message.
disconnect	process_remote_msg_failed	424	Expected response not provided when command was issued.
disconnect	process_admin_msg_failed	425	Expected response not provided when issued an admin command.
disconnect	process_rtmp_S2S_msg_failed	426	Expected response not provided when command issued.
disconnect	write_error	427	Client is not connected or client terminated; unable to write data.
disconnect	invalid_session	428	Client connection invalid; closed due to inactive or idle status.
disconnect	gc_client	429	Unable to obtain ping response or client states not connected.
disconnect	remote_onstop	430	Upstream connection closed.
disconnect	remote_on_client_disconnect	431	Upstream connection closed because the last client disconnected.
disconnect	gc_idle_client	432	Adobe Media Server autoclose feature automatically closed the connection.
disconnect	swf_hash_fail	433	SWF verification failure.
disconnect	swf_hash_timeout	434	SWF verification timeout.
disconnect	encoding_mismatch_error	435	Client disconnected due to incompatibility with object encoding.
disconnect	invalid_client	437	Client connects and fails to perform the standard RTMP handshaking.
play	invalid_transition	438	Client attempted to play a non-existent file during dynamic streaming.
disconnect	swf_hash_wrong_version	439	This version of SWF Verification version is not allowed.
stop	live_stream_destroyed	440	A call to <code>Stream.destroy()</code> destroyed an instance of the server-side Stream class. Publishers are disconnected and recordings are stopped.
disconnect, play	server_internal_error	500	Server internal error.
connect	bad_gateway	502	Bad gateway.
connect	service_unavailable	503	Service unavailable; for instance, too many connections pending for authorization by access module.
disconnect	js_disconnect	600	Application disconnect.
disconnect	js_close_previous_client	601	Network connection was closed or reused.

Field	Symbol	Status Code	Description
disconnect	js_exception	602	An unknown exception is thrown from the JS engine.
disconnect	js_chunkstream_error	603	Bad application data.
disconnect	js_debug_forbidden	604	Application does not allow debug connections.
play	js_gc_object	605	~fcstreamjshook() clean up.

Application logs

Application log file

The application log records information about activities in application instances. This log is used primarily for debugging (logging uncommon events that occur in an application instance).

The default application log is application.xx.log, located in the subdirectory within the Adobe Media Server logs directory. Adobe Media Server is configured, by default, to create one application log per application instance. The application folder is located in the matching virtual host directory. The “xx” in the filename is a two-digit number representing the history of the application log. The most recent logs can be found in application.00.log.

Fields in application logs

Field	Event(s)	Description
date	All	Date of the event.
time	All	Time of the event.
x-pid	All	Server process ID.
x-status	All	<p>Status code: The code is a 10-character string that represents the severity, category, and message ID.</p> <p>The first three characters represent severity, as follows:</p> <ul style="list-style-type: none"> (w) = warning (e) = error (i) = information (d) = debug (s) = trace from server-side script (_) = unknown <p>The next three characters represent the category. All categories are listed in “Status categories in diagnostic logs” on page 106.</p> <p>The last four characters represent message ID. All message IDs are listed in “Diagnostic logs” on page 106.</p>
x-ctx	All	Event-dependent context information.

Diagnostic logs

Diagnostic log file

The diagnostic log records information about Adobe Media Server operations (this is in addition to the information logged by the operating system). This log is used primarily for debugging (logging uncommon events that occur in Adobe Media Server processes).

The default diagnostic logs are master.xx.log, edge.xx.log, core.xx.log, admin.xx.log, and httpcache.xx.log. All the diagnostic logs are located in the Adobe Media Server logs directory. Adobe Media Server is configured, by default, to create a diagnostic log for each type of process. The “xx” in the filename is a two-digit number representing the version of the log.

For a list of messages that appear in the diagnostic log files, see [“Diagnostic logs”](#) on page 106.

Fields in diagnostic logs

Field	Event(s)	Description
date	All	Date on which the event occurred.
time	All	Time at which event occurred.
x-pid	All	Server process ID.
x-status	All	<p>Status code: The code is a 10-character string that represents the severity, category, and message ID.</p> <p>The first three characters represent severity, as follows:</p> <ul style="list-style-type: none"> (w) = warning (e) = error (i) = information (d) = debug (s) = trace from server-side script (_) = unknown <p>The next three characters represent category. All categories are listed in “Status categories in diagnostic logs” on page 106.</p> <p>The last four characters represent the message ID.</p>
x-stx	All	Event-dependent context information.

Status categories in diagnostic logs

Category	Description
257	TCSERVICE
258	TCSERVER
259	Presence
260	Storage
261	Stream

Category	Description
262	SMTP
263	Adaptor
264	JavaScript
265	TCApplcation
266	TCCconnector
267	Admin
268	SharedObject
269	Configuration
270	VirtualHost
271	SSL

Chapter 5: Administering the server

Perform regular administrative tasks to keep the server running smoothly.

Start and stop the server

Start and stop the server on Windows

Use one of the following methods to shut down or restart the server.

Start the server from the Start menu

❖ Do one of the following:

- Choose Start > All Programs > Adobe > Adobe Media Server 5 > Start Adobe Media Server 5
- Choose Start > All Programs > Adobe > Adobe Media Server 5 > Start Adobe Media Administration Server 5

Stop the server from the Start menu

❖ Do one of the following:

- Choose Start > All Programs > Adobe > Adobe Media Server 5 > Stop Adobe Media Administration Server 5
- Choose Start > All Programs > Adobe > Adobe Media Server 5 > Stop Adobe Media Server 5

Start, stop, or restart the server from the Services window

1 Choose Start > Control Panel > Administrative Tools > Services.

2 Do one of the following:

- Select Adobe Media Server from the Services list and click Stop, Start, or Restart.
- Select Adobe Media Administration Server from the Services list and click Stop, Start, or Restart.

Start and stop the server on Linux

Start, stop, or restart Adobe Media Server using `amsmgr`

1 Log in as a root user.

2 Change to the directory where the server is installed.

3 Open a shell window and type one of the following:

- `./amsmgr server start`
- `./amsmgr server stop`
- `./amsmgr server restart`

Start, stop, or restart the Administration Server using `amsmgr`

1 Log in as a root user.

- 2 Change to the directory where the server is installed.
- 3 Open a shell window and type one of the following:
 - `./amsmgr adminserver start`
 - `./amsmgr adminserver stop`
 - `./amsmgr adminserver restart`

Start, stop, or restart Adobe Media Server using the command line

- 1 * `cd /<the directory where Adobe Media Server is installed>`.
- 2 Enter `./server [start | stop | restart]`

Start, stop, or restart the Administration Server using the command line

- 1 * `cd /<the directory where Adobe Media Server is installed>`.
- 2 Enter `./adminserver [start | stop | restart]`

For more information about using `amsmgr`, see [“Managing the server on Linux”](#) on page 115.

Checking server status

View server events in the Windows Event Viewer

The Windows Event Viewer can be used for tracking Adobe Media Server activity and debugging server applications. The Event Viewer displays a list of events that the server generates. (The following steps are accurate if you are working directly on the server. To view the events from another Windows machine, use Event Viewer to open a remote connection to the server.)

- 1 From the Windows Start menu, select Settings > Control Panel > Administrative Tools > Event Viewer.
- 2 Select the Application panel.
- 3 Double-click an event generated by Adobe Media Server to view details.

Check server health

AMSCheck is a command line utility program that can be used to diagnose and determine server status. The tool is available for both Windows and Linux using different executable files. As a command line tool, AMSCheck is completely scriptable using the language of your choice, such as Cscript, bash, C shell, or Python. AMSCheck provides information about whether the server is running or not, what the response time is, and which `amscore` processes are not responding. A small video file for testing is included. The `Users.xml` file must be configured to accept a connection from this tool (this configuration is required to use `--allapps` and its dependent commands).

When the tool connects to Adobe Media Server, it does the following:

- Checks the connection to any instance of an application
- Checks all active instances of the server by connecting to those applications
- Can publish and play a stream
- Can play the available server-side stream for an application

Note: Currently, AMSCheck only supports RTMP connections and does not check for shared objects.

AMSCheck commands and options

Option	Description
--host <hostname>	Required; tells the program where to connect the server. Example: --host localhost
--port <number>	Port number is optional. The default value is 1935. Example: --port 1935
--app <app>	Allows the program to connect to an application. Administrator must specify the application. Example: --app appl/inst1
--allapps	Queries the Administration Server for active instances and makes a connection to each active instance. In this case, the administrator must use the --auser, --apswd, and --ahostport options in order to log in to the Administration Server. The administrator must configure Users.xml to accept connections from this program. This command can take time to finish; verify that the timeout value is adequate.
--help	Displays Help for using AMSCheck.
--logfile <file>	Allows the program to output a response to a file. If this option is not specified, a result cannot be provided. Example: --logfile output.log
--play <name> [start [duration]]	Instructs the program to play video files. Options are start and duration. Values for start and duration must be in positive numbers or 0 and represent the number of seconds. The default value of start is any, which plays the file from the beginning. The default value of duration is 1 second. You can specify all to play the entire file. You cannot give the play and publish commands at the same time. Example: --play foo 10 5
--publish <name> <duration> [record append]	Publishes files to the server. This command must be used along with --pubfile. The duration parameter is required; only a positive number, zero, or all is allowed. Both record and append are optional. If neither is specified, the default behavior is to record. If the file already exists and record is used, the existing file is overwritten. After the file is published, it is automatically played to verify the success of the publish operation. Example: --publish foo -1
--pubfile <file>	Specifies a filename. This command must be used with --publish. Specify the name of the input video file residing on the client side, the name of the output file to be created on server side, and the duration. Example: --pubfile input.flv --publish output 10
--parallel [<max>]	Allows the program to play multiple applications at the same time. This command is used with --allapps. If there is more than one application, tests are run on each application serially (connect to the first application, run test, connect to the second application, run test, and so on). Running parallel without specifying max tests every application in parallel. However, if there is a large number of applications, running all of them in parallel may not be desirable. Indicate the maximum number of applications that can be run in parallel by specifying a value for max. For example, to run 10 tests in parallel, use the following: --parallel 10
--stagger <sec>	Inserts a pause between tests. This command is used along with --parallel. The value of <sec> is in seconds, and the default value is 1 second. If you specify a very long stagger time (longer than the duration of the test), then you are effectively running in serial mode. Example: --stagger 2
--query <" ">	Allows you to input your own string for special purposes, such as authentication. Example: rtmp://host/app/inst?foo=abcd
--timeout <sec>	Specifies a timeout value, in seconds. If the program does not receive a response from the server within this interval, an error is returned.

Option	Description
<code>amscheck -v</code>	Prints a version string.
<code>--auser <username></code>	Specifies a user name for the Administration Server user. Example: <code>--auser admin</code>
<code>--apswd <password></code>	Specifies a password for the Administration Server. Example: <code>--apswd admin</code>
<code>--ahostport <port></code>	Specifies the Administration Server port number. If the port number is not specified in the command line, the default port is 1111. Example: <code>--ahostport 1111</code>

Usage examples for Windows:

- `* amscheck.exe --host localhost --app appl --logfile output.txt`
- `* amscheck.exe --host localhost --app appl --play foo 0 10 --logfile output.txt`
- `* amscheck.exe --host localhost --app appl --pubfile foo.flv --publish bar 10 --logfile output.txt`
- `* amscheck.exe --host localhost --allapps --auser admin --apswd admin --parallel 10 --stagger 2 --timeout 100 logfile output.txt`

All of the Windows examples can be adapted to Linux by using `* ./amscheck` instead of `* amscheck.exe`.

AMSCheck return codes

Return codes for the AMSCheck tool report the status after the tool has been run.

Code	Status
0	Success.
-1	Invalid command line argument.
-2	File not found.
-3	Connection failed.
-4	Operation timed out.
-5	Play failed.
-6	Publish failed.
-7	At least one application failed.

Checking video files

Checking FLV files created or modified with third-party tools

Third-party tools are available to create and modify FLV files, but some of the tools create files that do not comply with the FLV standard. Common problems include bad timestamps in the FLV file, invalid `onMetaData` messages, bad message headers, and corrupted audio and video. The FLVCheck tool can be used to analyze FLV files before they are deployed on Adobe Media Server. In addition, the tool can also add or update metadata to reflect file duration correctly. The tool verifies that metadata is readable, specifies an accurate duration, and checks that the FLV file is seekable by Adobe Media Server. The tool supports unicode filenames.

Note: The FLVCheck tool does not correct FLV file content corruption. The tool does fix metadata by scanning the Duration and Can Seek To End metadata fields. The tool can then merge the server metadata with the data present in the file.

Checking other video files

Adobe Media Server supports playback and recording of H.264-encoded video and HE-AAC-encoded audio within an MPEG-4-based container format. A subset of the MPEG-4 standards are supported. All MP4 files and Adobe F4V files are part of the supported subset.

For MPEG-4-based container formats, use the FLVCheck tool to verify that the server can play back your files.

Note: The FLVCheck tool does not correct corrupted H.264-encoded files or make any other fixes to MP4/F4V files.

Check a video file with the FLVCheck tool

The FLVCheck tool is a command line program; the executable is named flvcheck.

- 1 Open your operating system's command prompt and change directories to *RootInstall/tools*.
- 2 Use the following syntax to run the FLVCheck tool:

```
flvcheck [-option] <file ...>
```

For example, to check two files:

```
flvcheck -f abc.flv ../test/123.flv
```

The following table describes the command line options available.

Option	Description
-f [--file] file ...	Specifies the path to the video file(s) being checked. Relative paths may be used. (Avoid using the "\xd3 character; try the "/" character instead.)
-v [--verbose]	Sets the verbose flag.
-V [--version]	Prints version information.
-n [--nobanner]	Turns off header.
-h [--help]	Provides a description of options and an example.
-d [--duration]	Specifies the margin of error, in seconds, that FLVCheck reports. (The default is 2 seconds.) When validating metadata, the absolute difference between <code>metadata_duration</code> and <code>actual_duration</code> is calculated and compared against the margin specified in this command. If the margin is exceeded, the server logs a warning that the metadata duration is incorrect. If the margin has not been exceeded, nothing will be logged. To get the exact duration, specify <code>-d 0</code> .
-q [--quiet]	Specifies that only the status code, not the text output, be returned. The <code>--help</code> option overrides this option.
-w [--warnings]	Display warnings.
-W [--warnings_as_errors]	Treat warnings as errors.

Option	Description
-s [--fixvideostall]	Fix a stall in video playback (FLV only).
-u [--usage]	Displays an example and information about command-line parameters.
-m [--fixmeta]	(FLV files only) If a metadata tag is corrupted, creates a new copy of the original FLV file in the same directory as the original, with corrected metadata. (If the file contains no errors, a backup file is not created.) Only the Duration and Can Seek To End metadata fields are corrected.

3 If the FLVCheck tool finds no errors in the FLV file, the status code returned is 0. If there are one or more errors, a positive number indicating the total number of invalid files found is returned. If a return code of -1 is returned, an invalid command-line parameter was specified.

Errors and warnings are logged in a log file (stdout).

4 (FLV files only) If an error is returned from an FLV file due to a metadata error, you can use the tool to try to correct the problem. Try the following:

a Use the -m option to try to fix the metadata in the file:

```
flvcheck -m <file> [-quiet] [-help]
```

b Use the -d option to change the duration field margin of error. The duration field in the metadata may be inaccurate by a few seconds. For example, `flvcheck -f abc.flv -d 5` would allow the metadata duration to be inaccurate +/- 5 seconds.

Other types of errors cannot be fixed using the FLVCheck tool. MP4/F4V files cannot be fixed using the FLVCheck tool.

FLVCheck errors

If an error is found, the error is logged to the stdout file in the following format: Date, Time, ErrorNumber, ErrorMessage, and FileName. The possible error numbers, types of errors, and messages are as follows.

Error numbers	Error type	Error messages
-2	General	Invalid file system path specified.
-3	General	File not found.
-4	General	Cannot open file.
-5	General	File read error. Adobe Media Server cannot read the file, indicating that the encoding of part or all of the file is not compatible with the codecs that are supported.
-6	General	Cannot create corrected file. This error occurs if you run the tool with the -m option set, but the tool cannot create a file with corrected metadata.
-7	FLV	Invalid FLV signature.
-8	FLV	Invalid FLV data offset.
-9	FLV	Invalid FLV message footer.
-10	FLV	Unrecognized message type.
-11	FLV	Found backward timestamp.

Error numbers	Error type	Error messages
-12	FLV	Unparsable data message.
-13	MP4	File does not contain a movie box. This error occurs if the MP4 file is empty.
-14	MP4	File does not contain any valid tracks. This error could occur if the MP4 file contains audio or video encoded with unsupported codecs.
-15	MP4	Too many tracks. Maximum allowed is 64.
-16	MP4	Only one sample type allowed per track.
-17	MP4	Box is too large.
-18	MP4	Truncated box. The reported length of a box is longer than the remaining length of the file. The file may have been truncated, or the reported box length may be invalid.
-19	MP4	Duplicate box.
-20	MP4	Invalid box version.
-21	MP4	Invalid movie time scale.
-22	MP4	Invalid number of data entries in box.
-23	MP4	Invalid sample size.
-24	MP4	Invalid chapter time.
-25	MP4	Too many tag boxes. Max is 64.
-26	General	File appears to be FLV with wrong extension.
-27	MP4	Unsupported DRM scheme.
-28	MP4	Error reading MP4 tables.
-29	MP4	File contains unexpected movie fragments.
-30	MP4	File contains out-of-order movie fragments.
-127	MP4	Found negative CTTS Offset. The MP4 standard stores the CTTS offset as an unsigned 32-bit value. The RTMP standard stores the CTTS offset as a 24-bit unsigned value. However, if a file has negative CTTS offsets, the value becomes very large when converted to unsigned 32-bit, and RTMP strips off 8 bits. This behavior may cause artifacts to be seen during playback.

FLVCheck warnings

Generally, warnings are informative and are not fatal errors; Adobe Media Server will ignore the error that caused the warning and continue to load and play back the video or audio file, but you may experience problems with playback. Warnings are logged to the stdout file in the following format: Date, Time, Warning Number, Warning Message, and File Name.

Warning number	Warning type	Message
-100	General	Metadata duration is missing or is incorrect.
-101	FLV	canSeekToEnd is false.
-102	MP4	Unrecognized box.
-103	MP4	Found incomplete track.
-104	MP4	Found duplicate video track. Ignoring...
-105	MP4	Found duplicate audio track. Ignoring...
-106	MP4	Found duplicate data track. Ignoring...
-107	MP4	Track has unsupported sample type. Adobe Media Server ignores (will not play back) tracks that are encoded with unsupported codecs.
-108	MP4	Invalid video codec. This warning indicates that a track has an invalid video codec. Adobe Media Server cannot play back the track.
-109	MP4	Invalid audio codec. This warning indicates that a track has an invalid audio codec. Adobe Media Server cannot play back the track.
-110	FLV	Video may appear stalled due to lack of audio data.
-111	MP4	File has unsupported metadata format.
-112	MP4	Box has extraneous bytes at end.
-113	FLV	Video messages found but video flag not set.
-114	FLV	Audio messages found but audio flag not set.
-115	FLV	Video flag set but no video messages found.
-116	FLV	Audio flag set but no audio messages found.
-117	MP4	File is truncated. Will only be partially playable.
-118	MP4	Track contains unsupported edit list.
-119	FLV	Missing FLV metadata.
-120	MP4	Bad NellyMoser Frequency. Sample(s) skipped.
-121	MP4	Invalid Track Extends Box.
-122	MP4	Track contains unsupported sample flags.

Managing the server on Linux

Use the `amsmgr` utility to perform basic management tasks for Adobe Media Server running on Linux systems, such as starting and stopping the server and services. You must be a root user to use the `amsmgr` utility.

For tasks not listed in the following table, such as adding users or checking the status of applications, use the Administration Console. A root user must use the `amsmgr` utility to start the Adobe Media Administration Server before anyone can use the Administration Console. You do not need to be a root user to use the Administration Console.

Note: Running multiple Adobe Media Server services concurrently is not supported.

Syntax

`amsmgr server <service_name> <cmd>`

For commands that use the `<service_name>` parameter, if you do not specify `<service_name>`, the command is performed on the name of the server you selected during installation. The server name is `ams` by default. If `<service_name>` does not exist, the command fails.

The `amsmgr` utility supports the following commands:

Command	Description
<code>amsmgr adminserver start stop restart</code>	Starts, stops, or restarts Adobe Media Administration Server.
<code>amsmgr server <service_name> start stop restart</code>	Starts, stops, or restarts a Adobe Media Server service.
<code>amsmgr clearautostart</code>	Sets the Adobe Media Administration service to start manually.
<code>amsmgr list</code>	Lists all the services installed, including Administration services, with additional information about services that are currently running.
<code>amsmgr remove</code>	Removes the Adobe Media Server service from the <code>amsmgr</code> tables. If you remove a server service, the corresponding Administration service is also removed. Warning: Use this command only if you want to uninstall the server service; you still need to manually remove the installed files.
<code>amsmgr add <service_name> <install_dir></code>	Add an Adobe Media Server service to the <code>amsmgr</code> tables. If <code><service_name></code> already appears in the <code>amsmgr</code> tables, the old entry is updated with the new one. The <code><install_dir></code> parameter is the absolute directory path where you installed Adobe Media Server.
<code>amsmgr setadmin <service_name></code>	Changes the default Administration service. The Administration service name is the same as the Adobe Media Server service name. Any installed Administration service can be used to administer one or more servers. Only one Administration service can be running at a time.
<code>amsmgr getadmin</code>	Gets the name of the default Administration service.
<code>amsmgr setautostart<service_name></code>	Sets the Adobe Media Server service to start automatically when the system is started.
<code>amsmgr clearautostart <service_name></code>	Removes the named service from the set of services to start automatically when the system is started.
<code>amsmgr suggestname</code>	Suggests a service name that does not already appear in the <code>amsmgr</code> tables.

Scramble tool

The scramble tool obfuscates data on disk to prevent people who have access to the system from reading sensitive information. Protected HTTP Dynamic Streaming (PHDS) and Protected HTTP Live Streaming (PHLS) use the scramble tool to obfuscate credential passwords and the content encryption key. To create a new content encryption key, use the scramble tool.

The scramble tool uses AES-128 and a hard-coded key to obfuscate data. The data is unscrambled using the same hard-coded key. The key is a random sequence of bytes generated by the scramble tool.

Important: *The hard-coded key can be retrieved from source code. It's a good idea to use additional techniques to secure the data as well.*

The scramble tool is installed to the following location:

rootinstall/tools/scramble

The scramble tool supports the following options:

Option	Description
<code>-scramble <string-to-scramble></code>	Takes an input string from the command line, encrypts it, and prints out a Base64 encoded string. Use this option to encrypt credential passwords. This option assumes that the string to scramble is already Base64 encoded. The tool doesn't decode the string.
<code>-randCode <number-of-bytes></code>	Generates a sequence of random bytes. Specify the sequence length in the <code><number-of-bytes></code> command line parameter. The output is formatted as a C++ array initialization code snippet. The installer uses this option to generate the scrambling key and IV that are hard-coded in the scrambler tool and the unscrambling utility function.
<code>-randBase64 <number-of-bytes></code>	Generates a sequence of random bytes. Specify the sequence length in the <code><number-of-bytes></code> command line parameter. The output is a Base64 string encoding the random data. Use this option to generate a content encryption key. Use the output of this command as the <code><string-to-scramble></code> input for the <code>-scramble</code> option. Combining those two commands generates data in the same format as the <code>-KeyGen</code> option.
<code>-KeyGen <number-of-bytes> [-f <output-file>]</code>	<p>Generates a sequence of random bytes, Base64 encodes it, encrypts the resulting string and prints out a Base64 encoded string of the result.</p> <p>Specify the sequence length in the <code><number-of-bytes></code> command line parameter.</p> <p>Use the <code>-f</code> option to redirect the output to an output file.</p> <p>The Adobe Media Server installer uses this option to generate a unique PHDS and PHLS content encryption key.</p>

For example, the following command outputs a 16 byte key:

```
scramble -KeyGen 16 -f server.key
```

Chapter 6: Using the Administration API

Working with the Administration API

About the Administration API

Use the Administration API to monitor, manage, and configure the server from an Adobe Flash Player or Adobe AIR client over RTMP or RTMPE or from a web client over HTTP. The Adobe Media Server Administration Console was built using the Administration API. The API is documented in [Adobe Media Server Administration API Reference](#).

Here are a few important tips for working with the Administration API:

- Both Adobe Media Server and Adobe Media Administration Server must be running.
- This document assumes that you have not changed the default port number (1111) for the Administration Server; if you have, use your valid port number in all examples.
- If you do not specify an optional parameter, a default value may be used depending on the method. For example, if you do not specify a virtual host in the `scope` parameter, it is assumed that you want to perform the command on the virtual host to which you connected when you logged in to Adobe Media Server.
- By default, administrators are logged in to the “_defaultRoot_” adaptor. When logging in to a virtual host not on the default adaptor, virtual-host administrators must specify the name of the adaptor. For example, when logging in (over RTMP using `NetConnection`) to a virtual host on the adaptor `_secondAdaptor_`, the administrator `JLee` would enter the following information for the administrator user name parameter in the method call:
`_secondAdaptor_/JLee`.

Set permissions for Administration API method calls over HTTP

Note: You do not need to set permissions to call methods over RTMP.

- 1 Open the `ams.ini` file in the `RootInstall/conf` folder.
- 2 Make sure that the `USERS.HTTPCOMMAND_ALLOW` parameter is set to `true`.
- 3 Open the `Users.xml` file in the `RootInstall/conf` folder.
- 4 In the `<Allow>` element, enter method names in a comma-delimited list to allow HTTP requests to execute Administration API calls. For example, the following code allows the `ping()` and `changePswd()` methods:

```
<AdminServer>
  <HTTPCommands>
    <Enable>${USERS.HTTPCOMMAND_ALLOW}/Enable>
    <Allow>ping, changePswd</Allow>
    <Deny>All</Deny>
    <Order>Deny, Allow</Deny>
  </HTTPCommands>
</AdminServer>
```

Note: There are additional XML elements in the `Users.xml` file that give you finer control over permissions. For more information, see “XML configuration files reference” on page 125.

- 5 Save the file and restart Adobe Media Administration Server.

Call an Administration API method over HTTP

- 1 Verify that Adobe Media Administration Server is running.
- 2 Open a browser window and enter the following in the address bar:

```
http://localhost:1111/admin/ping?auser=username&apswd=password
```

If the web browser is on a different computer than Adobe Media Server, use the server address instead of localhost. Substitute your administrator user name and password, which are located in the ams.ini file.

Note: You can construct an Administration API call from any language that can send HTTP requests. This example uses a web browser because it's the simplest client and good for testing purposes.

- 3 The server sends the XML results back to the browser:

```
<?xml version="1.0" encoding="utf-8" ?>
<result>
  <level>status</level>
  <code>NetConnection.Call.Success</code>
  <timestamp>10/3/2007 05:31:01 PM</timestamp>
</result>
```

Constructing an HTTP request string

Many Administration APIs expect one or more parameters in addition to `auser` and `apswd`. Parameters passed in an HTTP request must adhere to the following formatting rules:

- Name the parameters in the string. For example, the following code calls the `addAdmin()` method:

```
http://localhost:1111/admin/addAdmin?auser=adminname&apswd=adminpassword&username="joe"&password="pass123"&vhost="_defaultRoot_/foo.myCompany.com"
```

- Strings are passed as literals surrounded by quotation marks. You can use either single quotation marks (`'`) or double quotation marks (`"`).

```
"Hello World"
```

```
'String2'
```

The only exceptions are the `auser` and `apswd` parameters, which should be passed without quotation marks.

- Numbers are passed as either integer or floating-point literals.

```
245
```

```
1.03
```

```
-45
```

Note: When a number is used for an application name, the application name must be included within quotation marks (`"`) for methods such as `reloadApp()` and `unloadApp()` to work properly.

- Arrays are passed as comma-separated values enclosed in square brackets.

```
[1,2,3,4]
```

```
['abcd',34,"hi"]
```

```
[10,20,[31,32],40]
```

- Objects are passed as inline object literals.

```
{foo:123,bar:456}
```

```
{user:"Joe",ssn:"123-45-6789"}
```

Call Administration API methods over RTMP or RTMPE

To call the Administration API over RTMP or RTMPE, you need a Flash Player or AIR client.

Note: To call the Administration API over RTMPE, follow the instructions below, but change the protocol in the example `NetConnection.connect()` call to RTMPE.

- 1 In the client application, create a `NetConnection` to Adobe Media Administration Server, passing in three parameters: the URI of the Administration Server, an administrator user name, and an administrator password. Only valid administrators, as defined in the `Users.xml` configuration file, can connect to the server.

The following code creates a `NetConnection` for the administrator `MHill` with password `635xjh27` to the server on `localhost`:

```
ncAdmin = new NetConnection();  
ncAdmin.connect("rtmp://localhost:1111/admin", "MHill", "635xjh27");
```

Note: If you want to connect to a virtual host, specify the virtual host's domain name or IP address as part of the URI—for example, `rtmp://www.myVhost.com/admin:1111`.

- 2 Call the `NetConnection.call()` method and pass it the Administration API method, a response object (if needed) and any parameters (if needed):

```
ncAdmin.call("getAppStats", new onGetAppStats(), "vod");
```

The `getAppStats()` method returns performance data for an application running on the server; the response object `onGetAppStats()` captures the result of the call; and `vod` is the name of the application instance from which to retrieve statistics.

- 3 Define a function to handle the information object returned by the Administration API method; in this example, `onGetAppStats()`:

```
ncAdmin.call("getAppStats", new onGetAppStats(), appName.text);  
...  
function onGetAppStats(){  
    this.onResult = function(info){  
        if (info.code != "NetConnection.Call.Success"){  
            outputBox.text = "Call failed: " + info.description;  
        } else {  
            outputBox.text = "Info for "+appName.text+ " returned:" + newline;  
            printObj(info, outputBox);  
        }  
    }  
}
```

The data is returned to the handler function in an information object. All information objects have `level`, `code`, and `timestamp` properties. Some information objects have a `data` property (containing return data, often in an object or array) and `description` and `details` properties, which typically provide information about errors.

Create your first Flash application

This section contains the code for a simple Flash application that calls the `getAppStats()` method.

To call Administration APIs over RTMP, you need a Flash Player or AIR client. The following sample is built in Flash.

Note: You can call Administration APIs from applications written in ActionScript 1.0, ActionScript 2.0, or ActionScript 3.0.

1 In Flash, create an application with the following elements:

- An input text field named `appName`
- A button called `button_btn`
- A multiline, dynamic text field called `outputBox`
- A scroll component attached to the `outputBox` text field

Note: Because this is a Adobe Media Server application, you must create an application directory with the application name in the `RootInstall\applications` directory.

2 Enter the following code on frame 1 of a Flash file:

```
/** Establishes the connection to Adobe Media Server */

ncAdmin = new NetConnection();
// Replace admin_name and admin_pass with your
// administrator name and password.
ncAdmin.connect("rtmp://localhost:1111/admin","admin_name","admin_pass");

/** Makes the API call, for example, "getAppStats" */
function doGetAppStats() {
    function onGetAppStats(){
        this.onResult = function(info){
            if (info.code != "NetConnection.Call.Success"){
                outputBox.text = "Call failed: " + info.description;
            } else {
                outputBox.text = "Info for "+appName.text+ " returned:" + newline;
                printObj(info, outputBox);
            }
        };
    }
    // Calls the getAppStats() API on the name of application
    // in the input text field
    // places the response in the onGetAppStats funtion
    ncAdmin.call("getAppStats", new onGetAppStats(), appName.text);
}

function printObj(obj, destination){
    for (var prop in obj) {
        destination.text += "\t";
        destination.text += prop + " = " + obj[prop] + newline;
        if (typeof (obj[prop]) == "object") { // recursively call printObj
            printObj(obj[prop], destination);
        }
    }
}

button_btn.onRelease = function(){
    doGetAppStats();
}
```

3 Before running this sample application, start another Adobe Media Server application.

4 Open the Administration Console to verify that the application you started in step 3 is running.

5 Run the sample Flash application and enter the name of the application you started in step 3 in the input text field.

This example uses the `NetConnection` class to make a single request to the Adobe Media Server. If you want to create a persistent connection that updates automatically (like the Administration Console), you can use the `NetStream` class to create a streaming connection over the `NetConnection`.

Create your first application in Flex

The following application built in Flex uses the `HTTPService` class to call the Adobe Media Server administration API. It calls the `getAppStats` method and prints the results to the `TextArea` control.

Before you compile and run this example, you must add your admin username and password to the request parameters.

When you run the example, enter an application name in the `TextInput` control and click the `Get App Stats` button. Default applications include “live” and “vod”, but you can use the name of any application that you created on the server.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/mx">
  <fx:Declarations>
    <s:HTTPService id="myService"
      url="http://localhost:1111/admin/getAppStats"
      method="GET"
      resultFormat="e4x"
      result="myServiceResultHandler()">
      <s:request xmlns="">
        <!-- Replace these with your admin username and password. -->
        <auser>your_admin_username</auser>
        <apswd>your_admin_password</apswd>
      </s:request>
    </s:HTTPService>
  </fx:Declarations>
  <fx:Script>
    <![CDATA[
      private function myServiceResultHandler():void {
        outputBox.text = myService.lastResult.toString();
      }
      private function doButtonClick():void {
```

```
        /* Add the appName parameter to the request. */
        /* Examples include "vod" and "live". */
        myService.request.appName = appName.text;
        myService.send();
    }
    ]]>
</fx:Script>
<s:layout>
    <s:VerticalLayout paddingTop="10" paddingLeft="10"/>
</s:layout>
<s:Label text="Enter an app name (e.g., live or vod):"
    fontSize="14"
    fontWeight="bold"/>
<s:HGroup>
    <s:TextInput id="appName" width="259"/>
    <s:Button id="button_btn" label="Get App Stats" click="doButtonClick()"/>
</s:HGroup>
<s:Label text="App Stats:" fontSize="14" fontWeight="bold"/>
<s:TextArea id="outputBox" height="800" width="359"/>
</s:Application>
```

This application was written for Flex 4.

If the call is successful, the `getAppStats` method returns a result similar to the following:

```
<result>
  <level>status</level>
  <code>NetConnection.Call.Success</code>
  <timestamp>6/30/2010 2:42:21 PM</timestamp>
  <data>
    <launch_time>6/30/2010 2:42:03 PM</launch_time>
    <up_time>17</up_time>
    <bw_in>17060</bw_in>
    <bw_out>17501</bw_out>
    <bytes_in>290075</bytes_in>
    <bytes_out>271130</bytes_out>
    <msg_in>202</msg_in>
    <msg_out>61</msg_out>
    <msg_dropped>0</msg_dropped>
    <total_connects>2</total_connects>
    <total_disconnects>0</total_disconnects>
    <connected>2</connected>
    <normal_connects>2</normal_connects>
    <virtual_connects>0</virtual_connects>
    <group_connects>0</group_connects>
    <service_connects>0</service_connects>
    <service_requests>0</service_requests>
    <admin_connects>0</admin_connects>
```

```
<accepted>2</accepted>
<rejected>0</rejected>
<total_instances_loaded>1</total_instances_loaded>
<total_instances_unloaded>0</total_instances_unloaded>
<swf_verification_attempts>0</swf_verification_attempts>
<swf_verification_exceptions>0</swf_verification_exceptions>
<swf_verification_failures>0</swf_verification_failures>
<swf_verification_unsupported_rejects>0</swf_verification_unsupported_rejects>
<swf_verification_matches>0</swf_verification_matches>
<swf_verification_remote_misses>0</swf_verification_remote_misses>
<cores>
  <_0>
    <pid>15720</pid>
    <core_id>0</core_id>
  </_0>
</cores>
<server_bytes_in>0</server_bytes_in>
<server_bytes_out>0</server_bytes_out>
</data>
</result>
```


Chapter 7: XML configuration files reference

Edit the configuration files in the *RootInstall/conf* directory to configure the server.

Important: Do not modify certain XML elements without contacting Adobe Support. These elements are marked with an *Important note*.

More Help topics

“[Edit a configuration file](#)” on page 17

Changes to configuration files from 4.5 to 5.0.1

When upgrading from version 4.5 of Flash Media Server to version 5.0.1 of Adobe Media Server, be aware of changes to the XML configuration files.

The changes are primarily related to the introduction of Closed captioning support in Adobe Media Server 5.0.1.

Server.xml changes

- See Closed captioning VOD case - Configuring Server.xml (only for RTMP)
- See Closed captioning support for OSMF

Application.xml changes

- See Closed captioning Live case - Configuring Application.xml (HDS and RTMP)
- Protected RTMP (introduced in 4.5.1) change:

```
<!-- Protected RTMP parameters. Change 'enabled' tag to "true" to enable Protected RTMP for VOD -->
<ProtectedRTMP enabled="false">
  <!-- DRM Metadata update interval (minutes) -->
  <!-- This would renew the playback expiration time stamp
and SWF verification white list -->   <UpdateInterval>60</UpdateInterval>
  <SWFVerification enabled="false">
    <!-- By default, the whitelist folder is set to the application folder. -->
    <!-- If a relative path is specified, it will be relative to
the application folder. --><WhiteListFolder></WhiteListFolder>
  </SWFVerification>
</ProtectedRTMP>
```

Changes to configuration files from 4.0 to 4.5

When upgrading from version 4.0 to version 4.5 of Flash Media Server, be aware of changes to the XML configuration files.

ams.ini (fms.ini) changes

The configuration file fms.ini has been renamed to ams.ini.

The cache cleaning tool can be started when the HTTP server starts:

```
SERVER.HTCACHECLEAN_ENABLED = true
```

The cache root for the web server can be specified in this configuration file:

```
SERVER.HTCACHEROOT = /opt/adobe/ams/Apache2.2/cacheroot
```

The Flash Media Server 4.5 fms.ini file does not store the Administrator password. The following parameter has been removed from the 4.5 fms.ini file:

```
> # Password for server admin
> # For example:
> #     SERVER.ADMIN_PASSWORD = bar
> #
> SERVER.ADMIN_PASSWORD = adminadmin
```

The password is stored in the Users.xml file as a salted hash. To reset the password, see [“Delete administrator accounts and reset passwords”](#) on page 92.

Adaptor.xml file changes

The default value of MaxHeaderLineLength is now 8192.

There are two new elements: Adaptor/SSL/SSLVerifyClient and Adaptor/SSL/SSLVerifyDepth.

Two new tags FlowSendMaxBufferSize and FlowSendMaxUnsentAge have been introduced:

```
<RTMFP enable="true">
  <!-- Max value of the send buffer size. Rtmfp can auto-tune the send -->
  <!-- buffer size at runtime, but not greater than this value.      -->
  <!-- Default is 2MB. This value can not be set less than 127 KB    -->
  <FlowSendMaxBufferSize>2097152</FlowSendMaxBufferSize>
  <!-- Max unsent age of the send buffer allowed. Default is 8000 ms -->
  <FlowSendMaxUnsentAge>8000</FlowSendMaxUnsentAge>
</RTMFP>
```

Application.xml file changes

The [“PeerLookupEvents”](#) on page 172 element is new. Use this element to determine how peer lookup events are dispatched.

The element /StreamManager/EventsDir specifies the physical location where Event information for live streaming is stored.

The [RTMFP](#) section provides the means to control the behavior of application-specific RTMFP functionality. It includes the following elements: PeerLookupEvents, GroupControl, and JoinLeaveEvents. The [HDS](#) section provides the means to control the behavior of application-specific HTTP dynamic streaming functionality. It includes the following elements: Recording and ContentProtection.

Logger.xml file changes

There are no changes to the Logger.xml file.

Server.xml file changes

You can specify how the cacheclean tool can be started:

```
<!-- The command line option to start the cacheclean tool -->
<!-- $Directory/$Program $RequiredOptions $AdditionalOptions -->
<!-- (where $Directory can be either relative to $AMSROOT or -->
<!-- absolute; All mandatory options for the Program need to be -->
<!-- taken care inside RequiredOptions tag, and any additional -->
<!-- options need to be included in AdditionalOptions tag.) -->
<!-- The cache is enabled/cleaned by default on Linux only. -->
<HtCacheClean enabled="{SERVER.HTCACHECLEAN_ENABLED}">
  <Directory>Apache2.2/bin</Directory>
  <Program>htcacheclean</Program>
  <RequiredOptions>-p${SERVER.HTCACHEROOT} -d5 -l1024K</RequiredOptions>
  <AdditionalOptions>-n -t -i</AdditionalOptions>
</HtCacheClean>
```

The `/RTMP/ConnectionTimeout` element and `/HTTP/ConnectionTimeout` element specify how long to wait before timing out an outgoing connection, in milliseconds. The default is -1 which uses the OS timeout to block outgoing connections.

```
<ConnectionTimeout>-1</ConnectionTimeout>
```

Enable a [Registry](#) core to activate a single AMSCore instance dedicated to handling services for Adobe Media Gateway. This AMSCore process comes to life on start-up and allows Adobe Media Gateway to register its services to Adobe Media Server. Enable this registry core only when using Adobe Media Gateway.

```
<Registry enabled="false">
  <!-- Adaptor to run registry core on -->
  <AdaptorName>_defaultRoot_</AdaptorName>
</Registry>
```

Users.xml file changes

The Users.xml file contains a salted password hash for the Administrator Console password:

```
<Password>e206a5e1b52dcb4eaf024ca6adbe321b86cf0079bb747b78134ddaf8375e10aff905da0b28e84c5a</Password>
```

Vhost.xml file changes

No changes.

Adaptor.xml file

The Adaptor.xml file is the configuration file for individual network adaptors. It determines the number of threads that can be used by the adaptor, the communications ports the adaptor binds to, and the IP addresses or domains from which the adaptor can accept connections.

You can also implement SSL with the Adaptor.xml file, if you want to use different digital certificates for different adaptors.

Each adaptor has its own directory inside the server's `conf` directory. The name of the directory is the name of the adaptor. Each adaptor directory must contain an `Adaptor.xml` file. For example, the default adaptor included with the server at installation is named `_defaultRoot_` and its directory is found in the `conf/` directory. To change an adaptor's settings, edit the elements in its `Adaptor.xml` file.

To see the element structure and default values in the `Adaptor.xml` file, see the `RootInstall/conf/_defaultRoot_/Adaptor.xml` file.

Adaptor

Root element. Contains all the elements in the `Adaptor.xml` file.

Allow

A comma-delimited list of host names, domain names, and/or full or partial IP addresses from which clients can connect to the server.

Example

```
<Allow>foo.yourcompany.com, adobe.com, 10.60.1.133, 10.60</Allow>
```

See also

[Deny](#), [Order](#)

Deny

A comma-delimited list of host names, domain names, and/or full or partial IP addresses from which clients cannot connect to the server.

Example

```
<Deny>foo.yourcompany.com, adobe.com, 10.60.1.133, 10.60</Deny>
```

See also

[Allow](#), [Order](#)

Edge

Container element.

Contains elements that specify an edge to configure for HTTP tunnelling. Edges are defined in `HostPort` elements in the `HostPortList` container. Each edge can have different `HTTPTunnel` configurations.

Attribute

name A name identifying an edge. Use the name specified in the `HostPort` element.

Contained elements

[Enable](#), [IdlePostInterval](#), [IdleAckInterval](#), [MimeType](#), [WriteBufferSize](#), [SetCookie](#), [Redirect](#), [NeedClose](#),

Enable

Specifies whether or not to allow HTTP tunnelling connections.

These are the possible values for the `Enable` element:

Value	Description
<code>true</code>	Allow all HTTP tunnelling connections. This is the default value.
<code>false</code>	Disallow all HTTP tunnelling connections.
<code>http1.1only</code>	Allow only HTTP 1.1 tunnelling connections.
<code>keepalive</code>	Allow HTTP 1.1 or HTTP 1.0 keepalive connections.

Important: Only one application can use a port at a time. For example, if you configure Adobe Media Server to use port 80 for HTTP tunnelling, the web server cannot use port 80.

Example

```
<Enable>true</Enable>
```

See also

[IdlePostInterval](#), [IdleAckInterval](#), [MimeType](#), [WriteBufferSize](#), [SetCookie](#), [Redirect](#), [MimeType](#),

EnableAltVhost

Determines whether an alternative virtual host may be specified as a part of the RTMP URL as query parameter `rtmp://host1/app/?_fcs_vhost=host2`. This does not apply to administrative connections, which are always on by default. The default value is `false`.

```
<EnableAltVhost>false</EnableAltVhost>
```

See also

[HTTPTunnel](#), [RTMP](#), [SSL](#)

Host

Redirects unknown requests to an external server.

See also

[Redirect](#)

HostPort

Specifies to which IP address and port(s) an adaptor binds. If you want to bind an adaptor to multiple IP addresses, add a `HostPort` element for each IP address. The format is `IP:port, port, . . . , port`. To bind to any IP address, don't specify anything in front of the colon.

This element is exposed as the `ADAPTOR.HOSTPORT` parameter in the `RootInstall/conf/ams.ini` file. The default value is `:1935,80,-443`. This value instructs the adaptor to bind to any IP address on ports 1935, 80, and 443, where 443 is designated as a secure port that will only receive RTMPS connections.

When assigning port numbers, keep in mind the following:

- There is a risk in assigning more than one adaptor to listen on the same IP:port pair. If another process tries to bind to the same IP:port combination, a conflict results. To resolve this conflict, the first adaptor to bind to the specified HostPort wins. Adobe Media Server logs a warning in the Access log file indicating that the specified IP:port is in use.
- Only one application can use a port at a time. For example, if you configure Adobe Media Server to use port 80 for HTTP tunnelling, the web server cannot use port 80.

Attributes

name A name to identify this HostPort element. If there are multiple HostPort elements, each must have a different name.

ctl_channel The internal port that an AMSEdge (amsedge in Linux) process listens on. When an AMSCore process is started (amscore in Linux), it connects to an AMSEdge process on this internal port to establish a control channel. Each HostPort element corresponds to an AMSEdge process. If there are multiple HostPort elements, each must have a different ctl_channel.

Example

The following HostPort value instructs the adaptor to bind to the IP address 127.0.0.1 on ports 1935, 80, and 443, where 443 is designated as a secure port that will only receive RTMPS connections. (A port is marked as secure by specifying a minus sign in front of the port number.) RTMPS connection attempts to ports 1935 or 80 will not succeed. The client will attempt to perform an SSL handshake that the server will fail to complete. Similarly, a regular RTMP connection to port 443 fails because the server tries to perform an SSL handshake that the client fails to complete.

```
<HostPort>127.0.0.1:1935,80,-443</HostPort>
```

If there is no colon in the HostPort value, or there is a colon with no ports specified, the data is assumed to be an IP address and binds to port 1935. The following values instruct the adaptor to bind to IP 127.0.0.1 on port 1935:

```
<HostPort>127.0.0.1</HostPort>  
<HostPort>127.0.0.1:</HostPort>
```

See also

[HostPortList](#)

HostPortList

Container element.

The elements in this container list HostPort elements associated with this adaptor.

Example

```
<HostPortList>  
  <HostPort name="edge1" ctl_channel=":19350">${ADAPTOR.HOSTPORT}</HostPort>  
  <HostPort name="edge2" ctl_channel=":19351">:1936,-444</HostPort>  
</HostPortList>
```

Contained elements

[HostPort](#)

HTTPIdent

Configures the server to respond to or reject an HTTP identification request from a client. For a response to be returned, the `HTTPIdent` function must be enabled and the client must do a `POST` or `GET` for the `"/fcs/ident"` resource.

When the server receives an HTTP Identification request, it logs the request to the `edge.xx.log` file.

Attributes

enable A Boolean value specifying whether the server responds to an HTTP identification request (`true`) or not (`false`). The default value is `false`. When the `enable` attribute is set to `true`, all elements in the `HTTPIdent` section are returned as a response. The entire response is enclosed in `<FCS></FCS>` elements, which are added by the server. If the `HTTPIdent` function is enabled but no content is specified, the `<FCS></FCS>` response is returned without content.

Example

```
<HTTPIdent enable="true">
  <Company>Adobe System Inc</Company>
  <Team>Adobe Media Server</Team>
</HTTPIdent>
```

The following is an example request:

```
http://serverIP:1935/fcs/ident
```

The following is an example response:

```
<fcs>
  <Company>Adobe System Inc</Company>
  <Team>Adobe Media Server</Team>
</fcs>
```

See also

[HTTPNull](#), [HTTPTunnel](#), [HTTPUserInfo](#)

HTTPIdent2

Configures the server to respond to or reject a special HTTP request from the client before the client attempts to make a RTMPT connection to Adobe Media Server. For a response to be returned, the `HTTPIdent2` function must be enabled. This function is available when using Flash Player Update 3 or later.

RTMPT (Tunnelled Real-Time Messaging Protocol) can have difficulties working with load balancers. If a single RTMPT session consists of multiple socket connections, each connection may be sent to any one of many Adobe Media Servers behind load balancers. This causes the session to be split across machines. Using `<HTTPIdent2>` enables Flash Player to send a special HTTP request before connecting to Adobe Media Server.

Attributes

enabled A Boolean value specifying whether the server responds to a special HTTP identification request before making a RTMPT connection (`true`) or not (`false`). This feature is enabled by default, even if the `<HTTPIdent2>` tag or the `enabled` attribute is missing. The IP address can be explicitly configured and does not need to be the IP of the Adobe Media Server machine. (This allows RTMPT connections to be redirected to a different server.) If the tag is left empty, the IP address is determined automatically.

If you are running Adobe Media Server on Linux and have enabled IPv6 but are using an IPv4 hostname (a hostname that resolves to IPv4), then use this tag to resolve RTMPTE and RTMPE connections more quickly: either set the `enabled` attribute to `false` or set it to `true` and set the value of the tag to the IP address to which you're connecting.

Example

```
<HTTPIdent2 enabled="true">10.133.128.71</HTTPIdent2>
```

See also

[HTTPNull](#), [HTTPTunnel](#), [HTTPUserInfo](#)

HTTPNull

Configures the server to respond to or reject an HTTP GET request for the "/" resource from a client. When the `enable` attribute is set to `true`, an HTTP 404 response is sent in response to an HTTP GET request. By default, the `HTTPNull` function is disabled.

If `HTTPNull` is disabled and the webserver is enabled, requests for the "/" resource are passed to the webserver.

Attributes

enable A Boolean value. The default value is `false`.

Example

```
<HTTPNull enable="false">
```

HttpProxy

Specifies whether the server proxies unknown requests to an external server. For proxying to work, `HTTPTunnel/Enable` must be set to `true`.

Request proxying to a specific host can be further controlled based upon port on which the request arrived. Flow control kicks in when the buffer in either direction fills up, automatically handling the case when producers and consumers differ widely in bandwidth.

Attributes

enable A Boolean value. The default value is `true`.

maxbuf The size of the io buffers. The default value is 16384.

Example

In the default `Adaptor.xml` file, the value of the `Host` element is a variable from the `ams.ini` file. You can set the value of `HTTPPROXY.HOST` in the `ams.ini` file, or replace the variable in the `Adaptor.xml` file. The following is from the default `Adaptor.xml` file:

```
<HttpProxy enable="true" maxbuf="16384">  
  <Host port="80">${HTTPPROXY.HOST}</Host>  
</HttpProxy>
```

The IP address and port to which the server proxies unknown HTTP requests. `HTTPPROXY.HOST = webfarm.example.com:80`

If you do not specify an IP address or domain name, the server uses a localhost port, as in the following:

```
HTTPPROXY.HOST = :8134
```


HTTPTunnel

Container element.

The elements in this section configure the incoming HTTP tunnelling connections to the adaptor. You can use the contained `Edge` element to configure each edge individually.

Contained elements

[Enable](#), [NodeID](#), [IdlePostInterval](#), [IdleAckInterval](#), [IdleTimeout](#), [MimeType](#), [WriteBufferSize](#), [SetCookie](#), [Redirect](#), [Host](#), [HttpProxy](#), [NeedClose](#), [MaxHeaderLineLength](#), [Edge](#)

HTTPUserInfo

Container element.

The elements in this section configure the absolute path to XML files and the cache settings.

Contained elements

[Path](#), [MaxSize](#), [UpdateInterval](#)

IdleAckInterval

Specifies the maximum time in milliseconds the server waits before it sends an acknowledgement code for a client idle post. An acknowledgement code is a transmission control character used to indicate that a transmitted message was received uncorrupted and without errors, and that the receiving server is ready to accept transmissions. The default value is 512 milliseconds, which provides medium latency.

The values for this element and the `IdlePostInterval` element affect the latency observed by a client tunnelling into the server. These elements should be configured at the same time.

Lower values reduce latency but increase the network bandwidth overhead. Applications desiring low latency may configure the combination of values 128/256 for the `IdlePostInterval` and `IdleAckInterval` elements. For applications not sensitive to high latencies, use the combination 1024/2048.

```
<IdleAckInterval>512</IdleAckInterval>
```

See also

[IdlePostInterval](#), [IdleTimeout](#)

IdlePostInterval

Specifies the interval in milliseconds at which the client sends idle posts to the server to indicate that Flash Player has no data to send.

The default settings for the `IdleAckInterval` and `IdlePostInterval` elements provide medium latency and are set to 512/512 milliseconds.

Low values reduce the latency but increase the network bandwidth overhead. Applications desiring low latency may configure the combination of values 128/256 for `IdlePostInterval` and `IdleAckInterval` elements. Applications that do not have high latencies can use the configuration 1024/2048.

Example

```
<IdlePostInterval>512</IdlePostInterval>
```

See also

[IdleAckInterval](#), [IdleTimeout](#)

IdleTimeout

Specifies the maximum inactivity time, in seconds, for a tunnel session before it is closed.

When a client using HTTP tunnelling disconnects unexpectedly, their session may remain open for a long period of time after disconnecting. The value of `IdleTimeout` indicates the maximum time, in seconds, that such a session may remain idle before it will be automatically disconnected. An `IdleTimeout` of -1 indicates that idle tunnel sessions will not be disconnected. The default setting for `IdleTimeout` is 8 seconds. Values that are too low may cause clients with very high latencies to become disconnected. Values that are too high may result in disconnected sessions consuming server resources for longer than necessary.

Example

```
<IdleTimeout>8</IdleTimeout>
```

See also

[IdlePostInterval](#), [IdleAckInterval](#)

MaxFailures

Specifies the maximum number of failures an edge server may incur before it restarts.

The default number of failures is 2.

Example

```
<MaxFailures>2</MaxFailures>
```

See also

[IdleAckInterval](#)

MaxHeaderLineLength

The maximum size of an HTTP header line, in bytes, that the server can accept. The default value is 8192. To accept longer header lines, increase this value.

See also

[“Specify the maximum HTTP header line length”](#) on page 54

MaxPipelinedRequests

Configures the maximum number of requests Flash Player will allow to be pipelined. The value must be greater than or equal to 1.

The default value is 5.

Example

```
<MaxPipelinedRequests>6</MaxPipelinedRequests>
```

MaxSize

Specifies the maximum number of HTTP requests the server keeps in the cache. When the cache size reaches the maximum size, the server reduces the cache. By default, the value is 100.

Example

```
<MaxSize>100</MaxSize>
```

See also

[Path](#), [UpdateInterval](#)

MimeType

Specifies the default MIME (Multi-purpose Internet Mail Extensions) type header sent on tunnel responses.

The server generally uses the MIME type specified by the incoming requests. The server uses the entry for the `MIMETYPE` element only if it is unable to determine the MIME type from the incoming requests.

Example

```
<MimeType>application/x-fcs</Mimetype>
```

See also

[HTTPTunnel](#)

NeedClose

A Boolean value specifying whether or not HTTP 1.0 non-keepalive connections are closed once the response is written. The default value is `true`, which closes the connections.

Example

```
<NeedClose>true</NeedClose>
```

NodeID

Specifies a unique node identification that supports the implementation of load balancers.

If the `NodeID` element is used, a following string of up to nine characters is prefixed to the tunnel session IDs and can be used by the load balancers to uniquely identify each node in the cluster.

The ID must contain URL-safe characters: alphanumerical A-Z, a-z, and 0-9, and the special characters `$-_.+!*'()`

See also

[HTTPTunnel](#)

Order

Specifies the sequence in which the server evaluates the `Allow` and `Deny` elements. The following is the default sequence:

```
<Order>Allow,Deny</Order>
```

The default sequence indicates that access to a server is denied unless it is specified in the `Allow` element.

The alternative sequence `Deny, Allow` indicates that access to a server is allowed unless specified in the `Deny` element and not specified in the `Allow` element.

Example

The following is the default sequence:

```
<Order>Allow,Deny</Order>
```

See also

[Allow, Deny](#)

Path

Specifies the absolute path of the folder where the server looks for XML files. By default, it is set to `uInfo/` in the Adobe Media Server installation directory.

See also

[MaxSize](#), [UpdateInterval](#)

RecoveryTime

Specifies the number of seconds an edge server waits before restarting after a failure.

When an edge server fails, it waits for the interval specified here before it restarts. The wait time is specified in seconds.

The number of failures is specified by the `MaxFailures` element.

Example

```
<RecoveryTime>30</RecoveryTime>
```

See also

[MaxSize](#)

Redirect

Specifies whether or not the adaptor redirects unknown requests to an external server.

Note: For redirection to work, HTTP tunnelling must be enabled.

An unknown request may connect only when it is the first request on a newly accepted connection. At any other time, the request is considered an error and the connection is closed.

The `maxbuf` attribute determines how big the IO buffers are. Flow control automatically handles the request when the bandwidth resources for producers and consumers differ widely. Flow control begins when the buffer in either direction fills up.

Example

This example instructs the server to redirect unknown requests to the specified redirect host.

```
<Redirect enable="false" maxbuf="16384">  
  <Host port="80">:8080</Host>  
  <Host port="443">:8443</Host>  
</Redirect>
```

See also

[NeedClose](#)

ResourceLimits

Container element.

The elements in this container configure the resource limits for the edge server.

Contained elements

[MaxFailures](#), [RecoveryTime](#)

RTMP

Container element.

The elements in this container determine whether encrypted RTMP (RTMPE) and encrypted tunnelling RTMP (RTMPTE) can be used.

Contained elements

[RTMPE](#)

RTMPE

Specifies whether encrypted RTMP (RTMPE) can be used. RTMPE is the encrypted RTMP protocol covering both RTMPE and RTMPTE. This element is enabled by default; setting `enabled` to `false` will not allow RTMPE or RTMPTE on this adaptor.

Example

```
<RTMPE enabled="true"></RTMPE>
```

See also

[RTMP](#)

SetCookie

Specifies whether or not the server sets a cookie.

If the server does not have an externally visible IP address, then for HTTP tunnelling to work, you should enable cookies when you deploy servers behind a load balancer. The load balancer checks the cookie and sends requests with this cookie to the same server. Keep in mind that the cookie adds to the HTTP header size and increases the bandwidth overhead.

Note: For tunnelling connections, cookies are currently supported only on Flash Player 9.0.28 or later, in Windows only.

Example

```
<SetCookie>false</SetCookie>
```

See also

[NodeID](#)

SSL

Container element.

The elements in this section configure the incoming connections via the Secure Sockets Layer protocol, known as SSL. The SSL elements in Adaptor.xml configure the server to act as an SSL-enabled server to accept incoming SSL connections.

You need to acquire a digital certificate to use SSL. Once you get your SSL certificate through a certificate authority, such as Verisign, or by creating it yourself with a product such as OpenSSL, you then use the SSL elements to configure the server for SSL.

The following is a quick start to allowing SSL-enabled connections to the server:

- to the SSL section of the Adaptor.xml file.
- Specify the location of the certificate in the `SSLCertificateFile` element.
- Specify where to find the associated private key file in the `SSLCertificateKeyFile` element.
- If the private key file is encrypted, specify the passphrase to use for decrypting the private key file in the `SSLPassPhrase` element.
- Save the modified Adaptor.xml file.

Contained elements

[SSLServerCtx](#)

SSLCertificateFile

Specifies the location of the certificate to return to clients who want to make a secure connection to the server.

If an absolute path is not specified, the certificate location is assumed to be relative to the adaptor directory.

Example

```
<SSLCertificateFile>c:\myCertFile</SSLCertificateFile>
```

See also

[SSLCertificateKeyFile](#), [SSLPassPhrase](#), [SSLCipherSuite](#), [SSLSessionTimeout](#)

SSLCertificateKeyFile

This specifies the location of the private key file that corresponds to the public key in the certificate specified in `SSLCertificateFile` element.

If this file is encrypted, a password must be specified for decrypting and placed in the `SSLPassPhrase` element described below. If an absolute path to the key file is not specified, it is assumed to be relative to the adaptor directory.

Example

```
<SSLCertificateKeyFile type="PEM"></SSLCertificateKeyFile>
```

The `type` attribute specifies the type of encoding used for the certificate key file. The encryption format is either PEM (Privacy Enhanced Mail) or ASN1 (Abstract Syntax Notation 1). The default is PEM.

See also

[SSLCertificateFile](#), [SSLPassPhrase](#), [SSLCipherSuite](#), [SSLSessionTimeout](#)

SSLCipherSuite

Specifies the suite of encryption ciphers that the server uses to secure incoming connections.

This element contains a list of colon-delimited components. A component can be a key exchange algorithm, authentication method, encryption method, digest type, or one of a selected number of aliases for common groupings.

Note: Contact Adobe Support before changing the default settings as listed in this example.

Example

```
<SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
```

See also

[SSLCertificateFile](#), [SSLCertificateKeyFile](#), [SSLPassPhrase](#), [SSLSessionTimeout](#)

SSLPassPhrase

Specifies the passphrase to use for encrypting the private key file.

Specifies the password to use for decrypting the key file if the key file is encrypted. If the key file is not encrypted, this element is left blank.

To prevent plain text passwords from appearing in the configuration file, encode the password in base64 format and set the `encrypt` attribute to `true`.

Example

```
<SSLPassPhrase encrypt="true">dGluY2Fu</SSLPassPhrase>
```

The encrypted password is equivalent to the plain text format:

```
<SSLPassPhrase>tincan</SSLPassPhrase>
```

or

```
<SSLPassPhrase encrypt="false" >tincan</SSLPassPhrase>
```

Even though the element attribute is named "encrypt", it is not a true encryption. It is a base64 encoding that makes the password less readable.

See also

[SSLCertificateFile](#), [SSLCertificateKeyFile](#), [SSLCipherSuite](#), [SSLSessionTimeout](#)

SSLServerCtx

Container element.

The elements in this section control the incoming SSL configuration for this adaptor.

Contained elements

[SSLCertificateFile](#), [SSLCertificateKeyFile](#), [SSLPassPhrase](#), [SSLCipherSuite](#), [SSLSessionTimeout](#), [SSLVerifyClient](#), [SSLVerifyDepth](#)

SSLSessionTimeout

Specifies in minutes how long a SSL-based session remains valid. The default time period is 5 minutes.

SSL sessions are used to improve performance by avoiding the need to perform the full SSL handshake for every connection. When a client connects to a server for the first time, it must perform the full handshake. After that first handshake, the server sends back a session object to the client, which the client can place in the cache and reuse at a later time.

If the client connects to the same server again at a later time, it can send back the cached session object. The server does not require the full SSL handshake if the session is still valid.

Example

```
<SSLSessionTimeout>5</SSLTimeout>
```

See also

[SSLCertificateFile](#), [SSLCertificateKeyFile](#), [SSLPassPhrase](#), [SSLCipherSuite](#)

SSLVerifyClient

Sets the certificate verification level for client authentication. The following are valid levels:

- none
Server does not send request for client certificate.
- optional
The client may send a certificate. If sent, it is checked. If certificate verification fails, TLS/SSL handshake is immediately terminated.
- required
Server sends a request for a client certificate. If the client does not send one, TLS/SSL handshake is immediately terminated.
- optional_no_ca
Server sends a request for a client cert. But the certificate need not be successfully verifiable. This should not be used in a production deployment, but can be useful for testing/debugging purposes.

The default value is none. The `log` attribute specifies whether or not to log when client certificate verification passes or fails.

Example

```
<SSLVerifyClient log="false">none</SSLVerifyClient>
```

SSLVerifyDepth

The maximum depth of CA certificates allowed in client certificate verification. The value 0 indicates that only self-signed certificates are accepted. The default value is 9.

UpdateInterval

Specifies how often the server refreshes the cache content, in milliseconds. By default, the value is 5000 milliseconds.

Example

```
<UpdateInterval>5000</UpdateInterval>
```


See also

[Path](#), [MaxSize](#)

WriteBufferSize

Specifies the size of the write buffer in kilobytes. The default size is 16 KB.

Example

```
<WriteBufferSize>16</WriteBufferSize>
```

See also

[ResourceLimits](#)

Application.xml file

The Application.xml file contains the settings for Adobe Media Server applications.

The Application.xml file in the virtual host directory configures the default settings for all applications within the virtual host. If you want to have different settings for a particular application, copy an Application.xml file to the application's registered application directory (`/applications/app_name`) and edit it to include the settings you want.

In most cases, the settings in the Application.xml file in the application directory override settings in the Application.xml file in the virtual host directory, but not always. You can add an `override` attribute to certain elements in a virtual host's Application.xml file, as in the following:

```
<LoadOnStartup override="no">false</LoadOnStartup>
```

The server uses the following rules when applying the `override` attribute:

- When the `override` attribute is included in an element and set to `no`, application-specific Application.xml files cannot override that element's setting.
- If an element has the `override` attribute set to `no`, then all subelements also cannot be overridden.
- If an Application.xml file is located in the application directory and specifies a different value than the default for an element that cannot be overridden, it is ignored, and the default is used.
- If the default Application.xml file is missing or invalid, the server will not start.
- If the user-specified Application.xml configuration file is invalid, it is ignored.
- All subelements under the `LoadOnStartup` element cannot be overridden.
- If you omit the `override` attribute, the `LoadOnStartup` element can be overridden.
- To change an element so it cannot be overridden, set the `override` tag to `no` in the uppermost tag.

Note: By default, the `Bandwidth` and `BandwidthCap` container elements include an `override` parameter set to `yes`, which allows the values for the `ClientToServer` and `ServerToClient` elements nested in these sections to be overridden. The `Client` element in this XML file includes an `override="no"` attribute by default.

To see the element structure and default values, see the Application.xml file installed with Adobe Media Server in the `RootInstall/conf/_defaultRoot/_defaultVhost_` directory.

Access

Container element.

The Access plug-in consists of the libconnect.dll file. It intercepts and examines each connection request to Adobe Media Server to determine whether the connection should be accepted or rejected.

Contained elements

[FolderAccess](#)

Access (Logging)

Container element.

Contains elements that configure logging in the Access log.

Contained elements

[Checkpoints](#)

AccumulatedIFrames

Container element.

The elements in this section specify the maximum size and duration of intermediate frames a live stream can hold in the buffer.

Contained elements

[MaxTime, MaxSize](#) ([AccumulatedIFrames](#))

AdaptorId

Availability

Flash Media Server 4

Description

Important: Do not change the value of this element.

Specifies an identifier for an adaptor to the server-side script engine. The default script adaptor is `<AdaptorId>as1</AdaptorId>`.

The `ScriptAdaptors` element in the `Server.xml` configuration file contains an element for each adaptor. The default adaptor is `as1`:

```
<ScriptAdaptors>
  <as1 lib="libasc"/>
</ScriptAdaptors>
```

See also

[ScriptEngine](#)

AggregateMessages (Client)

Specifies whether or not to send aggregate messages to clients. When the `enabled` attribute is set to `true`, the server will deliver aggregate messages to clients that support them. When this setting is disabled, aggregate messages are broken up into individual messages before being delivered to clients. The default is `false`.

AggregateMessages (Queue)

Container element; contains an element that control the size of the aggregate messages.

This element also specifies, when queuing is enabled, whether messages in the queue can be combined to form aggregate messages. When the `enabled` attribute is set to `true` (the default value), the server creates aggregate messages.

The server attempts to send aggregate messages to supported clients whenever possible. When this setting is disabled, aggregate messages are always broken up into individual messages before being delivered to clients.

Example

```
<AggregateMessages enabled="false"><\AggregateMessages>
```

See also

[MaxAggMsgSize](#), [HTTPTunnel](#), [MaxMessageSizeLosslessVideo](#), [OutChunkSize](#)

Allow

Specifies whether or not to allow the "following and Location:" header that is sent with redirection of an HTTP header. The default is `true`, allowing HTTP redirects.

Example

```
<Allow>true</Allow>
```

See also

[Max](#), [UnrestrictedAuth](#)

AllowDebugDefault

Specifies the default value for `application.allowDebug`. This is an opening that allows debug connections on a per application basis. The default value is `false`.

Example

```
<AllowDebugDefault>false</AllowDebug Default>
```

See also

[MaxPendingDebugConnections](#)

AllowHTTPTunnel

This element configures the server to allow HTTP tunnelling connections into this application.

By default, Flash Player communicates with the server using the RTMP protocol over port 1935. If that fails, it will try again over ports 443 and 80 in an attempt to get around firewall settings, which prevents TCP/IP connections over nonstandard ports.

In some cases, Flash Player has to negotiate a connection to Adobe Media Server through an edge server, or use the HTTP protocol to transmit RTMP packets (called *HTTP tunnelling*) if there is a firewall that allows only HTTP content to be sent out to public servers.

The values for this element are described in the following table.

Value	Description
true	Allows tunnelling connections.
false	Disallows tunnelling connections.
http1.1only	Allows HTTP 1.1 connections only.
keepalive	Allows HTTP 1.0 and 1.1 keepalive connections.

Example

```
<AllowHTTPTunnel>true</AllowHTTPTunnel>
```

See also

[Allow](#)

Application

This is the root element for Application.xml.

See also

[Process](#), [LoadOnStartup](#), [MaxAppIdleTime](#), [JSEngine](#), [StreamManager](#), [SharedObjManager](#), [AllowHTTPTunnel](#), [Client](#), [Debug](#), [HTTP](#), [SWFVerification](#)

AssumeAbsoluteTime

Determines whether an incoming live stream has timestamps that are based on an absolute clock. An example of an absolute clock is an SMPTE time signal that is contained within the encoder's input source.

The default value is false.

Example

```
<AssumeAbsoluteTime>true</AssumeAbsoluteTime>
```

Availability

Flash Media Server 4

Audio

Container element.

The elements in this section specify the settings for audio streams on the server.

Contained elements

[CombineSamples](#), [SendSilence](#)

AutoCloseIdleClients

Container element.

Contains elements that determine whether or not to close idle clients automatically.

Set the `enable` attribute to `true` to close idle clients. If the `enable` attribute is omitted or set to `false`, the feature is disabled. The default value is `false`.

A client is active when it is sending or receiving data. Use the `AutoCloseIdleClients` element to specify how often the server should check for idle clients. When a client has been idle longer than the maximum idle time (60 seconds by default), the server sends a status message to the `NetConnectionobject` (the client). The server closes the client connection to the server and writes a message to the access log. The server also writes a message such as “Client *x* has been idle for *y* seconds” in the core and event logs.

To configure the closing of idle connections, you must enable the feature in the `Server.xml` file. Once you enable the feature in the `Server.xml` file, you can disable the feature for individual virtual hosts in the `Vhost.xml` files or for individual applications in `Application.xml`. The values defined in the `Vhost.xml` configuration file apply to all clients connected to the `Vhost`, unless values are defined in the `Application.xml` file. The `Application.xml` values override the `Vhost.xml` values. (Subsequently, the values defined in the `Server.xml` configuration file apply to all clients connected to the server, unless the values are defined in the `Vhost.xml` file. The `Vhost.xml` values override the `Server.xml` values.

Example

```
<AutoCloseIdleClients enable="false">
  <CheckInterval>60</CheckInterval>
  <MaxIdleTime>600</MaxIdleTime>
</AutoCloseIdleClients>
```

AudioSampleAccess

Allows the client application to access the raw uncompressed audio data in a stream. By default, this element is disabled. To enable it, set the `enable` attribute to `true`. In the tag, specify a list of semicolon-delimited folders to which client applications have access. When this element is enabled, all clients can access the audio data in streams in the specified folders. To enable access to all audio data streamed by the server, specify `/` in the tag.

The folder path is restricted to the application’s streams folder or folders, so do not use absolute paths in the list of folders.

While you can also enable access through Server-Side ActionScript, this element allows access to the data without requiring Server-Side ActionScript. You can also override this element with the `Access` plug-in or Server-Side ActionScript.

Example

If an application is configured to store streams in folders `C:\low_quality` and `C:\high_quality`, the configuration to allow access to sample those streams is as follows:

```
<AudioSampleAccess enabled="true">low_quality;high_quality</AudioSampleAccess>
```

See also

[VideoSampleAccess](#)

AutoCommit

Specifies whether shared objects are automatically committed when they have been changed. Setting this element to `false` disables Flash Player function for all shared objects within this instance.

Note: If the `AutoCommit` function is disabled, the server-side script has to call the `save` function or the `SharedObject.commit` command for the shared object to persist; otherwise, all data will be lost when the application is shut down.

Example

```
<AutoCommit>true</AutoCommit>
```

See also

[StorageDir](#), [DuplicateDir](#), [ResyncDepth](#), [LockTimeout](#), [MaxProperties](#), [MaxPropertySize](#)

Bandwidth

Container element.

The elements nested in this container specify the bandwidth settings for upstream (client-to-server) and downstream (server-to-client) data.

By default, the `Bandwidth` element includes an `override` parameter set to `yes`, which allows the values for the `ClientToServer` and `ServerToClient` elements to be overridden as well.

Contained elements

[ClientToServer \(Bandwidth\)](#), [ServerToClient \(Bandwidth\)](#)

BandwidthCap

Container element.

The elements in this section specify the bandwidth settings that a user can set. By default, this element includes an `override` parameter set to `yes`, which allows the values for the `ClientToServer` and `ServerToClient` elements nested in this section to be overridden, too.

Contained elements

[ClientToServer \(BandwidthCap\)](#), [ServerToClient \(BandwidthCap\)](#)

BandwidthDetection

Container element.

This element contains settings for how the server detects bandwidth. Set the `enable` attribute to `true` or `false` to turn this feature on or off.

The server can detect client bandwidth in the core server code (native) or in a server-side script (script-based). Native bandwidth detection is enabled by default and is faster than script-based because the core server code is written in C and C++.

The server detects bandwidth by sending a series of data chunks to the client, each larger than the last. You can configure the size of the data chunks, the rate at which they are sent, and the amount of time the server sends data to the client.

The following table lists the values available for the `BandwidthDetection` element.

Element	Description
<code>BandwidthDetection</code>	Set the <code>enabled</code> attribute to <code>true</code> or <code>false</code> to turn this feature on or off.
<code>MaxRate</code>	The maximum rate in Kbps at which the server sends data to the client. The default value is -1, which sends the data at whatever rate is necessary to measure bandwidth.
<code>DataSize</code>	The amount of data in bytes that the server sends to the client. To detect the client's bandwidth, the server attempts to send a series of random blocks of data to the client, each time sending this much more data. For example, x bytes are sent, followed by 2x bytes, followed by 3x bytes, and so on until <code>MaxWait</code> time has elapsed.
<code>MaxWait</code>	The number of seconds the server sends data to the client. Increasing this number provides a more accurate bandwidth figure but also forces the client to wait longer.

Example

```
<BandwidthDetection enabled="true">
  <MaxRate>-1</MaxRate>
  <DataSize>16384</DataSize>
  <MaxWait>2</MaxWait>
</BandwidthDetection>
```

Contained elements

[MaxRate](#), [DataSize](#), [MinBufferTime](#) (Live)

Bits

This element contains the settings for Flash Player on the Windows and Macintosh platforms.

Example

```
<Bits from="WIN 6,0,0,0" to="WIN 7,0,55,0">0x01</Bits>
<Bits from="MAC 6,0,0,0" to="MAC 7,0,55,0">0x01</Bits>
```

See also

[UserAgent](#)

BufferRatio

Specifies the ratio of the buffer length used by server-side stream to live buffer.

Example

```
<BufferRatio>0.5</BufferRatio>
```

See also

[Server](#)

Cache

Container element; contains elements that configure the cache setting for SWF verification.

See also

[TTL](#), [UpdateInterval](#)

CachePrefix

Specifies the cache prefix that is passed from the origin server to the edge server.

This element is set on the origin server. The edge server uses the value of this element as a relative path to locate the cache file defined in the `CacheDir` element.

The `type` attribute provides additional specification for the cache prefix. The `type` attribute can be set to `path` or `sname`. The default is `path`.

Examples

```
<CachePrefix type="path"></CachePrefix>
```

When the attribute `type` is `path`, the server appends the physical path of the recorded stream to the prefix.

```
<CachePrefix type="sname"></CachePrefix>
```

When the attribute `type` is `sname`, the server appends the stream name to the prefix.

The cache prefix is any text with or without preset parameters. The prefix can be any name without special characters, such as `\`, `:`, `*`, `?`, `"`, `<`, `>`, `|`. All parameters are surrounded by `"?"`. The server substitutes the actual names for everything specified within the `"?"`.

By default, the prefix is set to `?IP?`.

Cache prefix	Actual name
?IP?	IP address of the server
?APP?	Application name
?APPINST?	Application instance
?VHOST?	vhost name

You can include the IP address in the prefix to avoid file collision. For example, the edge server might be connecting to two different origin servers with the same file in `c:\data\foo.flv`. Adding the IP to the prefix for these files points each file to the appropriate server.

If you want more than one origin server to share the cache file, do not include the IP as a parameter. Remember the cache prefix is a relative path used by the edge server to look up the cache stream file.

Examples

The cache prefix creates a relative path in the edge's `CacheDir`. All parameters are separated by `\` or `/`.

```
<CachePrefix type="path">c:\ams\flvs\foo.flv. data/?IP?</CacheDir>
```

resolves to:

```
data/xxx.xxx.xxx.xxx/c/ams/flvs/foo.flv  
<CachePrefix type="path">?APPINST?/data</CacheDir>
```


resolves to:

```
app1/inst1/data/c/ams/flvs/foo.flv  
<CachePrefix type="path">origin1/data/</CacheDir>
```

resolves to:

```
origin1/data/c/ams/flvs/foo.flv
```

See also

[StorageDir](#), [DuplicateDir](#), [CacheUpdateInterval](#)

CacheUpdateInterval

This element defines the wait interval for updating cache streaming in the edge server. The interval is defined in milliseconds. The default value is 10 minutes. The minimum interval is 10 seconds. The maximum interval is 24 hours.

Example

```
<CacheUpdateInterval>10</CacheUpdateInterval>
```

See also

[StorageDir](#), [DuplicateDir](#), [CachePrefix](#)

Checkpoints

Many companies use statistics from the access log to bill customers. If your programming includes live events or 24/7 programming, the events you use to calculate billing might not occur within a billing cycle. To solve this problem, you can enable *checkpoint events*. Checkpoint events log bytes periodically during an event. The following are available as checkpoint events: `connect-continue`, `play-continue`, and `publish-continue`.

Logging checkpoint events to the Access log is enabled in the `Vhost.xml` file by default. You can disable logging checkpoints for this application, or change the checkpoint interval of this application. If the checkpoint interval is not specified in the `Application.xml` file, or if the value is invalid, the server uses the value set in the `Vhost.xml` file.

Contained elements

[LogInterval](#)

Client

Container element.

The elements nested within this container configure the client.

By default, the `Client` element includes an `override="no"` parameter. Individual applications cannot override how the elements in the `Client` section are configured.

Contained elements

[Bandwidth](#), [BandwidthCap](#), [BandwidthDetection](#), [MsgQueue](#), [HTTPTunnel](#), [MaxMessageSizeLosslessVideo](#), [OutChunkSize](#)

Client (RPC)

Lets you list methods allowed for client RPCs. If a method is not listed then it cannot be called.

To add methods, use a comma-delimited list of method names in the `<Method><Allow>` sub-elements.

The default is to not allow any methods for client RPCs.

Contained elements

Method

Example

```
<Client>
  <Method>
    <Allow></Allow>
  </Method>
</Client>
```

See also

[NetConnection \(RPC\)](#), [Stream](#), [SharedObject](#)

Availability

Flash Media Server 4

ClientToServer (Bandwidth)

Specifies the bandwidth the client can use for sending data upstream to the server. The default bandwidth is 1,250,000 bytes per second.

You can configure this value in the `Bandwidth` section of XML and in the `BandwidthCap` section of XML. The values in the `Bandwidth` section can be overridden, but the values in the `BandwidthCap` section are not. This allows ISPs to host applications for customers and ensure that no customer abuses the bandwidth limit. For example, a customer can set any bandwidth limit for their applications, but cannot exceed the caps set by the ISP.

Example

```
<ClientToServer>1250000</ClientToServer>
```

See also

[ServerToClient \(Bandwidth\)](#)

ClientToServer (BandwidthCap)

Specifies the maximum bandwidth a client can send to the server. The default bandwidth is 100,000,000 bytes per second.

You can configure this value in the `Bandwidth` section of XML and in the `BandwidthCap` section of XML. The values in the `Bandwidth` section can be overridden, but the values in the `BandwidthCap` section are not. This allows ISPs to host applications for customers and ensure that no customer abuses the bandwidth limit. For example, a customer can set any bandwidth limit for their applications, but cannot exceed the caps set by the ISP.

Example

```
<ClientToServer>100000000</ClientToServer>
```

See also

[ServerToClient \(BandwidthCap\)](#)

CombineSamples

Container element.

The server conserves system resources by combining sound samples. This strategy saves the CPU and bandwidth overhead when transmitting individual audio packets only.

Note: Use this strategy of combining sound samples advisedly during periods of high CPU usage, as it can cause latency.

Contained elements

[LoCPU](#), [HiCPU](#), [MaxSamples](#), [Subscribers](#)

ContentProtection

The enabled attribute can be set to "true", "false" or "allow". Content protection is enabled when the attribute is set to "true", and disabled when set to "false". If enabled is set to "allow", settings in the Event.xml ContentProtection section can override the ContentProtection section in the Application.xml file.

```
<HDS>
  <!-- This section controls the behavior of HTTP live recording -->
  <Recording>
    <ContentProtection enabled="allow">
      </ContentProtection>
    </Recording>
  </HDS>
```

See also

[Configuring content protection](#)

Connections

Container element.

The elements in this section configure the HTTP connections for this application.

Contained elements

[MaxTimeout](#) ([Connections](#)), [Reuse](#), [Interface](#)

DataSize

Specifies the amount of data in bytes that the server sends to the client. To detect the client's bandwidth, the server attempts to send a series of random blocks of data to the client, each time sending this much more data. For example, x bytes are sent, followed by 2x bytes, followed by 3x bytes, and so on until MaxWait time has elapsed.

Example

```
<DataSize>16384</DataSize>
```

See also

[MaxRate](#), [MinBufferTime](#) ([Live](#))

Debug

Container element.

The elements in this section configure debug connections, including the maximum number of connections and the value for `application.allowDebug`.

Contained elements

[AllowDebugDefault](#), [MaxPendingDebugConnections](#)

Diagnostic

Specifies whether diagnostic logging for the message queue is enabled. The default value is false.

See also

[Enabled](#)

DirLevelSWFScan

Specifies the number of levels of subfolders within a parent folder to scan for SWF files. The parent folder is specified in the `SWFFolder` element.

Specifying a positive value scans that number of subfolder levels. Specifying zero scans the parent folder and no subfolders. Specifying a negative value scans all subfolder levels. The default value is 1, which means that the server scans only one subfolder level.

See also

[SWFFolder](#), [SWFVerification](#)

DisallowedProtocols

Specifies which protocols cannot be used to connect to an application. Specify protocols in a comma delimited list. Any protocol not specified is allowed. For example, the following disallows rtmp and rtmps connections:

```
<DisallowedProtocols>rtmp,rtmps</DisallowedProtocols>
```

The default value is blank, which allows all protocols to connect to the server.

Distribute

Specifies how to distribute application instances to processes. The default value is `insts`, meaning each application instance runs in its own process. This tag contains a `numprocs` attribute, which specifies the maximum number of processes to run concurrently. The default value of the `numprocs` attribute is 3.

This feature is turned on by default. To use this feature, the `numprocs` attribute must be set to a value higher than 0 or 1. With the default configuration, for all your applications and application instances under a single virtual host, three core processes will run. Each virtual host is allotted three core processes, so systems that use multiple virtual hosts will generate more running processes.

Note: *There is no limit to the value of the `numprocs` attribute, but you should never need more than 40.*

Scopes have an enclosing relationship with a strict ordering: adaptors contain virtual hosts, which contain applications, which contain instances, which contain clients. The value of the `Distribute` tag must be a scope that is lower in order than the value in the `Scope` tag. In other words, if the value of `Scope` is `adaptor`, the value of `Distribute` can be `vhosts`, `apps`, `insts`, or `clients`. If the value of `Scope` is `app`, the value of `Distribute` can be `insts` or `clients`. By default, the server uses the value immediately lower than the one specified in the `Scope` tag.

The following table lists the values available for the `Distribute` element:

Value	Description
<code>vhosts</code>	All instances of applications in a virtual host run together in a process.
<code>apps</code>	All instances of an application run together in a process.
<code>insts</code>	Each application instance runs in its own process. This is the default value. If you choose this value, you must also set the <code>Distribute numprocs</code> attribute to a value greater than 1.
<code>clients</code>	Each client connection runs in its own process. Use this value for stateless applications—applications that don't require clients to interact with other clients and don't have clients accessing live streams. Most vod (video on demand) applications are stateless because each client plays content independently of all other clients. Chat and gaming applications are not stateless because all clients share the application state. For example, if a shared chat application were set to <code>client</code> , the messages wouldn't reach everyone in the chat because they'd be split into separate processes.

Example

```
<Distribute numproc="1"></Distribute>
```

See also

[Scope](#), [LifeTime](#), [MaxFailures](#), [RecoveryTime](#)

DuplicateDir

Note: This feature has been deprecated.

This is one of two `DuplicateDir` elements in the `Application.xml` file: one is in the [SharedObjManager](#) container and one is in the [StreamManager](#) container.

Specifies the physical location where duplicate copies of shared objects or recorded streams are stored.

This location serves as a backup for shared object files and recorded stream files. This location must already exist when a shared object is copied to it.

Example

To include the application name in the paths for the backup files, change the `appName` attribute to `"true"`.

```
<DuplicateDir appName="true">c:\backupSharedObjects</DuplicateDir>  
<DuplicateDir appName="true">c:\backupStreams</DuplicateDir>
```

See also

[StorageDir](#)

Duration

This element instructs the server how long, in seconds, to wait before it notifies the client when the audio has stopped in the middle of a live or recorded audio stream.

The default wait time is 3 seconds. The minimum wait time is 1 second. There is effectively no maximum value (the maximum is the maximum value of a 32-bit integer).

Example

```
<Duration>3</Duration>
```

See also

[Password](#)

Enabled

Specifies whether diagnostic logging for the message queue is enabled. The default value is false.

See also

[Diagnostic](#)

EnhancedSeek

This element enables or disables fine-tuning of the seeking performance within streams by creating a keyframe.

Keyframes improve the visual display of video files while seeking. When set to `true`, a new keyframe is dynamically generated to provide smooth seeking to that index point.

***Note:** The server generates new keyframes for Sorenson Spark-encoded FLV files. For On2 VP6, the new keyframe is calculated and generated in Flash Player 9a or later. For H.264-encoded video, the new keyframe is calculated and generated in Flash Player 9 Update 3 or later.*

The default value is `true`. The server does not insert keyframes and all seeks begin at the nearest existing keyframe.

Example

```
<EnhancedSeek>true</EnhancedSeek>
```

See also

[KeyFrameInterval](#)

EraseOnPublish

When set to `true`, the server erases a recorded stream when a publisher starts publishing a live stream with the same name. By default, this setting is disabled and the server does not erase a recorded stream.

See also

[Live \(StreamManager\)](#)

EventsDir

The element `/StreamManager/EventsDir` specifies the physical location where Event information for live streaming is stored, as in the following example:

```
<EventsDir>c:\myapp\events</EventsDir>
```

See also

[“StreamManager” on page 182](#)

Exception

This element indicates that a specific user agent is an exception to authentication. Use the `from` and `to` attributes to indicate the lowest and highest versions to except. This is a string comparison with editing to make all numeric fields equal length.

For example, using a specific Flash Player will report WIN 9,0,28,0 as its UserAgent. Add `To="WIN 9,0,28,0"` and `From="WIN 9,0,28,0"` and only that version is an exception.

See also

[UserAgentExceptions](#)

FileObject

Container element.

As of Flash Media Server 4.0, the `FileObject` element has an `override` attribute set to "no", by default. When `override="no"`, the application-level `Application.xml` file cannot override the `VirtualDirectory` settings for the `FileObject` defined in a Vhost-level `Application.xml` file.

The `VirtualDirectory` element nested within this container configures the `ScriptEngine` file object settings.

Contained elements

[VirtualDirectory](#)

FinalHashTimeout

This element defines the maximum amount of time that a client can use to provide its final verification to the server.

See also

[SWFVerification](#)

FirstHashTimeout

This element defines the maximum amount of time that a client can use to provide its first verification to the server.

See also

[SWFVerification](#)

FlushOnData

Specifies whether the server flushes the message queue when a data message arrives. This element is important for streaming data-only messages, so the server can send out the messages immediately. The default is `true`.

See also

[Queue](#), [MaxQueueSize](#), [MaxQueueDelay](#), [AccumulatedIFrames](#)

FolderAccess

Configures the level of permission that the `Access` plug-in can set for accessing streams and shared objects. This allows two levels of permissions: file-level access (a value of `false`), which allow access to a particular file only, and folder-level access (a value of `true`), which allows access to a particular directory.

Example

```
<FolderAccess>false</FolderAccess>
```

See also

[Access](#)

GroupControl

Settings within the GroupControl section control server-side functionality corresponding to the server channel. A server channel is a channel that clients within a Flash Group can open to the server. The presence of a server channel depends on `serverChannelEnabled=true` for the GroupSpecifier used to generate the `groupspec` value to join a Flash Group.

See also

[JoinLeaveEvents](#)

HDS

Container for configurations for HTTP Dynamic Streaming.

Contained Elements

[“Recording \(HDS\)” on page 175](#)

[“ContentProtection” on page 151](#)

HiCPU

This element instructs the server to start combining samples when CPU utilization is higher than the specified percentage of CPU resources. The default percentage of utilization is 80.

Example

```
<HiCPU>80</HiCPU>
```

See also

[Subscribers](#), [LoCPU](#), [MaxSamples](#)

Host

This element identifies the HTTP proxy. The value of the `Host` element can be the host name or an IP address. The port number can also be specified in the `Port` element.

Example

```
<Host>www.example.com:8080</Host>
```

See also

[Port](#), [Type](#), [Tunnel](#), [Username](#), [Password](#)

HTTP

Container element.

The elements in this section configure the HTTP connection settings for this application.

Contained elements

[HTTP1_0](#), [Verbose](#), [Connections](#), [Proxy](#), [Redirect](#)

HTTP1_0

This element determines whether or not the server can use the HTTP 1.0 protocol. The default is `false`, disallowing the use of the HTTP 1.0 protocol.

Example

```
<HTTP1_0>false</HTTP1_0>
```

See also

[HTTP](#), [Verbose](#), [Connections](#), [Proxy](#), [Redirect](#)

HTTPTunnel

Container element.

The elements nested within this container configure the parameters for HTTP tunnelling (sending RTMP packets through HTTP).

The tunnelling protocol is based on the client continuously polling the server. The frequency of polling affects both network performance and the efficiency of the HTTP protocol. The `IdleAckInterval` and `IdlePostInterval` elements control the polling frequency on a per-client basis. Selecting too small a delay value for the above parameters will increase the polling frequency and reduce network performance and efficiency. Selecting too high values can adversely affect the interactivity of the application and the server.

The `Application.xml` configuration file offers three settings for these parameters. The following table presents these settings.

Acceptable Latency	IdlePostInterval	IdleAckInterval
Low	128 milliseconds	256 milliseconds
Medium	512 milliseconds	512 milliseconds
High	1024 milliseconds	2048 milliseconds

Example

```
<HTTPTunnel>  
  <IdlePostInterval>512</IdlePostInterval>  
  <IdleAckInterval>512</IdleAckInterval>  
  <MimeType>application/x-fcs</MimeType>  
  <WriteBufferSize>16</WritebufferSize>  
</HTTPTunnel>
```

Contained elements

[IdlePostInterval](#), [IdleAckInterval](#), [MimeType](#), [WriteBufferSize](#)

IdleAckInterval

Specifies the maximum time the server may wait before it sends back an `ack` (acknowledgement code) for an idle post sent by the client.

The server may respond sooner than the value of this element if it has data to send back to the client or if some other client is being blocked by the current idle request.

This interval implies that the client may not be able to reach the server for the selected duration. The interval cannot be set to a negative value.

The default interval is 512 milliseconds.

Example

```
<IdleAckInterval>512</IdleAckInterval>
```

See also

[IdlePostInterval](#), [MimeType](#), [WriteBufferSize](#)

IdlePostInterval

Specifies how long Flash Player should wait before sending an idle post to the server.

Idle posts are sent when Flash Player has no data to send, but posting is necessary to provide the server with an opportunity to send data downstream to the client.

The interval for an idle post ranges from 0 to 4064 milliseconds. If the `IdlePostInterval` element is set to a value that lies outside of this range, the default value of 512 milliseconds is used.

Note: At times, the server will not be able to send any data to the client for the selected duration.

Example

```
<IdlePostInterval>512</IdlePostInterval>
```

See also

[IdleAckInterval](#), [MimeType](#), [WriteBufferSize](#)

Interface

This element defines the name to use as the outgoing network interface.

The name can be an interface name, an IP address, or a host name.

Example

```
<Interface>www.example.com</Interface>
```

See also

[MaxTimeOut \(Connections\)](#), [Reuse](#)

Interval

Specifies the interval in milliseconds for sending silence messages when no audio is being published to a live stream.

Silence messages are used to support older versions of Flash Player. The server only sends the silence message to clients specified in the `UserAgent` element in the `Client` section. Bit-flag 0x01 is used to control the silence message.

The default interval is 3 seconds. Set this to 0 to disable silence message transmission.

Example

```
<Interval>3</Interval>
```

See also

[SendSilence](#)

JoinLeaveEvents

Flash Media Server 4.5

The `JoinLeaveEvents` element controls whether events for clients that join or leave a Group are dispatched to server-side script or handled internally. This element has a `mode` attribute with the following possible values:

`None` (default)—All Group join and leave events are handled internally at this Adobe Media Server node.

`All`—All Group join and leave events are dispatched to the server-side script.

See also

[PeerLookupEvents](#)

JSEngine

Availability

Flash Communication Server 1.

In Flash Media Server 4, this element is deprecated. See [ScriptEngine](#).

Availability

Container element.

The elements nested within this container configure the JavaScript engine.

Inside the `JSEngine` element, you can define properties for the server-side Application object. Defining properties in the default `Application.xml` file creates properties available for all applications on a virtual host. Defining properties in an `Application.xml` file in an application folder creates properties available for that application only.

To define a property, create an XML tag. The property name corresponds to the tag's name, and the property value corresponds to the tag's contents.

Example

The following XML fragment defines the properties `user_name` and `dept_name`, with the values `jdoue` and `engineering`, respectively:

```
<JSEngine>
  <config>
    <user_name>jdoue</user_name>
    <dept_name>engineering</dept_name>
  </config>
</JSEngine>
```

To access the property in server-side code, use the syntax in either of these examples:

```
application.config.prop_name
application.config["prop_name"]
```

Note: The properties you define are accessible from `application.config.property`, not from `application.property`.

Contained elements

[RuntimeSize](#), [MaxGCSkipCount](#), [MaxTimeout](#) (JSEngine and ScriptEngine), [ScriptLibPath](#), [FileObject](#), [XMLSocket](#), [NetConnection](#) (ScriptEngine)

KeyFrameInterval

This element defines how often to generate and save keyframes in an FLV file.

The initial value is 60000, which is the recommended value. However, if this tag is unspecified or set to a value out of range, the server uses a default value of 1000. Setting this element to a higher value than the initial value reduces the number of keyframes added to the FLV file and thus reduces the file size. Setting a higher value for the interval, however, reduces the seeking accuracy. The value of this element is defined in milliseconds.

For example, a 15-second video with a file size of 76 KB is increased only to 89 KB when the `KeyFrameInterval` element is set to 5000, which is an increase of 13 KB, or 17%. The same video has a size of 109 KB with the `KeyFrameInterval` element set to 1000, which is an increase of 33 KB, or 43%.

Note: Be aware of the correlation between file size and accuracy of seeking when you set this value.

Example

```
<KeyFrameInterval>1000</KeyFrameInterval>
```

See also

[StorageDir](#), [DuplicateDir](#), [CacheUpdateInterval](#)

LifeTime

Container element.

This element determines the lifetime of core processes. To roll over such processes, set this element to a nonzero value.

Process rollover happens only when the `Scope` element is set to `inst`.

Contained elements

[MaxCores](#), [RollOver](#)

Live (StreamManager)

Container element.

The elements nested within this container configure the intermediate frames in a live stream and the message queue, and the amount of time the server waits before allowing another publisher to take over a live stream.

Contained elements

[AccumulatedIFrames](#), [Queue](#), [PublishTimeout](#), [StartClockOnPublish](#), [MaxLatency](#), [EraseOnPublish](#), [AssumeAbsoluteTime](#)

Live (MsgQueue)

Container element.

The elements nested within this container configure live audio.

Contained elements

[MaxAudioLatency](#), [MinBufferTime](#) ([Live](#))

LoadOnStartup

This element determines whether or not the server loads an application instance when the server starts.

Having an application instance loaded at server start-up saves time when the first client connects to that application. The default value is `false`.

If you set this element to `true`, an instance of each application on the server will be loaded at start-up.

Example

```
<LoadOnStartup>false</LoadOnStartup>
```

See also

[Process](#), [MaxAppIdleTime](#), [JSEngine](#), [StreamManager](#)

LockTimeout

Specifies the timeout value before automatically unlocking a shared object if there is a client waiting for an update. The timeout value is specified in seconds. The default value is `-1`, which instructs the server to wait for an indefinite time.

Example

```
<LockTimeout>-1</LockTimeout>
```

See also

[StorageDir](#), [DuplicateDir](#), [ResyncDepth](#), [AutoCommit](#), [MaxProperties](#), [MaxPropertySize](#)

LoCPU

This element instructs the server to stop combining samples when CPU utilization is lower than the specified percentage of CPU resources. The default percentage of utilization is `60`.

Example

```
<LoCPU>60</LoCPU>
```

See also

[Subscribers](#), [HiCPU](#), [MaxSamples](#)

Logging

A container element.

Contains elements that control the server logs.

Contained elements

[Access \(Logging\)](#)

LogInterval

The interval at which the server logs checkpoint events.

See also

[Checkpoints](#)

Max

This element defines the maximum number of redirects allowed.

See also

[Allow](#)

MaxAggMsgSize

Specifies the maximum size in bytes of the aggregate messages created from the message queue, when aggregate messages are enabled. The default value is 4096.

See also

[Queue](#), [MaxQueueSize](#), [MaxQueueDelay](#), [FlushOnData](#), [AccumulatedIFrames](#)

MaxAppIdleTime

Specifies the maximum time an application instance can remain idle with no clients connected before it is unloaded from the server's memory.

An application instance is evaluated as idle after all clients disconnect from it. If the application instance is loaded with no clients connected, it is not evaluated as idle.

The maximum idle time is specified, in seconds. The default is 600 seconds (10 minutes).

Example

```
<MaxAppIdleTime>600</MaxAppIdleTime>
```

See also

[Process](#), [LoadOnStartup](#), [JSEngine](#), [StreamManager](#), [ApplicationGC](#)

MaxAudioLatency

Specifies that live audio should be dropped if audio exceeds the time specified. Time is expressed in milliseconds.

Example

```
<MaxAudioLatency>2000</MaxAudioLatency>
```

See also

[MinBufferTime \(Live\)](#)

MaxBufferRetries

Specifies the maximum number of attempts to start buffering stream data. This configuration tag prevents the server from indefinitely trying to buffer data that is never delivered by the origin server. It is only applicable in edge servers. The default value of 128 is sufficient in most cases.

Example

```
<MaxBufferRetries>128</MaxBufferRetries>
```

See also

[StorageDir](#), [DuplicateDir](#), [CachePrefix](#), [CacheUpdateInterval](#)

MaxCores

The value of this element determines how many core processes can exist for an application.

By default, the `MaxCores` function is disabled. The default value is zero. For more information on setting the maximum number of core processes, see “[Configure how applications are assigned to server processes](#)” on page 28.

Example

```
<MaxCores>0</MaxCores>
```

See also

[LifeTime](#), [RollOver](#)

MaxDuration

Specifies the maximum duration, in seconds, of a recorded file. The value 0 disables recording. The value -1 means there is no maximum duration. The default value is -1.

Set this parameter when you deploy a DVR application to prevent the disk from exceeding its capacity.

Note: The `F_STREAM_RECORD_MAXDURATION` field in the Authorization plug-in can override this value.

See also

[MaxDurationCap](#), [MaxSizeCap](#), [MaxSize](#) (Recording)

MaxDurationCap

Specifies the cap on the maximum duration (in seconds) of a recorded file. Values set in the `MaxDuration` element or in Server-Side ActionScript cannot exceed this value. Setting the value 0 disables recording. Setting the value -1 removes the maximum duration cap. The default value is -1.

This element allows ISPs to host applications for customers and ensure that no customer abuses the `MaxDuration` limit. For example, a customer can set any `MaxDuration` for their applications, but cannot exceed the `MaxDurationCap` set by the ISP.

Note: The `F_STREAM_RECORD_MAXDURATION` field in the Authorization plug-in can override this value.

See also

[MaxDuration](#), [MaxSizeCap](#), [MaxSize](#) (Recording)

MaxGCSkipCount

Specifies the maximum number of times that the server will skip garbage collection (GC) when the JS engine is busy. This element determines the frequency of the garbage collection process.

By default, the server only performs GC when the JS engine is not busy. However, the JS engine does not necessarily perform GC when it is busy, so in some cases you must force the server to perform GC regardless of the JS engine state. If MaxGCSkipCount is set to 0, the server forces a GC regardless of the JS engine state. If MaxGCSkipCount is set to a positive value, the server forces a GC when the skip count exceeds the value in MaxGCSkipCount.

Example

```
<MaxGCSkipCount>-1</MaxGCSkipCount>
```

See also

[RuntimeSize](#), [MaxTimeout](#) ([JSEngine](#) and [ScriptEngine](#)), [ScriptLibPath](#), [FileObject](#), [XMLSocket](#), [NetConnection](#) ([ScriptEngine](#))

MaxFailures

The value of this element determines the maximum number of core process failures that can occur before a core process is disabled.

Once the core processes are disabled, the server does not launch a core process until some minimum recovery time has elapsed. Having a time lag for recovery avoids a denial-of-service action, which can happen when a faulty core consumes all CPU resources by repeatedly launching itself.

Example

```
<MaxFailures>2</MaxFailures>
```

See also

[Scope](#), [Distribute](#), [LifeTime](#), [RecoveryTime](#)

MaxIdleTime

The maximum time a client can be idle before the server closes the connection. This value is set in the Server.xml file by default. You can set the value in the Vhost.xml file to override the value set in the Server.xml file. You can set the value in the Application.xml file to override the value set in the Vhost.xml file.

See also

[AutoCloseIdleClients](#)

MaxInitDelay

The maximum number of seconds used to process SWF files. The default value is 5 seconds.

Example

```
<MaxInitDelay>5</MaxInitDelay>
```

Example

[SWFVerification](#)

MaxLatency

The maximum latency, in milliseconds, of incoming messages in a live stream. When a server-side script injects a data message into a live stream, this setting determines the timestamp of the message. If the injection time is larger than `MaxLatency`, the server adds the time difference between the last message time and the injection time to the timestamp of the data message. Otherwise, the timestamp is the last message time. The default value is 500. The minimum value is 0.

See also

[Live \(StreamManager\)](#)

MaxMessageSizeLosslessVideo

Specifies the maximum size of messages for screen-sharing packets.

Example

```
<MaxMessageSizeLosslessVideo>0</MaxMessageSizeLosslessVideo>
```

See also

[OutChunkSize](#), [AccumulatedIFrames](#), [Access](#), [UserAgent](#)

MaxPendingDebugConnections

Specifies the maximum number of pending debug connections. The default is 50. (If the number is set to 0, debugging connections are disabled.)

Once the specified number is reached, the oldest pending debug connection is rejected to create space.

Example

```
<MaxPendingDebugConnections>50</MaxPendingDebugConnections>
```

See also

[AllowDebugDefault](#)

MaxProperties

The maximum number of properties for each shared object. To specify unlimited, use -1.

Example

```
<MaxProperties>-1</MaxProperties>
```

See also

[StorageDir](#), [DuplicateDir](#), [ResyncDepth](#), [LockTimeout](#), [AutoCommit](#), [MaxPropertySize](#)

MaxPropertySize

The maximum size in bytes for each property of a shared object. To specify unlimited size, use -1.

Example

```
<MaxPropertySize>-1</MaxPropertySize>
```

See also

[StorageDir](#), [DuplicateDir](#), [ResyncDepth](#), [LockTimeout](#), [AutoCommit](#), [MaxProperties](#)

MaxQueueDelay

Specifies how often the server will flush the message queue, in milliseconds. The default value is 500 milliseconds.

See also

[Queue](#), [MaxQueueSize](#), [FlushOnData](#), [AggregateMessages](#) ([Queue](#))

MaxQueueSize

Specifies how often the server will flush the message queue, in bytes. A value of 0 disables queuing. The default value is 4096.

See also

[Queue](#), [MaxQueueDelay](#), [FlushOnData](#), [AggregateMessages](#) ([Queue](#))

MaxRate

Specifies the maximum rate in Kbps at which the server sends data to the client. The default value of -1 sends the data at whatever rate is necessary to measure bandwidth without throttling.

Example

```
<MaxRate>-1</MaxRate>
```

See also

[DataSize](#), [MinBufferTime](#) ([Live](#))

MaxSamples

Specifies how many sound samples can be combined into one message.

The default number of samples is 4.

See also

[Audio](#)

MaxSize (AccumulatedIFrames)

Specifies the maximum size, in kilobytes, of intermediate frames that a live stream can hold in the buffer.

The buffer contains a history of the video messages up to the last keyframe. This enables clients to catch up to the latest message even if they join between keyframes. If the buffer size is larger than `MaxSize`, the server clears the messages. This setting prevents the buffer from growing too large and should be set larger than the total size of intermediate frames between keyframes. A default value of -1 means the size of intermediate frames is unlimited.

Example

```
<MaxSize>-1</MaxSize>
```

See also

[Subscribers](#), [LoCPU](#), [HiCPU](#)

MaxSize (Recording)

Specifies the maximum size, in kilobytes, of a recorded file. The value -1 means there is no maximum size. The value 0 disables recording. The default value is -1.

Set this parameter when you deploy a DVR application to prevent the disk from exceeding its capacity.

Note: The `F_STREAM_RECORD_MAXSIZE` field in the Authorization plug-in can override this value.

See also

[MaxSizeCap](#), [MaxDuration](#), [MaxDurationCap](#)

MaxSizeCap

Specifies the maximum size cap, in kilobytes, of a recorded file. The value -1 means there is no maximum size. The value 0 disables recording. The default value is -1.

This element allows ISPs to host applications for customers and ensure that no customer abuses the `MaxSize` limit. For example, a customer can set any `MaxSize` for their applications, but cannot exceed the `MaxSizeCap` set by the ISP.

Note: The `F_STREAM_RECORD_MAXSIZE` field in the Authorization plug-in can override this value.

See also

[MaxSize \(Recording\)](#), [MaxDuration](#), [MaxDurationCap](#)

MaxStreamsBeforeGC

Specifies that garbage collection (GC) should be forced if the stream list grows over the set value. The default value is -1 (unlimited). GC occurs during the application GC interval.

Example

```
<MaxStreamsBeforeGC>-1</MaxStreamsBeforeGC>
```

See also

[StorageDir](#), [DuplicateDir](#), [CachePrefix](#), [CacheUpdateInterval](#), [MaxBufferRetries](#)

MaxTime

Specifies the maximum duration, in seconds, of intermediate frames that a live stream can hold in the buffer.

The buffer contains a history of the video messages up to the last keyframe. This enables clients to catch up to the latest message even if they join between keyframes. If the duration in the buffer is larger than the `MaxTime`, the server clears the messages. This setting prevents the buffer from growing too large and should be set larger than the keyframe interval. The default value of -1 means the duration is unlimited.

Example

```
<MaxTime>-1</MaxTime>
```

See also

[RuntimeSize](#), [MaxGCSSkipCount](#), [ScriptLibPath](#), [FileObject](#), [XMLSocket](#), [NetConnection](#) ([ScriptEngine](#))

MaxTimeout (Connections)

This element defines the maximum time for a transfer to be completed. The default time is 60 seconds.

Operations such as DNS lock-ups may take more time. If the value of this element is too low, the risk of aborting correctly functioning operations increases.

Example

```
<MaxTimeout>60</MaxTimeout>
```

See also

[Reuse](#), [Interface](#)

MaxTimeout (JSEngine and ScriptEngine)

Note: In Flash Media Server 4, the `JSEngine` element is deprecated and replaced by the `ScriptEngine` element.

The maximum time, in seconds, a script can take to execute a JavaScript (Server-Side ActionScript) function. If its execution takes longer than the maximum allowed time, then the script is evaluated as a runaway script and its execution is terminated. Setting a maximum time to execute a script prevents infinite looping in scripts.

The default value is 0 and no checks are performed to detect runaway scripts. This setting may be useful in a debugging environment. In a production environment, after the applications and scripts have been thoroughly tested, you should set this element to a more realistic value that does not impose limits on the time scripts take to execute.

Example

```
<MaxTimeout>0</MaxTimeout>
```

See also

[RuntimeSize](#), [MaxGCSSkipCount](#), [ScriptLibPath](#), [FileObject](#), [XMLSocket](#), [NetConnection](#) ([ScriptEngine](#))

MaxUnprocessedChars

Specifies how much data can be received from an XML server (without receiving an end tag) before `XMLSocket` closes the connection. This can be overridden by each `XMLSocket` by specifying the property `XML.maxUnprocessedChars`, but that number cannot exceed the number specified in this element.

Example

```
<MaxUnprocessedChars>4096</MaxUnprocessedChars>
```

See also

[XMLSocket](#)

MaxUrlLength

Defines the maximum URL lengths, in bytes. Most modern browsers support up to 64KB URLs. The default value is 32KB. The maximum allowable value is 1000KB.

Use this element to restrict the lengths of outgoing SSAS-initiated NetConnection URLs. For incoming requests, use the `MaxUrlLength` element in the `Server.xml` file.

Example

```
<MaxUrlLength>65536</MaxUrlLength>
```

See also

[Security](#)

MaxWait

This element specifies the number of seconds to wait before the server sends data to the client.

Increasing this number provides a more accurate bandwidth figure, but it also forces the client to wait longer.

Example

```
<MaxWait>4096</MaxWait>
```

See also

[MaxRate](#), [DataSize](#)

MimeType

Specifies the default MIME (Multi-purpose Internet Mail Extensions) type header sent on tunnel responses.

The server generally uses the MIME type specified by the incoming requests. The server will use the entry for the `MIMETYPE` element only when it is unable to determine the MIME type from the incoming requests.

Example

```
<MimeType>application/x-fcs</MimeType>
```

See also

[IdleAckInterval](#), [IdlePostInterval](#), [WriteBufferSize](#)

MinBufferTime (Live)

Specifies the default buffer length in milliseconds for the live audio and video queue.

Example

```
<MinBufferTime>2000</MinBufferTime>
```

See also

[MaxAudioLatency](#)

MinBufferTime (Recorded)

Specifies the default buffer length in milliseconds for audio and video. The value cannot be set below this by Flash Player.

Example

```
<MinBufferTime>2000</MinBufferTime>
```

See also

[Recorded](#)

MinGoodVersion

Specifies the minimum accepted version of SWF verification allowed by the server. The default value is 0, which allows the current and all future versions.

Example

```
<MinGoodVersion>0</MinGoodVersion>
```

See also

[SWFFolder](#), [UserAgentExceptions](#)

MinQueuedVideo

The minimum number of video messages to queue at the start of a stream. Streams that use H.264 or any other pipelined codec need messages to begin playback. This setting ensures that regardless of the buffer setting there are enough messages to begin playback quickly. Setting this value to less than 64 may cause content with a low FPS to delay before starting. The default value is 64.

See also

[Server](#)

MsgQueue

Container element.

The elements nested within this container configure live and recorded audio.

Contained elements

[Live](#) ([MsgQueue](#)), [Recorded](#), [Server](#)

NetConnection (ScriptEngine)

Container element.

The element nested within this container specifies object encoding to use for Server-Side ActionScript NetConnection.

Contained elements

[ObjectEncoding](#)

NetConnection (RPC)

Lets you list methods allowed for NetConnection RPCs. Any method not listed is blocked.

When a client creates a NetConnection object and attempts to call an SSAS Client class method with that object, the method must be explicitly allowed in this section for the call to succeed.

To add methods, use a comma-delimited list of method names in the `<Method><Allow>` sub-elements.

The default is to allow the `onStatus` method for NetConnection RPCs.

Contained elements

Method

Example

```
<NetConnection>
  <Method>
    <Allow>onStatus</Allow>
  </Method>
</NetConnection>
```

See also

[Stream](#), [SharedObject](#), [Client](#) (RPC)

Availability

Flash Media Server 4

NotifyAudioStop

Container element.

The `Duration` element nested within this container determines whether or not the server is notified when an audio transmission ending on a stream is encountered.

Example

```
<NotifyAudioStop enabled="false"></NotifyAudioStop>
```

Contained elements

[Duration](#)

ObjectEncoding

Specifies the default object encoding to use for SSAS NetConnection. This can be `AMF0` or `AMF3`. The default is `AMF3`.

The default can be overridden for each individual NetConnection by setting the `NetConnection.objectEncoding` property to either 0 for `AMF0` or 3 for `AMF3`.

Example

```
<ObjectEncoding>AMF3</ObjectEncoding>
```

See also

[NetConnection](#) ([ScriptEngine](#))

OutChunkSize

Specifies the RTMP chunk size in bytes to use in all streams for this application. The server breaks stream content into chunks of this size. Larger values reduce CPU usage but cause larger writes that can delay other content on lower bandwidth connections. This can have a minimum value of 128 bytes and a maximum value of 65536 bytes. The default value is 4096.

Note that older clients might not support chunk sizes larger than 1024 bytes. If the chunk setting is larger than these clients can support, the chunk setting is capped at 1024 bytes.

Example

```
<OutChunkSize>4096</OutChunkSize>
```

See also

[Bandwidth](#), [BandwidthCap](#), [BandwidthDetection](#), [MsgQueue](#), [HTTPTunnel](#), [MaxMessageSizeLosslessVideo](#)

OverridePublisher

Deprecated; see the [PublishTimeout](#) element.

Specifies whether a second client is able to take over the ownership of a live stream when the stream is already published by another client. The default value is `false`. If set to `true`, add application logic to avoid stream name collision.

Example

```
<OverridePublisher>true</OverridePublisher>
```

See also

[Audio](#), [Live \(StreamManager\)](#), [SendDuplicateStart](#), [SendDuplicateOnMetaData](#)

Password

Specifies the password for connecting to the proxy.

See also

[Host](#), [Port](#), [Type](#), [Tunnel](#), [Username](#)

PeerLookupEvents

Flash Media Server 4.5

In a peer-assisted networking application, use this element to configure how the server handles peer lookup events.

RTMFP clients establish direct peer-to-peer connections through the use of NetStream `DIRECT_CONNECTIONS` and peerIDs, or by using NetGroup to join a Flash Group where clients can be automatically bootstrapped to neighbors or can manually initiate connections to them. The process of establishing a peer-to-peer connection begins with the initiating client sending a peer lookup request to the server. These events are processed differently depending upon the `mode` attribute.

The following are possible values of the `mode` attribute:

Value	Description
None	The default value. Lookup events are handled by the server. You cannot use Server-Side ActionScript to distribute peer introductions or filter peer introductions.
Partial	The server automatically handles peer lookup events for clients connected to the same core process. Developers must handle other peer lookup events with Server-Side ActionScript.
All	Developers must handle all lookup events in Server-Side ActionScript.

To filter peer lookup requests, set `<PeerLookupEvents mode>` to `All`:

```
<Application>
  ...
  <RTMFP>
    <PeerLookupEvents mode="All" />
  </RTMFP>
  ...
</Application>
```

To distribute introductions across multiple servers, set `mode` to `Partial` or `All`.

For more information, see [Filter introduction requests](#) and [Build peer-assisted networking applications to run on multiple servers](#).

Port

Specifies the proxy port to connect to if the port is not specified as part of the host in the `Host` element.

See also

[Host](#), [Password](#), [Type](#), [Tunnel](#), [Username](#)

Prioritization

Specifies whether outgoing messages are prioritized by message type when sending across a server-to-server connection. This setting is relevant for multi-point publishing. By default, prioritization is set to `false`, which is the correct setting to avoid possible latency when server-side `NetStream` objects are used to publish messages to remote servers. Messages are sent out through one channel and all messages have the same priority.

If the value is set to `true`, the server sends messages through multiple channels and prioritizes messages based on the message type, as follows (where 1 has the highest priority):

- 1 Data
- 2 Audio
- 3 Video

See also

[Server](#)

Process

Container element.

The elements nested within this container determine how a core process is managed.

The following table lists descriptions of the contained elements.

Value	Description
Scope	Specifies the level at which application instances are assigned to core processes. Scopes have an enclosing relationship with a strict ordering: adaptors contain virtual hosts, which contain applications, which contain instances, which contain clients.
Distribute	Specifies how to distribute application instances to processes. The value of the <code>Distribute</code> tag must be a scope that is lower in order than the value in the <code>Scope</code> tag (for example, if the value of <code>Scope</code> is <code>adaptor</code> , the value of <code>Distribute</code> can be <code>vhosts</code> , <code>apps</code> , <code>insts</code> , or <code>clients</code>). Distribution may be turned off by setting <code>numproc</code> to 0 or 1.
LifeTime	Specifies the lifetime of core processes. Process rollover happens only when the <code>Scope</code> element is set to <code>inst</code> .
MaxFailures	The value for this element determines the maximum number of core process failures that can occur before a core process is disabled.
RecoveryTime	Specifies the recovery time for a core.

Contained elements

[Scope](#), [Distribute](#), [LifeTime](#), [MaxFailures](#), [RecoveryTime](#),

Proxy

Container element.

The elements nested within this container configure the HTTP Proxy settings.

Contained elements

[Host](#), [Port](#), [Type](#), [Tunnel](#), [Username](#), [Password](#)

PublishTimeout

Specifies how long in milliseconds the server waits to receive a response from a publisher when another client tries to publish to the same stream.

If a client tries to publish to the same live stream that is being published by another client, Adobe Media Server pings the first publisher and waits to receive a response. If the first publisher fails to respond within the time specified in this tag, the server allows the second publisher to take over the live stream. The default value is 2000 milliseconds. To prevent the server from pinging the first client, disable this setting by setting the value of the tag to -1.

This tag replaces the `OverridePublisher` tag.

QualifiedStreamsMapping

Enables mapping of virtual stream paths to different physical locations. This tag is applicable when using the Authorization plug-in.

When a client requests to play a stream, an `E_FILENAME_TRANSFORM` event occurs in the Authorization plug-in. You can map the stream differently for each client in the `F_STREAM_PATH` property of your Authorization plug-in code.

For example, suppose client 1 and client 2 both request to play `myStream.flv`. You can remap the stream to `c:\video1\myStream.flv` for client 1 and `c:\video2\myStream.flv` for client 2.

Queue

Container element; contains elements that configure the settings of the message queue. A message queue is used to buffer incoming messages from the publisher so that the server can send messages in chunks to the subscribers. You can disable queuing so that individual messages are immediately sent to subscribers. To disable queuing, set the `enabled` attribute to `false`.

Contained elements

[MaxQueueSize](#), [MaxQueueDelay](#), [FlushOnData](#), [AggregateMessages](#) ([Queue](#))

Recorded

Container element.

The element nested within this container specifies the ratio of buffer length used by the server-side stream to the live buffer.

Contained elements

[MinBufferTime](#) ([Recorded](#))

Recording (HDS)

Container for configurations for HTTP Dynamic Streaming.

Contained Elements

[“ContentProtection”](#) on page 151

Recording (StreamManager)

Container element.

The elements nested within this container specify the duration and file size of files recorded by the server.

Contained elements

[MaxDuration](#), [MaxDurationCap](#), [MaxSize](#) ([Recording](#)), [MaxSizeCap](#)

RecoveryTime

Specifies the recovery time for a core.

The server will not launch a core process until some minimum recovery time has elapsed. The time lag for recovery can avoid a denial-of-service action, which happens when a faulty core process consumes all CPU time by repeatedly launching itself.

The recovery time for a core process is specified, in seconds. A value of 0 disables any checking for process failures.

Note: Loading an application with Adobe Media Administration Server tools or APIs bypasses this check.

Example

```
<RecoveryTime>300</RecoveryTime>
```

See also

[Scope](#), [Distribute](#), [LifeTime](#), [MaxFailures](#)

Redirect

Container element.

The elements nested within this container configure the settings for redirecting the HTTP connection.

Contained elements

[Allow](#), [Max](#), [UnrestrictedAuth](#)

ResyncDepth

This element instructs the server to resynchronize a shared object file. The shared object is resynchronized when its version number is greater than the head version minus the current version. The default value of -1 sends a resynchronized version of the file with every connection.

Example

```
<ResyncDepth>-1</ResyncDepth>
```

See also

[StorageDir](#), [DuplicateDir](#), [LockTimeout](#), [AutoCommit](#), [MaxProperties](#), [MaxPropertySize](#)

Reuse

This element configures whether or not the server explicitly closes the HTTP connection after each transfer. The default is to reuse connections. Set this to `false` to use a new connection after every transfer.

Example

```
<Reuse>true</Reuse>
```

See also

[MaxTimeOut \(Connections\)](#), [Interface](#)

RollOver

Specifies how many seconds a core process can be in use before the server creates a new core process.

After the time limit for a core is reached, a new core is instantiated. All subsequent connections are directed to the new core.

The rollover functionality is disabled by default. The default value is 0 (seconds). For more information on rollover processes, see “[Configure how applications are assigned to server processes](#)” on page 28.

Example

```
<RollOver>0</RollOver>
```

See also

[MaxCores](#)

RPC

Configures RPCs (remote procedure calls) for each class. By default, all methods are rejected. Only methods that are explicitly listed for each class are allowed. This whitelisting approach is more secure than a blacklist.

Contained Elements

[NetConnection](#) (RPC), [Stream](#), [SharedObject](#), [Client](#) (RPC)

Availability

Flash Media Server 4

RTMFP

This section provides the means to control the behavior of application-specific RTMFP functionality.

Contained Elements

[“GroupControl”](#) on page 156

[“JoinLeaveEvents”](#) on page 159

[“PeerLookupEvents”](#) on page 172

RuntimeSize

Specifies the maximum size in kilobytes that an application instance can use to run Server-Side ActionScript code before the server removes unreferenced and unused JavaScript objects.

The default size is 1024 kilobytes, which is the equivalent of 1 megabyte. The lower limit is 10 kilobytes. There is no upper limit. The default value applies when the engine size lies outside of these limits.

If your application consumes a significant amount of memory, you must increase the engine size. If you create a new script object that will cause the runtime size of the application instance to exceed the value of this element, an out-of-memory error occurs and the application instance is shut down. In most cases, increasing the engine size to 30720 (30 MB) is sufficient to run intensive Server-Side ActionScript operations.

Example

```
<RuntimeSize>1024</RuntimeSize>
```

See also

[MaxGCSkipCount](#), [MaxTimeout](#) (JSEngine and ScriptEngine), [ScriptLibPath](#), [FileObject](#), [XMLSocket](#), [NetConnection](#) (ScriptEngine)

Scope

This element determines the level at which application instances are assigned to core processes.

Starting Adobe Media Server starts a process called AMSMaster.exe (Windows) or amsmaster (Linux). Application instances run in processes called AMSCore.exe (Windows) amscore (Linux). The master process is a monitor that starts core processes when necessary. Only one master process can run at a time, but many core processes can run at the same time.

Settings in an Application.xml file in a virtual host folder apply to all applications running in that virtual host. Settings made in an Application.xml file in an application’s folder apply only to that application.

The following table lists the values available for the `Scope` element.

Value	Description
adaptor	All application instances in an adaptor run together in a process.
vhost	All application instances in a virtual host run together in a process. This is the default value.
app	All instances of a single application run together in a process.
inst	Each application instance runs in its own process. If you choose this value, you must also set the <code>Distribute numprocs</code> attribute to a value greater than 1.

Example

```
<Scope>vhost</Scope>
```

See also

[Distribute](#), [LifeTime](#), [MaxFailures](#), [RecoveryTime](#)

ScriptEngine

Availability

Flash Media Server 4

Replaces the `JSEngine` element.

Description

Container element.

The elements nested within this container configure the server-side script engine.

Inside the `ScriptEngine` element, you can define properties for the server-side Application object. Defining properties in the default `Application.xml` file creates properties available for all applications on a virtual host. Defining properties in an `Application.xml` file in an application folder creates properties available for that application only.

To define a property, create an XML tag. The property name corresponds to the tag's name, and the property value corresponds to the tag's contents.

Example

The following XML fragment defines the properties `user_name` and `dept_name`, with the values `jdoe` and `engineering`, respectively:

```
<ScriptEngine>
  <config>
    <user_name>jdoe</user_name>
    <dept_name>engineering</dept_name>
  </config>
</ScriptEngine>
```

To access the property in server-side code, use the syntax in either of these examples:

```
application.config.prop_name
application.config["prop_name"]
```

Note: The properties you define are accessible from `application.config.property`, not from `application.property`.

Contained elements

[AdaptorId](#), [RuntimeSize](#), [MaxGCSSkipCount](#), [MaxTimeout \(JSEngine and ScriptEngine\)](#), [ScriptLibPath](#), [FileObject](#), [XMLSocket](#), [NetConnection \(ScriptEngine\)](#), [Security](#)

ScriptLibPath

A list of paths delimited by semicolons instructing the server where to look for server-side scripts loaded into a main.asc file with the `load()` method.

These paths are used to resolve a script file that is loaded with the load API. The server first looks in the location where the main.asc or *application_name*.asc file is located. If the script file is not found there, the script engine searches, in sequence, the list of paths specified in this element.

Example

```
<ScriptLibPath>${APP.JS_SCRIPTLIBPATH}</ScriptLibPath>
```

See also

[RuntimeSize](#), [MaxGCSSkipCount](#), [MaxTimeout \(JSEngine and ScriptEngine\)](#), [FileObject](#), [XMLSocket](#), [NetConnection \(ScriptEngine\)](#)

Security

Container element.

Contains elements that define security features for the application.

Contained elements

[MaxUrlLength](#), [RPC](#)

Availability

Flash Media Server 4

SendDuplicateOnMetaData

Specifies whether an `onMetaData` message is sent at the beginning of video files when the play and seek commands are called. The default value is `true`.

The following values are available:

- `true` sends `onMetaData` for the play and seek commands.
- `false` sends `onMetaData` for play only.
- `once` falls back to Flash Media Server 1.x behavior and sends `onMetaData` based on the start position, regardless of the command. If no `onMetaData` is found at the start position, no `onMetaData` is sent.

Example

```
<SendDuplicateOnMetaData>true</SendDuplicateOnMetaData>
```

See also

[SendDuplicateStart](#)

SendDuplicateStart

Specifies whether status message `NetStream.Play.Start` is sent for all commands, including play, seek, and unpause. If set to `false`, only the play command receives the start message.

Example

```
<SendDuplicateStart>true</SendDuplicateStart>
```

See also

[SendDuplicateOnMetaData](#), [OverridePublisher](#)

SendSilence

Container element.

The `Interval` element nested within this container configures the settings for sending silent messages.

Contained elements

[Interval](#)

Server

Container element.

Contains two elements: `BufferRatio`, which specifies the ratio of the buffer length used by the server-side stream to the live buffer, and `Prioritization`, which specifies whether to prioritize outgoing messages for server-to-server connections.

Contained elements

[BufferRatio](#), [Prioritization](#)

ServerToClient (Bandwidth)

Specifies the bandwidth in bytes per second that the server can use for sending data downstream to the client.

The default bandwidth is 250,000 bytes per second.

You can configure this value in the `Bandwidth` section of XML and in the `BandwidthCap` section of XML. The values in the `Bandwidth` section can be overridden, but the values in the `BandwidthCap` section are not. This allows ISPs to host applications for customers and ensure that no customer abuses the bandwidth limit. For example, a customer can set any bandwidth limit for their applications, but cannot exceed the caps set by the ISP.

Example

```
<ServerToClient>250000</ServerToClient>
```

See also

[ClientToServer \(Bandwidth\)](#)

ServerToClient (BandwidthCap)

Specifies the maximum bandwidth in bytes per second that the server can use for sending data downstream to the client.

The default bandwidth is 10,000,000 bytes per second.

You can configure this value in the `Bandwidth` section of XML and in the `BandwidthCap` section of XML. The values in the `Bandwidth` section can be overridden, but the values in the `BandwidthCap` section can not. This allows ISPs to host applications for customers and ensure that no customer abuses the bandwidth limit. For example, a customer can set any bandwidth limit for their applications, but cannot exceed the caps set by the ISP.

Example

```
<ServerToClient>10000000</ServerToClient>
```

See also

[ClientToServer](#) ([BandwidthCap](#))

SharedObject

Lets you list methods allowed for SharedObject RPCs. Any method not explicitly listed here is blocked.

If a client subscribes to a remote SharedObject and tries to call remote methods on SSAS using the `SO.send()` method, the method must be explicitly listed here to succeed.

To add methods, use a comma-delimited list of method names in the `<Method><Allow>` sub-elements.

The default is to not allow any method calls for SharedObject RPCs.

Contained elements

Method

Example

```
<SharedObject>
  <Method>
    <Allow>setFps</Allow>
  </Method>
</SharedObject>
```

See also

[NetConnection](#) (RPC), [Stream](#), [Client](#) (RPC)

Availability

Flash Media Server 4

SharedObjManager

Container element.

The elements nested within this container configure the Shared Object Manager setting of an application.

Contained elements

[StorageDir](#), [DuplicateDir](#), [ResyncDepth](#), [LockTimeout](#), [AutoCommit](#), [MaxProperties](#), [MaxPropertySize](#)

StartClockOnPublish

Specifies whether the stream time is sensitive to the difference between publish time and the arrival of the first stream message. The default value is `false`.

To use dynamic, multibitrate streaming, leave this value `false`.

See also

[Live \(StreamManager\)](#)

StorageDir

Specifies the physical location where shared objects or streams are stored.

By default the physical location is not set. Set this element only if the files for shared objects or recorded streams must be stored in a location other than the application directory.

Example

```
<StorageDir>C:\myapp\sharedobjects\</StorageDir>
<StorageDir>C:\myapp\streams\</StorageDir>
```

See also

[DuplicateDir](#)

Stream

Lets you list methods allowed for stream RPCs.

To add methods, use a comma-delimited list of method names in the `<Method><Allow>` sub-elements.

The default is to allow `onStatus` method calls only for stream RPCs.

Contained elements

Method

Example

```
<Stream>
  <Method>
    <Allow>onStatus</Allow>
  </Method>
</Stream>
```

See also

[NetConnection \(RPC\)](#), [SharedObject](#), [Client \(RPC\)](#)

Availability

Flash Media Server 4

StreamManager

Container element.

The elements in this section configure the Stream Manager settings for this application.

Contained elements

[StorageDir](#), [DuplicateDir](#), [EventsDir](#), [CachePrefix](#), [CacheUpdateInterval](#), [MaxBufferRetries](#), [ThrottleBoundaryRequest](#), [ThrottleLoads](#), [ThrottleDisplayInterval](#), [EnhancedSeek](#), [KeyFrameInterval](#), [MaxStreamsBeforeGC](#), [Audio](#), [Live \(StreamManager\)](#), [SendDuplicateStart](#), [SendDuplicateOnMetaData](#), [MaxSize \(Recording\)](#), [MaxSizeCap](#), [MaxDuration](#), [MaxDurationCap](#)

Subscribers

This element instructs the server to combine sound samples only if there are more than the default number of subscribers to that stream. The default number of subscribers is 8.

Example

```
<Subscribers>8</Subscribers>
```

See also

[LoCPU](#), [HiCPU](#), [MaxSamples](#)

SWFFolder

Specifies a single folder or a semicolon-delimited list of folders containing copies of client SWF files. The server compares these SWF files to client SWF files connecting to applications on the server.

The default value of the `SWFFolder` element is the application's folder appended with `/SWFs`. Use a semicolon to separate multiple directories. SWF files located under an instance named folder can only connect to that specific instance.

Example

For an application named `myApplication` located at `C:\applications\`, authenticating SWF files should be placed in `C:\applications\myApplication\SWFs`.

See also

[MinGoodVersion](#), [UserAgentExceptions](#)

SWFVerification

Container element.

Specifies how the server verifies client SWF files before allowing the files to connect to an application. Verifying SWF files is a security measure that prevents someone from creating their own SWF files that can attempt to stream your resources. For more information, see “[Verify SWF files](#)” on page 33.

Note: SWF files connecting to Adobe Media Administration Server cannot be verified.

Contained elements

[SWFFolder](#), [MinGoodVersion](#), [DirLevelSWFScan](#), [MaxInitDelay](#), [FinalHashTimeout](#), [FirstHashTimeout](#), [UserAgentExceptions](#), [Cache](#)

ThrottleBoundaryRequest

Controls the maximum number of concurrent boundary requests per recorded stream. When streaming through a proxy server, the boundary information about video segments are sent to the proxy server by request.

The default value is 8.

Example

```
<ThrottleBoundaryRequest enable="false">8</ThrottleBoundaryRequest>
```

See also

[ThrottleDisplayInterval](#), [ThrottleLoads](#)

ThrottleDisplayInterval

Controls the interval at which the server displays the throttle queue length. The default value is 64, which means the server displays the message 1 out of 64 times when the throttle queue is full.

Example

```
<ThrottleDisplayInterval>64</ThrottleDisplayInterval>
```

See also

[ThrottleBoundaryRequest](#), [ThrottleLoads](#)

ThrottleLoads

Controls the maximum number of concurrent segment loads per recorded stream. When streaming through a proxy server, video segments are sent to the proxy server by request. The default value is 8.

Example

```
<ThrottleLoads enable="true">8</ThrottleLoads>
```

See also

[ThrottleBoundaryRequest](#), [ThrottleDisplayInterval](#)

Tunnel

Specifies whether or not to tunnel all operations through a given HTTP proxy. The default setting is `false`.

Example

```
<Tunnel>false</Tunnel>
```

See also

[Host](#), [Port](#), [Type](#), [Username](#), [Password](#)

TTL

Specifies in minutes how long each SWF file remains in the cache. The default value is 1440 minutes (24 hours).

See also

[Cache](#), [UpdateInterval](#)

Type

Specifies the type of proxy being connected to. The value for this element can be `HTTP` or `SOCKS5`. The default is `HTTP`.

Example

```
<Type>HTTP</Type>
```

See also

[Host](#), [Port](#), [Tunnel](#), [Username](#), [Password](#)

UnrestrictedAuth

A Boolean value that determines whether or not to allow sending the user name/password combination with each HTTP redirect. Sending the user name/password combination is useful only if the `Allow` element permits redirections. The default setting is `true`.

Example

```
<UnrestrictedAuth>true</UnrestrictedAuth>
```

See also

[Allow](#), [Max](#)

UpdateInterval

Specifies the maximum time in minutes to wait for the server to scan the SWF folders for updates when there is a miss in the cache. The default value is 5 minutes.

See also

[Cache](#), [TTL](#)

UserAgent

Container element.

The settings for clients vary according to whether Flash Player platform is Windows or Macintosh. Setting the value `0x01` will configure the player and platform for silent messages.

Contained elements

[Bits](#)

UserAgentExceptions

Container element.

Contains an element that specifies a user agent that should be an exception to authentication. Use the `to` and `from` attributes to indicate the lowest and highest versions to except.

Example

```
<UserAgentExceptions>  
  <Exception to="WIN 9,0,28,0" from="WIN 9,0,28,0"></Exception>  
</UserAgentExceptions>
```

Contained elements

[Exception](#)

Username

Specifies the user name for connecting to the edge.

See also

[Host](#), [Port](#), [Type](#), [Tunnel](#), [Password](#)

Verbose

This element determines whether or not the server outputs verbose information during HTTP operations.

Example

```
<Verbose>>false</Verbose>
```

See also

[HTTP1_0](#), [Connections](#), [Proxy](#), [Redirect](#)

VideoSampleAccess

Allows the client application to access the raw uncompressed video data in a stream. By default, this element is disabled. To enable it, set the enable attribute to true. In the tag, specify a list of semicolon-delimited folders to which client applications have access. When this element is enabled, all clients can access the video data in streams in the specified folders. To enable access to all video data streamed by the server, specify / in the tag.

The folder path is restricted to the application's streams folder or folders, so do not use absolute paths in the list of folders.

While you can also enable access through Server-Side ActionScript, this element allows access to the data without requiring Server-Side ActionScript. You can also override this element with the Access plug-in or Server-Side ActionScript.

Example

If an application is configured to store streams in folders C:\low_quality and C:\high_quality, the configuration to allow access to sample those streams is as follows:

```
<VideoSampleAccess enabled="true">low_quality;high_quality</VideoSampleAccess>
```

See also

[AudioSampleAccess](#)

VirtualDirectory

Specifies virtual directory mappings for Server-Side ActionScript File objects.

If you map File objects in the Application.xml file and don't have the feature enabled in the Server.xml file, you'll see the following message in the log:

```
"Virtual directories for file objects is not supported due to the Server level security setting."
```

The `VirtualDirectory` element is contained by a `FileObject` element. The `FileObject` element has an `override` attribute whose default value is "no". When `override="no"`, the application-level `Application.xml` file cannot override the `VirtualDirectory` settings for the `FileObject` defined in a Vhost-level `Application.xml` file.

Use the following syntax:

```
<VirtualDirectory>virtual_dir_name;physical_dir_path</VirtualDirectory>
```

To specify multiple directories, use multiple `<VirtualDirectory>` elements, as in the following:

```
<FileObject>
  <VirtualDirectory>/key1;C:\example\folder1</VirtualDirectory>
  <VirtualDirectory>/key2;C:\example\folder2</VirtualDirectory>
</FileObject>
```

For more information, see comments in the `Application.xml` file and “[Mapping virtual directories to physical directories](#)” on page 51.

See also

[FileObject](#)

WindowsPerAck

Controls how many messages can be sent before receiving an acknowledgement from the other end.

Note: Adobe recommends that you do not change the value of this element.

WriteBufferSize

Specifies in kilobytes the size of the write buffer. The default size is 16 KB.

Example

```
<WriteBufferSize>16</WriteBufferSize>
```

See also

[IdlePostInterval](#), [IdleAckInterval](#), [MimeType](#)

XMLSocket

Container element.

Contains an element that specifies how much data can be received from the XML server (without receiving an end tag) before `XMLSocket` closes the connection. This can be overridden by each `XMLSocket` by specifying the property `XML.maxUnprocessedChars`, but that number cannot exceed the number specified in this element.

Example

```
<XMLSocket>
  <MaxUnprocessedChars>4096</MaxUnprocessedChars>
</XMLSocket>
```

Contained elements

[MaxUnprocessedChars](#)

Logger.xml file

The Logger.xml file is located at the root level of the /conf directory and is the configuration file for the logging file system. Logger.xml contains the elements and information used to configure log files. You can edit this file to add or change configuration information, including the location of the log files. The default location of the log files is in the /logs directory in the server installation directory.

Log files are written in English. Field names displayed in the log file are in English. Some content within the log file, however, may be in another language, depending on the filename and the operating system. For example, in the Access.log file, the columns `x-sname` and `x-suri-stem` show the name of the stream. If the name of the recorded stream is in a language other than English, the name is written in that language, even if the server is running on an English-language operating system.

The `Logging` section in the Server.xml file enables or disables the log files.

To see the element structure and default values, see the Logger.xml file installed in the `RootInstall/conf/` folder.

Note: Log file rotation cannot be disabled. To effectively turn off rotation, choose a large maximum size and a long maximum duration for the log files.

Access

Container element.

The elements nested within this container configure the Access log settings.

Contained elements

[LogServer](#), [Directory](#), [FileName](#), [Time](#)

admin

Container element.

The elements nested within this container configure the admin log settings.

Contained elements

[LogServer](#)

Application

Container element.

The elements nested within this container configure the Application log file settings.

Contained elements

[Directory](#), [Time](#), [Rotation](#)

AuthEvent

Container element.

The elements in this section configure the AuthEvent log file settings.

Contained elements

[LogServer](#), [Directory](#), [FileName](#), [Time](#), [Rotation](#), [Events](#), [Fields](#), [Delimiter](#), [QuoteFields](#), [EscapeFields](#)

AuthMessage

Container element.

The elements in this section configure the AuthMessage log file settings.

Contained elements

[Directory](#), [Time](#), [Rotation](#)

core

Container element.

The elements nested within this container configure the core log settings.

Contained elements

[LogServer](#)

Delimiter

Specifies whether or not to use a single quotation mark (') as a delimiter to separate the fields in the log file.

A delimiter is used to separate the fields in the log file. The use of the number sign (#) as a delimiter is not recommended, since # is used as the comment element in the Logger.xml file.

The following characters are not allowed as delimiters:

- triple quotation marks (' ' ')
- paired double quotation marks ("")
- comma (,)
- colon (:)
- hyphen (-)

See also

[Directory](#), [EscapeFields](#), [QuoteFields](#)

Diagnostic

Container element.

The elements in this section configure the diagnostic log file.

Contained elements

[Directory](#), [Time](#), [Rotation](#)

Directory

Specifies the directory where the log files are located.

By default, the log files are located in the logs directory in the server installation directory.

Example

```
<Directory>${LOGGER.LOGDIR}</Directory>
```

See also

[Time](#), [Rotation](#)

DisplayFieldsHeader

Formatting element. Specifies how many lines to write to the log file before repeating the field headers. The default line count is 100 lines.

Example

```
<DisplayFieldsHeader>100</DisplayFieldsHeader>
```

See also

[Delimiter](#), [EscapeFields](#), [QuoteFields](#)

edge

Container element.

The elements nested within this container configure the edge log settings.

Contained elements

[LogServer](#)

EscapeFields

Formatting element. This element controls whether or not the fields in the log file are escaped when unsafe characters are found. This optional flag can be set to `enable` or `disable`. By default, it is set to `enable`.

The unsafe characters are as follows: the space character; open or closed angle brackets (< >); a double quotation mark ("); the number sign (#); the percent sign (%); open or closed curly braces ({ }); bar (|); the carat (^); the tilde (~); square brackets ([]); and the apostrophe (').

Example

```
<EscapeFields>enable</EscapeFields>
```

See also

[LogServer](#), [Directory](#), [FileName](#), [Time](#), [Rotation](#), [Events](#), [Fields](#), [Delimiter](#), and [QuoteFields](#)

Events

Specifies the events written to the log file.

Specify events in a semicolon-separated list. The keyword `*` instructs the server to log all events. For a list of events that are recorded in the Access log file, see “[Access events defined in access logs](#)” on page 99.

See also

[Fields](#)

Fields

Specifies which fields for an event are logged in the Access log file.

Fields are associated with the events found in the Access log file. The field specification is a semicolon-separated list of one or more fields associated with an event in the log file.

The keyword * specifies that all fields are to be logged. Fields without data are left empty. Adobe recommends that you include the following fields in the fields to be logged: the type, category, date, and time fields.

For a list of fields associated with events in the Access log file, see “[Fields in access logs](#)” on page 100. Not every field is associated with each event in the log file.

See also

[Events](#)

FileIO

Container element.

The elements in this section configure the settings for the File plug-in log file settings.

Contained elements

[Directory](#), [Time](#), [Rotation](#)

FileName

Specifies the name of the Access log file.

The Access log filename includes a date stamp and version number. Y represents the year of its creation; the format YYYY must be used. M represents the month of its creation; the formats M or MM are both allowed. D represents the day of the month of the file’s creation; the formats D or DD are both allowed. N represents the version number of the file. Note that there is no limit on the number of versions.

The repetition of a letter represents the number of digits. For example, M represents 4 (April). MM represents 04 (April).

Example

```
access.2007103043.log
```

This example identifies version 43 of the access log file for October 30, 2007.

See also

[LogServer](#), [Directory](#), [Time](#), [Rotation](#), [Events](#), [Fields](#), [Delimiter](#), [QuoteFields](#), [EscapeFields](#)

History

Specifies the maximum number of log files to keep.

The files are named access.01.log, access.02.log, access.03.log, and so on. The default number of files to retain is 5.

Example

```
<History>5</History>
```

See also

[MaxSize](#), [Schedule](#), [Rename](#)

HostPort

Specifies the IP and port of the log server.

Example

```
<HostPort>xxx.xxx.xxx.xxx:1234</HostPort>
```

See also

[ServerID](#), [DisplayFieldsHeader](#)

Logger

Root element.

The `Logger` element is a container for all the other elements in `Logger.xml`.

LogServer

Container element.

The elements nested in this section configure the server to send messages to a remote log server.

Contained elements

[HostPort](#), [ServerID](#), [DisplayFieldsHeader](#)

master

Container element.

The elements nested within this container configure the master log settings.

Contained elements

[LogServer](#)

MaxSize

Specifies the maximum log file size in bytes. The default file size is 10240 KB, or approximately 1 MB.

Example

```
<Maxsize>10240</MaxSize>
```

See also

[Schedule](#), [History](#), [Rename](#)

QuoteFields

Formatting element. Specifies whether or not to use quotation marks to surround those fields in the log file that include a space.

This element can be set to `enable` or `disable`. By default, it is set to `disable`.

Example

```
<QuoteFields>disable</QuoteFields>
```

See also

[LogServer](#), [Directory](#), [FileName](#), [Time](#), [Rotation](#), [Events](#), [Fields](#), [Delimiter](#), [EscapeFields](#)

Rename

Specifies a new name for log files when rotation occurs. The default is `true`.

If `Rename` is set to `true`, `application.00.log` is renamed `application.01.log`, and `application.01.log` is renamed `application.02.log` (and so on) when it is time to rotate the log files. This occurs until the maximum history setting is reached. The log file with the highest version number keeps the oldest log history.

If `Rename` is set to `false`, a new log file is created with the next available version when rotation occurs. The log file with the lowest version number keeps the oldest log history.

Examples

```
<Rename>true</Rename>
```

See also

[MaxSize](#), [Schedule](#), [History](#)

Rotation

Container element.

The elements in this section configure the rotation of the log files.

Contained elements

[MaxSize](#), [Schedule](#), [History](#), [Rename](#)

Schedule

Specifies the rotation schedule for the log files.

There are two types of scheduling: daily rotation and rotation that occurs when the log exceeds a specified length.

Examples

If the `type` attribute is `daily`, the server rotates the log files every 24 hours.

```
<Schedule type="daily"></Schedule>
```

If the `type` attribute is `hh:mm`, the timestamp `00:00` causes the file to rotate every midnight.

```
<Schedule type="hh:mm"></Schedule>
```

If the `type` attribute is `duration`, rotation occurs when the duration of the log exceeds a specified length. The duration is specified in minutes.

```
<Schedule type="duration"></Schedule>
```

See also

[MaxSize](#), [History](#), [Rename](#)

ServerID

By default, the value of the `ServerID` element is the IP address of the server whose events are being logged.

Example

```
<ServerID>xxx.xxx.xxx.xxx:1234</ServerID>
```

See also

[HostPort](#), [DisplayFieldsHeader](#)

Time

The `Timefield` in a log file can be logged either in UTC (GMT) or local time. Valid values are `utc`, `gmt`, or `local`.

The setting for the `Time` element can be used to override the server-wide configuration. The default is local time.

See also

[Logging](#)

Server.xml file

The `Server.xml` file is located at the root level of the `conf` directory. Edits made in the `Server.xml` file affect the entire server unless they are overridden in another configuration file.

To see the element structure and default values in `Server.xml`, see the `Server.xml` file installed with Adobe Media Server in the `RootInstall/conf/` directory.

Access

Container element.

The elements nested within the `Access` container configure the Access log settings. The Access logs are located in the `RootInstall/logs` directory.

Contained elements

[Enable](#), [Scope](#)

ACCP

Container element.

The elements nested within the `ACCP` container configure the Admin Core Communication Protocol (ACCP). Adobe Media Administration Server and the active cores communicate over ACCP. This protocol is also used for collecting performance metrics and issuing administrative commands to Adobe Media Server cores.

The Adobe Media Administration Console connects to Adobe Media Administration Server, which in turn connects to Adobe Media Server.

Contained elements

[MinIOThreads](#), [MaxIOThreads](#), [SocketTableSize](#), [SocketOverflowBuckets](#)

ActiveProfile

Specifies the limits enforced by the server on each license key (set in the `LicenseInfo` element). Select a profile to determine bandwidth, connection, and other licensed limits.

See also

[Process](#), [Mask](#), [LicenseInfo](#)

AdaptorName

Enable a registry core to activate a single `AMSCore` instance dedicated to handling services for Adobe Media Gateway. This `AMSCore` process comes to life on start-up and allows Adobe Media Gateway to register its services to Adobe Media Server. Enable this registry core only when using Adobe Media Gateway.

```
<Registry enabled="false">  
  <!-- Adaptor to run registry core on -->  
  <AdaptorName>_defaultRoot_</AdaptorName>  
</Registry>
```

See also

[“Registry” on page 222](#)

Admin

Container element.

The elements nested within the `Admin` container configure the RTMP (Real-Time Messaging Protocol) for the `amsadmin` process. RTMP is the protocol used for communication between Flash Player and Adobe Media Server.

Contained elements

[MinIOThreads](#), [MaxIOThreads](#), [SocketTableSize](#), [SocketOverflowBuckets](#)

AdminElem

Specifies the format used to display an element name in an HTTP command. The default value is `false`, which means the element name is displayed as `<_x>`; otherwise the element name is displayed as `<elem name="x">`.

The `getActiveVHost()` command lists only the active virtual hosts. The `getActiveVHostStats()` command allows administrators to query the statistics information for all active virtual hosts. To display `<elem name="x">` instead of `<_x>` in the HTTP command, set the `AdminElem` element to `true`.

Example

```
<AdminElem>true</AdminElem>
```

See also

[Process](#), [Allow](#), [Deny](#), [Order](#)

AdminServer

Container element.

The elements nested within the `AdminServer` container configure Adobe Media Administration Server.

Contained elements

[RTMP](#) ([AdminServer](#)), [HostPort](#), [SocketGC](#), [Process](#), [AdminElem](#), [Allow](#), [Deny](#), [Order](#), [CrossDomainPath](#)

Allow

Specifies the administrator connections that are to be accepted. By default, a client can connect to Adobe Media Administration Server from any domain or IP address. This potential security risk can be managed by the `Allow` element. Permissible administrator connections are detailed as a comma-delimited list of host names, domain names, and full or partial IP addresses. The keyword `all` can also be used.

Example

```
<Allow>x.foo.com, foo.com, 10.60.1.133, 10.60</Allow>
```

or

```
<Allow>all</Allow>
```

See also

[Deny](#), [Order](#)

AllowAnyAACProfile

By default, the server does not stream any AAC audio profiles that are not supported by Flash Player 10. To override this behavior, set this value to `true`.

See also

[Playback](#)

AllowAnyAVCProfile

By default, the server does not stream any H.264 video profiles that are not supported by Flash Player 10. To override this behavior, set this value to `true`.

See also

[Playback](#)

AllowedVideoLag

The number of milliseconds the server holds audio messages in the recording buffer while waiting for a video message. This lag allows the server to sort timestamps before flushing the buffer to disk when video is delayed. The default value is 5.

Application

Container element.

The `Enable` element nested within the `Application` container enables the Application log file.

Contained elements

[Enable](#)

ApplicationGC

Specifies in minutes how often the server checks for and removes unused application instances. The default interval is 5 minutes, which is also the minimum value for this element. An application is considered idle if it has no clients connected for longer than the amount of time specified in `MaxAppIdleTime` in `Application.xml`.

AudioAutoBufferSize

The server caches F4V/MP4 video, audio, and other data in memory. Caching data allows the server to make fewer disk reads and to perform better. Each data type (audio, video, and other) has its own cache. You can tune the size of each cache, depending on the type of content you are delivering, to achieve better disk performance. For example, if you are delivering audio-only content, increase the `AudioAutoBufferSize` buffer, and decrease the `VideoAutoBufferSize` buffer.

The minimum size is 1024 bytes. The default value is 51200 bytes.

See also

[VideoAutoBufferSize](#), [OtherAutoBufferSize](#)

AuthEvent

Container element. The `Enable` element nested within the `AuthEvent` container enables the logging of events from the Authorization plug-in.

Contained elements

[Enable](#)

AuthMessage

Container element. The `Enable` element nested within the `AuthMessage` container enables the logging of messages from the Authorization plug-in.

Contained elements

[Enable](#)

AutoCloseIdleClients

Container element. Determines whether or not to automatically close idle clients.

Set the `enable` attribute to `true` to close idle clients. If the `enable` attribute is omitted or set to `false`, the feature is disabled. The default value is `false`.

A client is active when it is sending or receiving data. Use `AutoCloseIdleClients` to specify how often the server should check for idle clients. When a client has been idle longer than the maximum idle time (60 seconds by default), the server sends a status message to the `NetConnection` object (the client). The server closes the client connection to the server and writes a message to the access log. The server also writes a message such as “Client x has been idle for y seconds” in the core and event logs.

To configure the closing of idle connections, you must enable the feature in the `Server.xml` file. Once you enable the feature in the `Server.xml` file, you can disable the feature for individual virtual hosts in the `Vhost.xml` files or for individual applications in `Application.xml`. The values defined in the `Server.xml` configuration file apply to all clients connected to the server, unless the values are defined in the `Vhost.xml` file. The `Vhost.xml` values override the `Server.xml` values. (Subsequently, the values defined in the `Vhost.xml` configuration file apply to all clients connected to the virtual host, unless values are defined in the `Application.xml` file. The `Application.xml` values override the `Vhost.xml` values.)

Example

```
<AutoCloseIdleClients enable="false">
  <CheckInterval>60</CheckInterval>
  <MaxIdleTime>600</MaxIdleTime>
</AutoCloseIdleClients>
```

Contained elements

[CheckInterval](#), [MaxIdleTime](#)

Cache

Container element. Contains elements that configure the cache setting for SWF verification.

See also

[TTL](#), [UpdateInterval](#) (Cache)

CheckInterval

Specifies the interval, in seconds, at which the server checks for active client connections. The minimum and default value is 60 seconds.

A client is disconnected the first time the server checks for idle connections if the client has exceeded the `MaxIdleTime` value. A shorter interval results in more reliable disconnection times, but can also result in decreased server performance.

Example

```
<CheckInterval>60</CheckInterval>
```

See also

[Checkpoints](#)

Checkpoints

Enables logging checkpoint events. Checkpoint events log bytes periodically from the start to the end of an event. The following are available as checkpoint events: `connect-continue`, `play-continue`, and `publish-continue`.

This element contains the `enable` attribute which you can set to `true` or `false`. Set the `enable` attribute to `true` to turn on checkpoint events in logs. The default value is `false`.

You must enable checkpoint events at the server level in the `Server.xml` file. You can disable checkpoints at the `vhost` and application level in the `Vhost.xml` and `Application.xml` files. You can also override the logging interval at the `vhost` and application levels.

Contained elements

[CheckInterval](#), [LogInterval](#)

ConnectionTimeout

Flash Media Server 4.5

The `/RTMP/ConnectionTimeout` element and `/HTTP/ConnectionTimeout` element specify how long to wait before timing out an outgoing connection, in milliseconds. The default is `-1` which uses the OS timeout to block outgoing connections.

```
<ConnectionTimeout>-1</ConnectionTimeout>
```

See also

[“HTTP” on page 207](#)

[“RTMP” on page 137](#)

Connector

Container element.

The elements nested within the `Connector` container configure the connector subsystem. Adobe Media Server provides connectors that allow application scripts to connect to other Adobe Media Servers or HTTP servers.

Contained elements

[HTTP](#), [RTMP](#) ([Connector](#))

Content

Enables operations in the current File plug-in on content (streams or SWF files). The value of `true` enables the operation on the specified content type and `false` disables it.

This element contains the `type` attribute, which you can set to the following values:

- `Streams`. Enables stream file operations. Enabled by default.
- `SWF`. Enables operations on SWF files. Disabled by default. Enable if you plan to use the File plug-in to retrieve SWF files for verification by the server.

This element is nested within the `FilePlugin` tag.

Example

```
<Content type="Streams">true</Content>  
<Content type="SWF">false</Content>
```

See also

[FilePlugin](#)

Core

Container element.

The elements nested within the `Core` container configure the RTMP protocol for the `AMSCore.exe` process.

Contained elements

[MinIOThreads](#), [MaxIOThreads](#), [SocketTableSize](#), [SocketOverflowBuckets](#)

CoreExitDelay

Specifies how much wait time, in seconds, an idle core is given to exit on its own before it is removed from the server. The default wait time is 20 seconds.

Example

```
<CoreExitDelay>60</CoreExitDelay>
```

See also

[CoreGC](#)

CoreGC

Specifies how often, in seconds, to check for and remove idle or unused cores. The default is 300 seconds.

Example

```
<CoreGC>300</CoreGC>
```

See also

[CoreExitDelay](#)

CoreTimeout

Specifies the timeout value, in seconds, for detecting unresponsive cores. The default timeout is 30 seconds. A value of 0 disables the timeout check.

Example

```
<CoreTimeout>30</CoreTimeout>
```

See also

[CoreGC](#)

CPUMonitor

Specifies, in seconds, how often the server monitors CPU usage. The default interval is 1 second. The value cannot be set to less than 1 second.

Example

```
<CPUMonitor>1</CPUMonitor>
```

See also

[ResourceLimits](#)

CrossDomainPath

Specifies an absolute path on the server to a cross-domain file. This element lets you specify a list of domains from which a client can access the server. The default value is *, which allows clients from all domains to access the server.

This element is nested within both the `Server` element and the `AdminServer` element. Use it for both edge and administration server requests.

Example

```
<CrossDomainPath>C:/Security/config/files/ams/crossdomain.xml</CrossDomainPath>
```

Deny

Specifies administrator connections that should be ignored. The connections are specified as a comma-delimited list of host names, domain names, and full or partial IP addresses, or the keyword `all`.

Example

```
<Deny>x.foo.com, foo.com, 10.60.1.133, 10.60</Deny>
```

or

```
<Deny>all</Deny>
```

See also

[Allow, Order](#)

Diagnostic

Container element.

The `Enable` element nested within the `Diagnostic` section enables the diagnostic log file.

Contained elements

[Enable](#)

DiffServ

Flash Media Server 3.5.2

Use this element with the `DiffServMask` element to specify the `DiffServ` field for all sockets that connect to the RTMP listener (these include tunneling and HTTP proxy sockets). There is little use for this unless you have set up a DSCP domain. Also, some values may lead to errors, depending on your platform and your router configuration. For example, to set priority class 4, you might set `DiffServ` to 128 (top 3 bits set to 4) and set `DiffServMask` to 224 (top 3 bits on).

To verify that bits are set on a socket, call the Administration API `getNetStreamStats()` method. Each stream has a `diffserv_bits` field whose value is updated when you successfully set bits on a socket.

DiffServMask

Flash Media Server 3.5.2

Use this element with the `DiffServ` element to specify the DiffServ field for all sockets that connect to the RTMP listener (these include tunneling and HTTP proxy sockets). There is little use for this unless you have set up a DSCP domain. Also, some values may lead to errors, depending on your platform and your router configuration. For example, to set priority class 4, you might set `DiffServ` to 128 (top 3 bits set to 4) and set `DiffServMask` to 224 (top 3 bits on).

Directory

Located in the `Httpd` container.

The directory containing the web server. The default value is `Apache2.2`.

DirLevelSWFScan

Specifies the number of levels of subfolders within a parent folder to scan for SWF files. The parent folder is specified in the `SWFFolder` element.

Specifying a positive value scans that number of subfolders. Specifying zero scans the parent folder and no subfolders. Specifying a negative value scans all subfolders. The default value is 1, which means that the server scans only one subfolder level.

See also

[SWFFolder](#)

ECCP

Container element.

The elements nested within the `ECCP` container configure ECCP (Edge Server-Core Server Communication Protocol). Adobe Media Server edge processes and Adobe Media Server core processes use ECCP to migrate socket connections and to proxy connections that have not been migrated.

Contained elements

[MinIOThreads](#), [MaxIOThreads](#), [SocketTableSize](#), [SocketOverflowBuckets](#), [CoreTimeout](#)

Edge

Container element.

The elements nested within the `Edge` container configure the RTMP protocol for the `amsedge` process.

Contained elements

[MinIOThreads](#), [MaxIOThreads](#), [SocketTableSize](#), [SocketOverflowBuckets](#)

Edge (ResourceLimits)

Container element.

The elements nested within the `Edge` container configure resources for the `amsedge` process.

Contained elements

[MaxEventQueueThreads](#), [MinEventQueueThreads](#), [NumSchedQueues](#)

EdgeCore

Container element.

The elements nested within the `EdgeCore` container control the IPC (interprocess communication) message queue used by edge and core processes to communicate with each other.

Contained elements

[HeapSize](#), [MaxQueueSize](#)

Enable

`Server.xml` uses `Enable` elements in the `Logging` container to enable or disable the `Access`, `Diagnostic`, `Application`, `AuthEvent`, `AuthMessage`, and `FileIO` logs. A value of `true` enables the logging process; `false` disables the logging process. The default value is `true`.

Example

```
<Access>
  <enable>true</Enable>
</Access>
```

See also

[Access](#), [Diagnostic](#), [Application](#), [AuthEvent](#), [AuthMessage](#), [FileIO](#)

EnableAggMsgs (playback)

Located in the `Playback` container.

Aggregating messages increases server performance. The default value is `true`. Do not change this value if you want to achieve the highest possible server performance.

Note: This setting applies to playback of MP4/F4V files only.

EnableAggMsgs (raw)

Specifies whether the RAW adaptor generates aggregate messages (`true`) or not (`false`). Aggregating messages improves server performance.

The default value is `true`.

Example

```
<EnableAggMsgs>true</EnableAggMsgs>
```

See also

[Raw](#)

EnableSystemLogging

Enables logging of high severity entries to the system log (`true`) or not (`false`).

The default value is true.

Example

```
<EnableSystemLogging>true</EnableSystemLogging>
```

FileCheckInterval

Located in the [FLVCache](#) container.

Specifies, in seconds, how often the server reloads the video segment in the cache when there is a file change. The default value is 120 seconds. The minimum value is 1 second. There is no maximum value. A very large number means the server does not refresh the cache even when there is a file change.

Example

```
<FileCheckInterval>120</FileCheckInterval>
```

FileIO

Container element.

The `Enable` element nested within the `FileIO` container enables logging from the File plug-in.

Contained elements

[Enable](#)

FilePlugin

Container element.

This element contains elements to configure file operations that are handled by the File plug-in. Setting the `enabled` attribute to `false` disables the File plug-in.

Contained elements

[MaxNumberOfRequests](#), [Content](#)

FLVCache

Container element.

Contains elements that control the size and features of the recorded media cache.

Contained elements

[FileCheckInterval](#), [MaxSize \(FLVCache\)](#), [MaxKeyframeCacheSize](#)

FLVCachePurge

Located in the [ResourceLimits](#) container.

Specifies how often, in minutes, to remove unused files from the cache. The default value is 60 minutes. The minimum value is 1 minute. If you specify a value of less than 1 minute, the default value of 60 is used.

FLVCacheSize

Located in the [ResourceLimits](#) container.

Specifies the maximum size of the recorded media cache. The recorded media cache size is specified as a percentage of the total available RAM on the system. The default setting for the cache size is 10 (10%). The maximum setting is 100 (100%), in which case virtual memory will also be used.

Use this setting to configure the cache for optimal memory use. If you are receiving “cache full” events in the core log file or want to increase the chance that streams will find the information needed in the cache, increase the size of the cache. To minimize the amount of memory used in the server process, decrease the size of the cache.

Example

```
<FLVCacheSize>10</FLVCacheSize>
```

FreeMemRatio

Located in the [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), and [SegmentsPool](#) containers.

Specifies the maximum percentage of total memory that the total pool size may occupy. The range of this setting is from 0 (0%) to 1 (100%). The default setting is 0.5 (50%).

Example

```
<FreeMemRatio>0.5</FreeMemRatio>
```

FreeRatio

Located in the [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), and [SegmentsPool](#) containers.

Specifies the percentage of the message cache to be consumed by the free list on a per-thread basis. The range of this setting is from 0 (0%) to 1 (100%). The default setting is 0.125 (12.5%).

When more free memory is available to a thread than the specified ratio, the freed memory returns to the global pool.

Example

```
<FreeRatio>0.125</FreeRatio>
```

GCInterval

Located in the [HandleCache](#) container.

Specifies in minutes how often to remove idle handles. The default is 60 minutes.

Example

```
<GCInterval>60</GCInterval>
```

GID

Located in the [Process](#) containers.

Specifies the group ID of the process. If you do not specify a `UID` or `GID`, the server or Administration Server runs as root. This element is applicable to Adobe Media Server running on Linux systems only.

See also

[UID](#)

GlobalQueue

Container element.

The elements nested within the `GlobalQueue` container control the IPC message queue used by all processes to communicate with each other.

Contained elements

[HeapSize](#), [MaxQueueSize](#)

GlobalRatio

Located in the [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), and [SegmentsPool](#) containers.

Specifies the percentage of the message cache to be consumed by the free list on a global basis. When more free memory is available to a thread than the specified ratio, the freed memory returns to the operating system.

The range of this setting is from 0 (0%) to 1 (100%). The default setting is 0.4 (40%).

Example

```
<GlobalRatio>0.4</GlobalRatio>
```

HandleCache

Container element.

Contains elements that configure how to cache handles used for HTTP requests. This section is used for connections to Flash Remoting.

Contained elements

[MaxSize \(HandleCache\)](#), [IdleTime](#), [TrimSize](#), [GCInterval](#)

HeapSize

Specifies the maximum size, in kilobytes, of the shared memory heap used for an IPC (interprocess communication) message queue. The default value for this element varies according to its container.

Container	Default Value	Description
EdgeCore	1024	If the maximum size of this element is not specified, the value is 100 KB.
GlobalQueue	2048	
Services	2048	

Example

```
<EdgeCore>  
  <HeapSize>1024</HeapSize>  
</EdgeCore>
```

See also

[MaxQueueSize](#)

HostPort

Specifies the IP address and port number that Adobe Media Administration Server binds to. The default is to bind to any available IP on port 1111. Only one port number may be specified in this element.

The Administration Service is separate from Adobe Media Server. When administrators connect to the server with the Administration Console, they are connecting to Adobe Media Administration Server, which in turn connects to Adobe Media Server.

Example

```
<HostPort>ip:port</HostPort>
```

See also

[AdminServer](#)

HTTP

Container element.

The elements nested within the HTTP container configure the HTTP connector used to connect to Flash Remoting.

The following reference table gives the default values for all thread configurations.

Default Value	Description
0	Allocates the default number of threads.
>0	Allocates the exact number of threads specified.
-1	Allocates 1xN threads, where N is the number of processors available in the computer.
-2	Allocates 2xN threads, where N is the number of processors available in the computer. For example, suppose you have a quad-core computer, where applications see 4 local processors. If you specify a value of -2, the HTTP connector can use up to 8 threads.

Contained elements

[MinConnectionThreads](#), [MaxConnectionThreads](#), [MaxConnectionQueueSize](#), [HandleCache](#)

Httpd

Specifies how the built-in webserver is stopped and started. By default the `enabled` attribute is set to "`{SERVER.HTTPD_ENABLED}`". By default, the `SERVER.HTTPD_ENABLED` variable is set to `true` in the `ams.ini` file. When `enabled` is set to `true`, Adobe Media Server starts and stops the web server automatically. Adobe Media Server uses the settings in the `Directory`, `Program`, `Options`, and `Service` elements to execute the following command lines:

```
$Directory/$Program -d $Directory $Options -k start  
$Directory/$Program -d $Directory $Options -k stop
```

Note: The value of `$Directory` can be absolute or relative to `$AMSROOT`.

On Windows only, `Service` specifies the name used to install the webserver as an NT service. If a value for `Service` is present, `-n $Service` is appended to both commands. Linux ignores this section of the command.

Contained elements

[Directory](#), [Program](#), [Options](#), [Service](#)

IdleTime

Located in the [HandleCache](#) container.

Specifies the amount of time to wait before releasing cached handles. If no HTTP requests have been made to the host for the length of time specified, some cached handles are cleared. The default wait time is 10 minutes.

Example

```
<IdleTime>10</IdleTime>
```

IpcHostPort

The interface and port the server listens on internally for connections from the amscore process to the Administration Server. The default value is `localhost:11110`.

IPCQueues

Container element.

The elements nested within the `IPCQueues` container configure the IPC queues. Adobe Media Server uses IPC queues to send messages from one core to another or from one process to another, such as master to core, or core to edge.

Unlike protocols, queues are used for short or one-time messages that may have more than one target.

Contained elements

[GlobalQueue](#), [EdgeCore](#), [Services](#)

LargeMemPool

Container element.

The elements nested within the `LargeMemPool` container configure the large memory pool, which caches large chunks of memory within the server to increase the performance of large allocations.

Contained elements

[MaxCacheUnits](#), [MaxCacheSize](#), [MaxUnitSize](#), [FreeRatio](#), [GlobalRatio](#), [MaxAge](#), [UpdateInterval](#), [FreeMemRatio](#)

LicenseInfo

The license number. By default, this value is a parameter in the `ams.ini` file:

```
<LicenseInfo>${SERVER.LICENSEINFO}</LicenseInfo>
```

Note: Serial numbers that are added manually (that is, added by editing those files directly) to either `ams.ini` or the `LicenseInfo` tag of `Server.xml` file cannot be removed using the Administration Console. Only serial numbers that are added using the Administration Console can be deleted using the Administration Console.

Example

```
<LicenseInfo>${SERVER.LICENSEINFO}</LicenseInfo>
```

LicenseInfoEx

Contains license keys added using the Administration Console.

Localhost

Specifies the IP loopback address.

The server must reference itself locally. The IP loopback address is usually the default localhost address. With more than one network interface, localhost can map to an erroneous interface. The server uses the default loopback address as the local loopback.

LockoutLimit

The maximum number of login failures before a user is prevented from logging into the admin server. If this value is reached, the admin server must be restarted before that user can log in again.

The default value is 300.

Example

```
<LockoutLimit>300</LockoutLimit>
```

See also

[LogInLimits](#), [MaxFailures](#), [RecoveryTime](#)

LogInLimits

Container element.

Configures the number of failed admin login attempts that are allowed, and the amount of recovery time required before the admin can attempt to log in again.

Contained Elements

[LockoutLimit](#), [MaxFailures](#), [RecoveryTime](#)

See also

[Security](#)

LogInterval

Specifies how often to log a checkpoint, in seconds. This value should be larger than the value for `CheckInterval`. If the value is smaller, the server logs a checkpoint every check interval. The default value is 3600 seconds (60 minutes).

See also

[Checkpoints](#)

Logging

Container element.

The elements nested within the `Logging` container configure general logging properties for the server. Set the configuration properties of the individual log files in the `Logger.xml` file.

Log files are written in English. Field names in the log file are in English. Some content within the log file, however, may be in another language, depending on the filename and the operating system. For example, in the `access.log` file, the columns `x-sname` and `x-suri-stem` show the name of the stream. If the name of the recorded stream is in a language other than English, the name is written in the log file in that language, even if the server is running on an English-language operating system.

Contained elements

[Time](#), [RecordsNumber](#), [RetryNumber](#), [RetrySleep](#), [Access](#), [Diagnostic](#), [Application](#), [AuthEvent](#), [AuthMessage](#), [FileIO](#)

Mask

A three-digit octal value used by the Linux `umask` (user permissions mask) command to set a file creation mask. The user must enter the mask in a three-digit octal format.

The default setting for this element is 017 in octal.

This element is applicable to Adobe Media Server running on Linux systems only. This element controls who has read/write access to shared object and stream files on the server. All Adobe Media Server object files, such as stream files or shared object files, are created on the server side with permission 0666. This key is used by `umask` to set the file creation mask. By default, the creation mask is set to 017 in octal. Therefore, all Adobe Media Server object files are created with permission $0666 \& \sim 017 = 0660 = rw-rw----$.

The owner and the users who belong to the same group as the owner get read/write permission to the files. If the mask is set to 022, the file created is assigned permission $0666 \& \sim 022 = 0644 = rw-r--r--$.

Master

Container element.

The elements nested within the `Master` container configure the resource limits for the master server.

Contained elements

[CoreGC](#), [CoreExitDelay](#)

MaxAge

Located in the [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), and [SegmentsPool](#) containers.

This element defines the maximum reuse count before the cache unit is freed. The default count is 1,000,000.

Example

```
<MaxAge>1000000</MaxAge>
```

MaxAggMsgSize (playback)

The maximum size of an aggregate message, in kilobytes. The default value is 64.

Note: This setting applies to playback of MP4/F4V files only.

See also

[Playback](#)

MaxAggMsgSize (raw)

The maximum size of an aggregate message, in bytes. You can use any positive integer.

The default value is 65536.

Example

```
<MaxAggMsgSize>65536</MaxAggMsgSize>
```

See also

[Raw](#)

MaxAudioSampleDescriptions

Located in the [Recording](#) container.

Each change in codec for a content type has a sample description. For example, two different audio codecs each have their own sample description. The server allocates space for sample descriptions when it creates a file. If the codec type changes more than the number of available descriptions, the server stops recording. However, adding too many descriptions takes up unnecessary space for every file that the server records.

The default value is 20.

Note: This setting applies to recording MP4/F4V files only.

MaxCacheSize

Located in the [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), and [SegmentsPool](#) containers.

This element defines the maximum size of the cache in megabytes. The default is 100 MB.

Example

```
<MaxCacheSize>100</MaxCacheSize>
```

MaxCacheUnits

Located in the [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), and [SegmentsPool](#) containers.

This element defines the maximum number of free units in the cache. Keep in mind that the number of free units may be less than maximum if the value of the `MaxCacheSize` limit is reached.

The default is 4096 units.

Example

```
<MaxCacheUnits>4096</MaxCacheUnits>
```

MaxConnectionQueueSize

Located in the [HTTP](#) container.

Specifies the maximum number of connection requests that can be pending. Connection requests are rejected if this limit is exceeded.

The default number of pending requests is 1000. To use the default, specify -1.

Example

```
<MaxConnectionQueueSize>-1</MaxConnectionQueueSize>
```

MaxConnectionRate

Located in the [RTMP \(Protocol\)](#) and [Edge](#) containers.

Specifies the maximum number of incoming connections per second that the server's socket listener accepts. You can set a fractional maximum connection rate, such as 12.5. A value of 0 or -1 disables the feature.

This is a global setting for all socket listeners. If the element is set to 10 connections per second, each listener has a limit of 10 connections per second. If there are three listeners and the `MaxConnectionRate` is set to 10, the server imposes a maximum total combined rate of 30 connections per second. The socket listeners are configured in the `Adaptor.xml` configuration file using the `HostPort` element under the `HostPortList` container element.

Connections requested at a rate above the value specified in this element remain in the TCP/IP socket queue and are silently discarded by the operating system whenever the queue becomes too long.

Example

```
<MaxConnectionRate>100</MaxConnectionRate>
```

MaxConnectionThreads

Located in the [HTTP](#) and [RTMP \(Connector\)](#) containers.

Specifies the maximum number of threads used to process connection requests. For HTTP, the default number is 10. To use the default, specify 0. For RTMP, the default number is 5. To use the default, specify 0. If the server is taking a long time to process connections, raise the value of `HTTP/MaxConnectionThreads` to 20.

MaxDataSampleDescriptions

Located in the [Recording](#) container.

Each change in codec for a content type has a sample description. For example, two different data encoding formats each have their own sample description. The server allocates space for sample descriptions when it creates a file. If the encoding format type changes more than the number of available descriptions, the server stops recording. However, adding too many descriptions takes up unnecessary space for every file that the server records. The default value is 10.

Note: This setting applies to recording MP4/F4V files only.

MaxELSTEntries

Located in the [Recording](#) container.

The maximum number of ELST entries in a recording. The server uses ELST entries when there are gaps in content. Gaps occur during an append to the file or when video, audio, or data content ends while other such content continues. If more gaps or appends occur than the value specified in the `MaxELSTEntries` tag, recording ends. However, setting this value too high takes up unnecessary space in each recorded file. The default value is 100.

Note: This setting applies to recording MP4/F4V files only.

MaxEventQueueThreads

Located in the [Edge \(ResourceLimits\)](#) container.

The maximum number of threads per scheduler queue. The default value is -2.

MaxFailures

The maximum number of login attempts a user can have before having to wait to be allowed again. The wait time is defined by the `RecoveryTime` element.

The default value is 3. If the number of failed login attempts equals the value of `LockoutLimit`, the admin server must be restarted before that user can attempt to log in again.

Example

```
<MaxFailures>3</MaxFailures>
```

See also

[LogInLimits](#), [LockoutLimit](#), [RecoveryTime](#)

MaxFlushSize

Located in the [RecBuffer](#) container.

The maximum number of kilobytes to accumulate before flushing the recording buffer. The default value is 256. The minimum value is 32.

MaxFlushTime

Located in the [RecBuffer](#) container.

The maximum number of seconds to wait before flushing the recording buffer. The default value is 5. The minimum value is 1.

MaxIdleTime

Located in the [AutoCloseIdleClients](#) container.

Specifies the maximum idle time allowed, in seconds, before a client is disconnected. The default value is 3600 seconds (60 minutes). If you set a value of 60 or less, the server uses 1200 seconds (20 minutes).

A low value may cause unneeded disconnections. When you configure this element, consider the type of applications running on the server. For example, if you have an application with which users watch short video clips, a user might leave the window to idle out.

MaxInitDelay

The maximum number of seconds that is used to process SWF files. The default value is 5 seconds. This element is used in the verification of SWF files.

Example

```
<MaxInitDelay>5</MaxInitDelay>
```

See also

[SWFVerification](#)

MaxIOThreads

Located in the [RTMP \(Connector\)](#), [Edge](#), [Core](#), [Admin](#), [ECCP](#), and [ACCP](#) containers.

Specifies the maximum number of threads that can be created for I/O processing.

Use the following information to configure all I/O and connection threads processing:

- A value of 0 allocates the default number of threads (10).
- A value greater than 0 allocates the exact number of threads specified.
- A value less than 0 ties the number of connection threads to the number (N) of processors, as follows:
 - -1 means 1 x N threads.
 - -2 means 2 x N threads, and so on.

Adobe Media Server can receive connections through various protocols. The default value for this element varies according to which container protocol it is nested within.

Container	Default Value	Description
ACCP	10	Use 0 for the default value.
Admin	10	Use 0 for the default value.
Core	10	Use 0 for the default value.
ECCP	10	Use 0 for the default value.
Edge	10	Use 0 for the default value.
RTMP	32	Use -1 for the default value.

Example

```
<RTMP>  
  <MaxIOThreads>32</MaxIOThreads>  
</RTMP>
```

MaxKeyframeCacheSize

Located in the [FLVCache](#) container.

Specifies the maximum number of keyframes in the cache for each recorded media file. The default value is 2000 keyframes.

When enhanced seeking is enabled, the server generates keyframes that are saved in the cache. (For more information, see [EnhancedSeek](#) in the Application.xml file.) If you lower `MaxKeyframeCacheSize`, the cache uses less memory. If an application uses many large recorded media files, you may want to lower this number.

Example

```
<MaxKeyframeCacheSize>0</MaxKeyframeCacheSize>
```

MaxNumberOfRequests

Specifies the maximum number of pending requests to the File plug-in. The default value is 0, which allows unlimited requests.

If the plug-in does not respond to a request from the server, the number of pending requests increases up to the value of `MaxNumberOfRequests`. Once `MaxNumberOfRequests` is reached, subsequent requests are rejected.

By default, Adobe Media Server does not count the number of requests made to the File plug-in or responses returned from the File plug-in. If necessary, you can use this value to identify problems with the File plug-in. Adobe generally recommends keeping the default value of 0.

MaxQueueSize

Located in the `GlobalQueue`, `EdgeCore`, `Services` containers.

Specifies the maximum number of pending IPC messages in the queue. When messages are sent to a process that is not running, the message can be put in a pending queue for the process. When the process starts again, it picks up the messages. Use the `MaxQueueSize` element to limit the number of messages left in the pending queue. Limiting messages saves shared memory if the process never starts. The value is specified in kilobytes. The default size is 100 KB.

MaxSampleSize

The maximum allowable size of a sample in an F4V/MP4 file, in kilobytes. You can increase this value up to 4GB if a file has larger samples. The default value is 16777216 (16 MB).

See also

[Playback](#)

MaxSize (FLVCache)

Located in the `FLVCache` container.

Specifies the maximum size of the recorded media cache in megabytes. The default value is 500 MB. This value shares memory with the running process and has a limit of 2 GB in Windows and 3 GB in Linux on 32-bit configurations. On 64-bit configurations, the file memory available to the process is only limited by the amount of RAM.

The size of the cache limits the number of unique streams the server can publish. To increase the probability that a requested stream will be located in the recorded media cache, increase the value of `MaxSize`. To decrease the amount of memory the server process uses, decrease the value of `MaxSize`. Note that increasing the file cache size to too large of a size can cause substantially deteriorated system performance.

MaxSize (HandleCache)

Located in the `HandleCache` container.

Specifies the maximum number of handles to cache. The minimum value is 0, which means that no handles are cached. There is no maximum value; it can be the maximum number of handles that the operating system can support. The default value is 100 handles.

Example

```
<MaxSize>100</MaxSize>
```

MaxSize (RecBuffer)

Located in the `RecBuffer` container.

Specifies the maximum size to which the buffer can grow before messages are committed to file. The default value is 200 and the minimum value is 0. The higher the value, the longer the data will be held in the buffer before written to disk.

When you add DVR functionality to an application, the size of the recording buffer affects the play-while-record experience. A shorter buffer decreases latency between real-time and playback, but hurts performance.

Example

```
<MaxSize>5120</MaxSize>
```

MaxTimestampSkew

Located in the [RecBuffer](#) container.

Specifies the maximum gap in milliseconds between two adjacent messages when comparing the message timestamps with the real time. The server logs a warning when the timestamps between two adjacent messages are bigger than the difference in real time plus the value set here for `MaxTimestampSkew`. This element is disabled by default. To enable the element, set the value to a positive number.

Example

```
<MaxTimestampSkew>2</MaxTimestampSkew>
```

MaxTracks

Located in the [Playback](#) container.

The maximum number of tracks allowed in a file. While only one video, audio, and data track may be played, files with up to this many tracks can be parsed. More than this amount is a file error. The default value is 64.

MaxUnitSize

Located in the [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), and [SegmentsPool](#) containers.

Specifies the maximum size, in kilobytes, of a memory chunk allowed in a memory pool. The default size is 16 KB.

Adobe Media Server has several memory pools (distinct from the recorded media cache) that hold memory in chunks. This setting ensures that chunks larger than `MaxUnitSize` are released to system memory instead of being held in the pool so that large memory chunks are available.

For example, if this tag is under the `MessageCache` tag, the server doesn't cache any messages greater than `MaxUnitSize`.

Example

```
<MessageCache>  
  <MaxUnitSize>16</MaxUnitSize>  
</MessageCache>
```

MaxUrlLength

Defines the maximum allowed RTMP and SWF file URL lengths, in bytes. Most modern browsers support up to 64KB URLs. The default value is 32KB. The maximum allowable value is 1000KB.

Use this element to restrict the URL lengths of incoming requests. To restrict the length of outbound `NetConnection` URLs from SSAS, use the `MaxUrlLength` element in the `Application.xml` file.

Example

```
<MaxUrlLength>65536</MaxUrlLength>
```

See also

[Security](#)

MaxVideoSampleDescriptions

Located in the [Recording](#) container.

Each change in codec for a content type has a sample description. For example, two different video codecs each have their own sample description. The server allocates space for sample descriptions when it creates a file. If the codec type changes more than the number of available descriptions, the server stops recording. However, adding too many descriptions takes up unnecessary space for every file that the records. The default value is 10.

Note: This setting applies to recording MP4/F4V files only.

MessageCache

Container element.

The elements nested within the `MessageCache` container control how the message cache holds onto messages used by the system running Adobe Media Server and keeps them in memory for reuse instead of returning them and requesting them from the operating system.

Messages are the essential communication units of Adobe Media Server. Recycling them improves the server's performance.

Contained elements

[MaxCacheUnits](#), [MaxCacheSize](#), [MaxUnitSize](#), [FreeRatio](#), [GlobalRatio](#), [MaxAge](#), [UpdateInterval](#), [FreeMemRatio](#)

MinConnectionThreads

Located in the [HTTP](#) and [RTMP \(Connector\)](#) containers.

Specifies the minimum number of threads in the pool for I/O operations. The default is 1 multiplied by the number of processors. To use the default, specify the value 0.

Example

```
<MinConnectionThreads>0</MinConnectionThreads>
```

MinEventQueueThreads

Located in the [Edge \(ResourceLimits\)](#) container.

The minimum number of threads per scheduler queue. The default value is 2.

MinGoodVersion

Located in the [SWFVerification](#) container.

Specifies the minimum accepted version of SWF verification feature that the server allows. The default value of 0 accepts current and all future versions.

Example

```
<MinGoodVersion>0</MinGoodVersion>
```

MinIOThreads

Located in the [RTMP \(Connector\)](#), [Edge](#), [Core](#), [Admin](#), [ECCP](#), and [ACCP](#) containers.

The element specifies the minimum number of threads that can be created for I/O operations.

Adobe Media Server can receive connections through various protocols. The default value for this element varies according to which container protocol it is nested within.

Container	Default Value	Description
ACCP	2X number of processors	Use 0 for the default value.
Admin	2X number of processors	Use 0 for the default value.
Core	2X number of processors	Use 0 for the default value.
ECCP	2X number of processors	Use 0 for the default value.
Edge	2X number of processors	Use 0 for the default value.
RTMP	2X number of processors	Use -1 for the default value.

Example

```
<ECCP>
  <MinIOThreads>0</MinIOThreads>
</ECCP>
```

Mp4

Container element.

The `Mp4` elements configure how the server plays and records MP4, F4V, MOV, and other MP4 file types.

Contained elements

[Playback](#), [Recording](#),

MsgPoolGC

Specifies how often the server checks for content in and removes content from the global message pool.

The default interval for checking and removing content is 60 seconds.

NetworkingIPv6

Enables or disables Internet Protocol version 6 (IPv6). This is an empty tag.

The operating system network stack should be configured to support IPv6, if desired. Adobe Media Server automatically detects the operating system configuration; this element can force Adobe Media Server to use IPv4 even if IPv6 is available.

Example

```
<NetworkingIPv6 enable="false" />
```

NumCRThreads

Located in the [RTMP \(Connector\)](#) container.

Specifies the number of completion routine threads in Windows 32-bit systems for edge server I/O processing.

Example

```
<NumCRThreads>0</NumCRThreads>
```

NumSchedQueues

Located in the [Edge \(ResourceLimits\)](#) container.

The number of scheduler queues. The default value is 1.

Options

Located in the [Httpd](#) container.

Specifies additional switches for the command-line argument that starts and stops the web server.

Order

Located in the [AdminServer](#) container.

Specifies the order in which to evaluate the `Allow` and `Deny` elements.

Example

To process the request if not in `<Deny>` or in `<Allow>`, set:

```
<Order>Deny, Allow</Order>
```

To process the request if in `<Allow>` and not in `<Deny>`, set:

```
<Order>Allow, Deny</Order>
```

OtherAutoBufferReadSize

Located in the [Playback](#) container.

The server caches F4V/MP4 video, audio, and other data in memory. Caching data allows the server to make fewer disk reads and to perform better. Each data type (audio, video, and other) has its own cache. You can tune the size of each cache, depending on the type of content you are delivering, to achieve better disk performance.

The minimum size is 1024 bytes. The default value is 1024 bytes.

Playback

Container element.

The `Playback` elements configure how the server plays back MP4, F4V, MOV, and other MP4 file types.

Contained elements

[VideoAutoBufferReadSize](#), [AudioAutoBufferReadSize](#), [OtherAutoBufferReadSize](#), [EnableAggMsgs \(playback\)](#), [MaxAggMsgSize \(playback\)](#), [MaxTracks](#), [MaxSampleSize](#), [AllowAnyAACProfile](#), [AllowAnyAVCProfile](#)

Plugins

Container element.

The elements nested within the `Plugins` element configure plug-ins.

Contained elements

[FilePlugin](#), [UserDefined](#)

Process

Container element.

There are 2 `Process` elements, one for Adobe Media Server and one for Adobe Media Administration Server. The elements nested within the `Process` element contain the `UID` and `GID` elements for all server processes. The elements are applicable to servers running on Linux systems only. For security reasons, on Linux, all the server processes switch from root to the `UID` and `GID`. If you do not specify a value for `UID` and `GID`, the server runs as root.

Contained elements

[UID](#), [GID](#)

ProcVMSizeMonitorInterval

The monitoring interval for process virtual memory use, in seconds. The default value is 60.

ProcVMSizeNominal

The log notification threshold for reverting to a nominal process virtual memory use, in megabytes, after the `ProcVMSizeWarn` threshold has been triggered. The default value is 1600.

ProcVMSizeWarn

The log warning threshold for high process virtual memory use, in megabytes. The default value is 1800.

Program

Located in the [Httpd](#) container.

The location of the executable file for the web server. The default value is `bin/httpd`.

Protocol

Container element.

Adobe Media Server receives connections through various protocols. The elements in this container configure those protocols and how the connection requests are received.

To set the values of all I/O and connection threads processing, follow these guidelines:

- A value of 0 allocates the default number of threads (10).
- A value greater than 0 allocates the exact number of threads specified.
- A value less than 0 ties the number of connection threads to the number (N) of processors:
 - -1 means 1 x N threads
 - -2 means 2 x N threads, etc.

Contained elements

[ACCP](#), [ECCP](#), [RTMP](#) ([Protocol](#))

PublicIP

Specifies that if the system has more than two network ports, use this element to create a public IP address.

Raw

Contains elements that control the RAW adaptor.

Adobe Media Server supports a RAW (Record and Watch) file format that records media into configurable chunks that can be streamed to any version of Flash Player. The RAW file format enables you to serve long-length, multi-bitrate DVR streams without running into performance issues. Use the RAW file format to record and play back all streams that Adobe Media Server supports, including H.264 video, data-only, audio-only, and so on. The RAW file format is a server feature; any version of Flash Player can publish or play a RAW stream. However, multi-bitrate stream support (also called dynamic streaming) requires Flash Player 10 and higher.

Container element.

Note: The RAW file format is internal to Adobe Media Server. At this time, you cannot edit these files with third-party tools or convert the files to FLV format or MP4 format.

Availability

Flash Media Server 4

Contained elements

[MaxAggMsgSize](#) ([raw](#))

See also

[Streams](#)

ReadBufferSize

The size of the largest I/O read buffers that the server uses, in kilobytes. Larger buffers provide better performance for some use cases. However, larger buffers use more memory. The default value is 4096.

RecBuffer

Container element.

Contains elements that configure the buffer for recording.

When you add DVR functionality to an application, the size of the recording buffer affects the play-while-record experience. A shorter buffer decreases latency between real-time and playback, but hurts performance.

Contained elements

[MaxSize \(RecBuffer\)](#), [MaxTimestampSkew](#), [MaxFlushSize](#), [MaxFlushTime](#), [AllowedVideoLag](#)

Recording

Container element.

The Recording elements configure how the server records F4V/MP4 file types.

Contained elements

[MaxELSTEntries](#), [MaxAudioSampleDescriptions](#), [MaxDataSampleDescriptions](#),
[MaxVideoSampleDescriptions](#)

RecordsNumber

The number of records in the log queue. If an attempt to open the log fails and the number of records in the queue is greater than or equal to this number, the core process is shut down. The default value of -1 allows an unlimited number of records.

RecoveryTime

The time to wait, in seconds, before allowing another login attempt after a user has reached the value of `MaxFailures`.

The default value is 30 seconds.

If the number of failed login attempts equals the value of `LockoutLimit`, the admin server must be restarted before that user can attempt to log in again.

Example

```
<RecoveryTime>30</RecoveryTime>
```

See also

[LogInLimits](#), [LockoutLimit](#), [MaxFailures](#)

Registry

Flash Media Server 4.5

Enable a registry core to activate a single AMSCore instance dedicated to handling services for Adobe Media Gateway. This AMSCore process comes to life on start-up and allows Adobe Media Gateway to register its services to Adobe Media Server. Enable this registry core only when using Adobe Media Gateway.

```
<Registry enabled="false">  
  <!-- Adaptor to run registry core on -->  
  <AdaptorName>_defaultRoot_</AdaptorName>  
</Registry>
```

See also

[“AdaptorName” on page 195](#)

ResourceLimits

Container element.

The elements nested within the `ResourceLimits` container specify the maximum resource limits for the server, including the HTTP and RTMP protocols.

Contained elements

[FLVCachePurge](#), [FLVCache](#), [RecBuffer](#), [CPUMonitor](#), [ThreadPoolGC](#), [MsgPoolGC](#), [ApplicationGC](#), [FLVCacheSize](#), [SocketGC](#), [ProcVMSizeMonitorInterval](#), [ProcVMSizeWarn](#), [ProcVMSizeNominal](#), [SSLSessionCacheGC](#), [Connector](#), [Protocol](#), [IPCQueues](#), [MessageCache](#), [SmallMemPool](#), [LargeMemPool](#), [SegmentsPool](#), [Master](#)

RetryNumber

Number of times to retry opening a log file. Relevant when opening a log file fails. The default value is 0.

RetrySleep

Number of milliseconds the server waits before retrying to open a log file. Relevant when opening a log file fails. The default value is 100 ms.

Root

Container element.

The `Root` element is a container for all the other elements in the `Server.xml` file.

RTMP (AdminServer)

Container element.

This container holds elements that configure which versions of RTMP can be used to connect to the Administration Server. RTMP is the protocol used for communication between Flash Player and Adobe Media Server.

Contained elements

[RTMPE](#)

RTMP (Connector)

Container element.

This container holds the elements that configure RTMP (Real-Time Messaging Protocol) for communication between Flash Remoting and Adobe Media Server.

The following reference table lists the default values for all thread configurations.

Default Value	Description
0	Allocates the default number of threads (10).
>0	Allocates the exact number of threads specified.

Default Value	Description
<0	Associates the default value with the number (N) of processors.
-1	Allocates 1xN threads.
-2	Allocates 2xN threads.

Contained elements

[MinIOThreads](#), [MaxIOThreads](#), [NumCRThreads](#), [MinConnectionThreads](#), [MaxConnectionThreads](#),
[MaxConnectionQueueSize](#)

RTMP (Protocol)

Container element.

This container holds the elements that configure RTMP (Real-Time Messaging Protocol). RTMP is the protocol used for communication between Flash Player and Adobe Media Server.

Contained elements

[Edge](#), [Core](#), [Admin](#), [SocketSndBuf](#), [SocketRcvBuff](#)

RTMPE

Located in the [RTMP](#) container.

Specifies whether Encrypted Real-Time Messaging Protocol (RTMPE) can be used to connect to the Administration Server. (RTMPE also covers RTMPTE.) The default value is `true`. Setting this element to `false` prohibits RTMPE and RTMPTE from being used.

Scope

Located in the [Logging](#) container.

This element determines whether to write a separate log file for each virtual host or to write one log file for the server.

The value for this element is `server` or `vhost`. The default is `server`, which enables logging for all processes on the server.

Security

Description

Container element.

The elements in this section configure server security.

Contained elements

[LogInLimits](#), [MaxUrlLength](#), [VirtualDirectoryForFile](#)

Availability

Flash Media Server 4

SegmentsPool

Container element.

The elements in this section configure how the segments pool caches segments of video files within Adobe Media Server to increase performance of video streaming and keep frequently used video files in memory.

Contained elements

[MaxCacheUnits](#), [MaxCacheSize](#), [MaxUnitSize](#), [FreeRatio](#), [GlobalRatio](#), [MaxAge](#), [UpdateInterval](#), [FreeMemRatio](#)

Server

Container element.

The elements nested within the `Server` element contains the elements that configure the server.

Contained elements

[NetworkingIPv6](#), [CrossDomainPath](#), [Plugins](#), [SSL](#), [Process](#), [Mask](#), [LicenseInfo](#), [LicenseInfoEx](#), [ReadBufferSize](#), [AdminServer](#), [AutoCloseIdleClients](#), [ResourceLimits](#), [Logging](#), [Localhost](#), [PublicIP](#), [SWFVerification](#), [Streams](#), [Httpd](#)

ServerDomain

Specifies the host name (with the domain) of the server computer. The server uses this value to set the hostname in the referrer header tag when making a net connection to a remote server. If this element is empty the hostname field is empty in referrer header. For security purposes, some application servers require this information as a part of incoming connection requests. If this element is not set, the host name field is not supplied in the referrer header.

Example

```
<ServerDomain>mydomain.global.mycompany.com</ServerDomain>
```

Service

Located in the [Httpd](#) container.

On Windows only, specifies the name used to install the webserver as an NT service.

Services

Container element.

The elements in this section control the IPC message queue used by the edge and core processes to communicate with each other.

Contained elements

[HeapSize](#), [MaxQueueSize](#)

SmallMemPool

Container element.

The elements in this section configure the small memory pool, which saves small chunks of memory in the server to increase performance of small allocations.

Contained elements

[MaxCacheUnits](#), [MaxCacheSize](#), [MaxUnitSize](#), [FreeRatio](#), [GlobalRatio](#), [MaxAge](#), [UpdateInterval](#), [FreeMemRatio](#)

SocketGC

Located in the [AdminServer](#) and [ResourceLimits](#) containers.

Specifies how often, in seconds, the server checks for and removes inactive sockets. The default value is 60 seconds.

Example

```
<SocketGC>60</SocketGC>
```

SocketOverflowBuckets

Located in the [Edge](#), [Core](#), [Admin](#), [ECCP](#), [ACCP](#) containers.

Specifies the number of overflow buckets if all slots in the socket table are in use.

The default number of buckets is 16; specify -1 to use the default number of buckets.

Example

```
<Admin>  
  <SocketOverflowBuckets>-1</SocketOverflowBuckets>  
</Admin>
```

SocketRcvBuff

The size of the client socket receive buffer, in bytes. The default value is 0, which tells the server to use the operating system default values.

You should explicitly set this value only if you have a very high bandwidth connection that requires a large socket buffer. Setting a high value significantly increases the amount of memory used by each client. It is recommended that you do not explicitly set this value.

SocketSndBuf

The size of the client socket send buffer, in bytes. The default value is 0, which tells the server to use the operating system default values.

You should explicitly set this value only if you have a very high bandwidth connection that requires a large socket buffer. Setting a high value significantly increases the amount of memory used by each client. It is recommended that you do not explicitly set this value.

SocketTableSize

Located in the [Edge](#), [Core](#), [Admin](#), [ECCP](#), [ACCP](#) containers.

Specifies the size of the direct-access socket table for quick look-up. The default size is 200. Use -1 for the default value.

Example

```
<Admin>  
  <SocketTableSize>-1</SocketTableSize>  
</Admin>
```

SSL

Container element.

The SSL elements in Server.xml configure the server to act as an SSL-enabled client by securing the outgoing connections.

Contained elements

[SSLRandomSeed](#), [SSLSessionCacheGC](#), [SSLClientCtx](#)

SSLCACertificateFile

Located in the [SSLClientCtx](#) container.

Specifies the name of a file that contains one or more CA (Certificate Authority) digital certificates in PEM (Privacy Enhanced Mail) encryption format.

SSLCACertificatePath

Located in the [SSLClientCtx](#) container.

Specifies the name of a directory containing CA certificates. Each file in the directory must contain only a single CA certificate. File names must be the hash with "0" as the file extension.

For Win32 only: If this element is empty, attempts are made to find CA certificates in the certs directory located at the same level as the conf directory. The Windows certificate store can be imported into this directory by running `AMSMaster - console - initialize` from the command line.

SSLCipherSuite

Located in the [SSLClientCtx](#) container.

Specifies the suite of encryption ciphers that the server uses to secure communications.

This element is a colon-delimited list of encryption resources, such as a key-exchange algorithm, authentication method, encryption method, digest type, or one of a selected number of aliases for common groupings. Each item in the cipher list specifies the inclusion or exclusion of an algorithm or cipher. In addition, there are special keywords and prefixes. For example, the keyword `ALL` specifies all ciphers, and the prefix `!` removes the cipher from the list.

The default cipher list instructs the server to accept all ciphers, but block those using anonymous Diffie-Hellman authentication, block low-strength ciphers, block export ciphers, block MD5 hashing, and sort ciphers by strength from highest to lowest level of encryption.

Important: Contact Adobe Support before changing the default settings.

The cipher list consists of one or more cipher strings separated by colons. Commas or spaces are also acceptable separators, but colons are normally used.

The string of ciphers can take several different forms.

- It can consist of a single cipher suite, such as RC4-SHA.

- It can represent a list of cipher suites containing a certain algorithm, or cipher suites of a certain type.
 For example, `SHA1` represents all cipher suites using the digest algorithm SHA1, and `SSLv3` represents all SSL v3 algorithms.
- Lists of cipher suites can be combined in a single cipher string using the `+` character as a logical and operation.
 For example, `SHA1+DES` represents all cipher suites containing the `SHA1` and `DES` algorithms.
- Each cipher string can be optionally preceded by the characters `!`, `-`, or `+`.
- If `!` is used, then the ciphers are permanently deleted from the list. The ciphers deleted can never reappear in the list even if they are explicitly stated.
- If `-` is used, then the ciphers are deleted from the list, but some or all of the ciphers can be added again later.
- If `+` is used, then the ciphers are moved to the end of the list. This option doesn't add any new ciphers—it just moves matching existing ones.
- If none of these characters is present, then the string is just interpreted as a list of ciphers to be appended to the current preference list.
- If the list includes any ciphers already present, the server does not evaluate them.
- The cipher string `@STRENGTH` sorts the current cipher list in order of the length of the encryption algorithm key.

The components can be combined with the appropriate prefixes to create a list of ciphers, including only those ciphers the server is prepared to accept, in the order of preference.

Example

This cipher string instructs the server to accept all ciphers except those using anonymous or ephemeral Diffie-Hellman key exchange.

```
<SSLCipherSuite>ALL:!ADH:!EDH</SSLCipherSuite>
```

These cipher strings instruct the server to accept only RSA key exchange and refuse export or null encryption. The server evaluates both strings as equivalent.

```
<SSLCipherSuite>RSA:!NULL!EXP</SSLCipherSuite>
<SSLCipherSuite>RSA:LOW:MEDIUM:HIGHS</SSLCipherSuite>
```

This cipher list instructs the server to accept all ciphers but place them in order of decreasing strength. This sequencing allows clients to negotiate for the strongest cipher that both they and the server can accept.

```
<SSLCipherSuite>ALL:+HIGH:+MEDIUM:+LOW:+EXP:+NULL</SSLCipherSuite>
```

This string instructs the server to accept only high- and medium-strength encryption, with the high being preferred, and reject export-strength versions.

```
<SSLCipherSuite>ALL:+HIGH:!LOW:!EXP:!NULL</SSLCipherSuite>
```

This string instructs the server to accept all ciphers but to order them so that SSLv2 ciphers come after SSLv3 ciphers.

```
<SSLCipherSuite>ALL:+SSLv2</SSLCipherSuite>
```

The following is the complete list of components that the server can evaluate.

Key exchange algorithm	Description
kRSA	Key exchange
kDhr	Diffie-Hellman key exchange with RSA key
kDHd	Diffie-Hellman key exchange with DSA key

Key exchange algorithm	Description
RSA	Ephemeral Diffie-Hellman key exchange
DH	RSA key exchange
EDH	Ephemeral Diffie-Hellman key exchange
ADH	Anonymous Diffie-Hellman key exchange

Authentication methods	Description
aNULL	No authentication
aRSA	RSA authentication
aDSS	DSS authentication
aDH	Diffie-Hellman authentication

Encryption methods	Description
eNULL	No encoding
DES	DES encoding
3DES	Triple-DES encoding
RC4	RC4 encoding
RC2	RC2 encoding
IDEA	IDEA encoding
NULL	No encryption
EXP	All export ciphers (40-bit encryption)
LOW	Low-strength ciphers (no export, DES)
MEDIUM	128-bit encryption
HIGH	Triple-DES encoding

Digest types	Description
MD5	MD5 hash function
SHA1	SHA1 hash function
SHA	SHA hash function

Additional aliases	Description
All	All ciphers
SSLv2	All SSL version 2.0 ciphers
SSLv3	All SSL version 3.0 ciphers
DSS	All ciphers using DSS authentication

SSLClientCtx

Container element.

Configures the server to act as an SSL-enabled client by securing the outgoing connections.

Contained elements

[SSLVerifyCertificate](#), [SSLCACertificatePath](#), [SSLCACertificateFile](#), [SSLVerifyDepth](#), [SSLCipherSuite](#)

SSLRandomSeed

Located in the [SSL](#) container.

Specifies the number of bytes of entropy to use for seeding the PRNG. You cannot specify anything less than 8 bytes, and the default is 16. Entropy is a measure of randomness. The more entropy, the more random numbers from the PRNG will be.

Example

```
<SSLRandomSeed>16</SSLRandomSeed>
```

SSLSessionCacheGC

Located in the [SSL](#) container.

Specifies how often to check for and remove expired sessions from the server-side session cache.

Example

```
<SSLSessionCacheGC>5</SSLSessionCacheGC>
```

SSLVerifyCertificate

Located in the [SSLClientCtx](#) container.

Specifies whether the certificate returned by the server should be verified. Certificate verification is enabled by default. To disable certificate verification, specify `false`.

Note: Disabling certificate verification can result in security problems.

Example

```
<SSLVerifyCertificate>>true</SSLVerifyCertificate>
```

SSLVerifyDepth

Located in the [SSLClientCtx](#) container.

Specifies the maximum depth of the certificate chain to accept. If a self-signed root certificate cannot be found within this depth, certificate verification fails. The default value is 9.

Example

```
<SSLVerifyDepth>9</SSLVerifyDepth>
```

StreamAdaptors

Specifies which library to load for a particular stream, based on the stream's prefix. This effectively defines a whitelist of libraries that can be loaded by the server for each stream type.

When you play a stream, you specify a prefix. The prefix indicates what kind of stream it is. The following example shows a prefix that indicates that the stream is of type MP3:

```
play("mp3:mystream.mp3");
```

Note that the the prefix does not necessarily match the file type. For example, for MP4, you can specify the following:

```
play("mp4:mystream.f4v");
```

In this case, MP4 is the encoding, not the file type. So you can use mp4:mystream.mov, mp4:mystream.mp4, and so on.

To specify a library for a particular stream, add a sub-element with the following syntax:

```
<prefix lib="library_name"/>
```

Supported streams are:

- FLV (processed by the library specified by the `f1v` sub-element)
- F4F (processed by the library specified by the `f4f` sub-element)
- ID3 (processed by the library specified by the `id3` sub-element)
- MP3 (processed by the library specified by the `mp3` sub-element)
- MP4 (processed by the library specified by the `mp4` sub-element)
- RAW (processed by the library specified by the `raw` sub-element)

For FLV streams, you do not specify the prefix or the extension; for example:

```
play("mystream");
```

In this case, the adaptor that you define for the FLV file does not match the prefix (because there is none). The library defined by the `f1v` sub-element is matched by Adobe Media Server internally to determine which library to use to process the stream.

Example

```
<f1v lib="libf1v"/>  
<f4f lib="libf4f"/>  
<mp3 lib="libmp3"/>  
<mp4 lib="libmp4"/>  
<raw lib="libraw"/>  
<id3 lib="libid3"/>
```

See also

[Streams](#)

StreamLogLevel

Controls log levels for all stream adaptors (FLV, MP4, and RAW). Possible values are `verbose`, `warning`, and `error`.

The default value is `warning`.

Example

```
<StreamLogLevel>warning</StreamLogLevel>
```

See Also

[Streams](#)

Streams

Container element.

The Streams elements configure how the server plays and records MP4, F4V, MOV, and other MP4 file types.

Contained elements

[Mp4](#), [StreamLogLevel](#), [StreamAdaptors](#)

SWFFolder

Located in the [SWFVerification](#) container.

Specifies a folder containing SWF files that are verified to connect to any application on this server. Use a semicolon to separate multiple directories.

Example

The following example allows SWF files from either the C or the D directory to be authenticated:

```
<SWFFolder>C:\SWFs;D:\SWFs</SWFFolder>
```

SWFVerification

Container element.

Contains elements that configure how SWF files connecting to an application are verified.

Contained elements

[SWFFolder](#), [DirLevelSWFScan](#), [MaxInitDelay](#), [MinGoodVersion](#), [Cache](#)

TerminatingCharacters

Located in the [Logging](#) container.

Specifies the final characters of each log entry in log files. The default is CRLF (carriage return and line feed).

Example

```
<TerminatingCharacters>CRLF</TerminatingCharacters>
```

ThreadPoolGC

Located in the [ResourceLimits](#) container.

Specifies in minutes how often Adobe Media Server checks for and removes unused I/O threads.

The default time is 20 minutes. You cannot specify less than 20 minutes.

Example

```
<ThreadPoolGC>25</ThreadPoolGC>
```

Time

Located in the [Logging](#) container.

Specifies the time field in a log file.

The time field in a log file can be specified either as UMT (GMT) or local time. The default setting is `local`.

Example

```
<Time>local</Time>
```

TrimSize

Located in the [HandleCache](#) container.

Specifies a percentage of cached handles to remove. Can be specified as a number between 0 and 1, representing 0% to 100%. The default value is 0.2 (20%).

Example

```
<TrimSize>0.2</TrimSize>
```

TTL

Located in the [Cache](#) container.

Specifies in minutes how long each SWF file remains in the SWF verification data cache. The default value is 1440 minutes (24 hours).

UID

Located in the [Process](#) containers.

This element contains the server process user ID.

If you do not specify a `UID` or `GID`, the server or Administration Server runs as root. This element is applicable to Adobe Media Server running on Linux systems only.

See also

[GID](#)

UpdateInterval

Specifies how often thread statistics are collected. The default value is every 1024 messages.

Example

```
<UpdateInterval>1024</UpdateInterval>
```

UpdateInterval (Cache)

Located in the [Cache](#) container.

Specifies the maximum time in minutes to wait for the server to scan the SWF folders for updates when there is a miss in the cache. The default value is 5 minutes.

UserDefined

Container element.

Define elements within the `UserDefined` element to store data that File plug-ins and Authorization plug-ins can retrieve.

VirtualDirectoryForFile

Enables virtual directory mappings for File objects. The default is disabled.

If you map File objects in the `Application.xml` file but don't have the feature enabled in the `Server.xml` file, you'll see the following message in the log:

```
"Virtual directories for file objects is not supported due to the Server level security setting."
```

To avoid this log message, enable `VirtualDirectoryForFile`. For more information, see ["Enable virtual directory mappings for server-side File objects"](#) on page 44.

Example

```
<VirtualDirectoryForFile enable="false"></VirtualDirectoryForFile>
```

See also

[Security](#)

Availability

Flash Media Server 4

VideoAutoBufferReadSize

Located in the `Playback` container.

The server caches F4V/MP4 video, audio, and other data in memory. Caching data allows the server to make fewer disk reads and to perform better. Each data type (audio, video, and other) has its own cache. You can tune the size of each cache, depending on the type of content you are delivering, to achieve better disk performance. For example, if you are delivering audio-only content, increase the `AudioAutoBufferReadSize` buffer, and decrease the `VideoAutoBufferReadSize` buffer.

The minimum size is 1024 bytes. The default value is 153600 bytes.

Users.xml file

The `Users.xml` file is located at the root level of the `RootInstall/conf` directory. Edit this file to identify Adobe Media Server administrators and set their access permissions. Edit this file to set permissions for Administration API calls to Adobe Media Administration Server.

To see the element structure and default values, see the `Users.xml` file installed with Adobe Media Server in the `RootInstall/conf/` directory.

AdminServer

Container element.

The `HttpCommands` container nested within the `AdminServer` container configures the access level to Adobe Media Administration Server.

The Administration Service is separate from Adobe Media Server. When administrators use the Administration Console to connect to Adobe Media Server, they are connecting to Adobe Media Administration Server, which in turn connects to the server.

Contained elements

[HTTPCommands](#)

Allow (HTTPCommands)

Lists the Adobe Media Administration Server commands that the administrator can access using HTTP. You can authorize an administrator to use multiple HTTP commands for access by creating a comma-separated list of the commands. The value `All` authorizes the administrator to use all HTTP commands. However, Adobe does not recommend this usage as it creates a security risk.

Example

```
<Allow>foo.yourcompany.com, adobe.com, 10.60.1.133, 10.60</Allow>
```

See also

[Enable](#), [Deny \(HTTPCommands\)](#), [Order \(HTTPCommands\)](#)

Allow (User)

Lists the specific hosts from which an administrator can connect to Adobe Media Administration Server. The administrator can only connect to the server from those hosts specified in this `Allow` element. You authorize the administrator's access by creating a comma-delimited list of the accessible host names or domain names and/or full or partial IP addresses. Whenever possible, use the IP addresses in the `Allow` element to improve the server's performance when processing connection requests.

Example

```
<Allow>foo.yourcompany.com, adobe.com, 10.60.1.133, 10.60</Allow>
```

See also

[Password](#), [Deny \(User\)](#), [Order \(User\)](#)

Deny (HTTPCommands)

Adobe Media Server uses two elements named `Deny`: the `Deny` element in the `User` container and the `Deny` element in the `HTTPCommands` container.

This `Deny` element lists the Adobe Media Administration Server commands that an administrator cannot use through HTTP.

You can deny an administrator the use of multiple HTTP commands to access the Administration Service by creating a comma-separated list of those HTTP commands.

Example

```
<Deny>Deny,Allow</Deny>
```

See also

[Enable, Allow \(HTTPCommands\)](#), [Order \(HTTPCommands\)](#)

Deny (User)

Adobe Media Server uses two elements named `Deny`: the `Deny` element in the `User` container and the `Deny` element in the `HTTPCommands` container.

This element lists those hosts from which the administrator is not authorized to connect to Adobe Media Administration Server. You restrict the administrator's access by creating a comma-delimited list of those host names or domain names and/or (full or partial) IP addresses.

Example

This example lists the computers sending connection requests that Adobe Media Administration Server will not accept.

```
<Deny>foo.yourcompany.com,adobe.com,10.60.1.133,10.60</Deny>
```

See also

[Password, Allow \(User\)](#), [Order \(User\)](#)

Enable

This element enables or disables the use of HTTP requests to execute administrative commands.

Setting this element enables HTTP requests to execute administrative commands. To disable administrative access through the use of HTTP requests, do not set this element.

Example

```
<Enable>>true</Enable>
```

See also

[Allow \(HTTPCommands\)](#), [Deny \(HTTPCommands\)](#), [Order \(HTTPCommands\)](#)

HTTPCommands

Container element.

This section contains the settings for those Adobe Media Administration Server commands that can be accessed through HTTP. The default value is `ping`. Specify each Administration API that may be called over HTTP in a comma-delimited list. When finished, restart the server.

Contained elements

[Enable, Allow \(HTTPCommands\)](#), [Deny \(HTTPCommands\)](#), [Order \(HTTPCommands\)](#)

Order (HTTPCommands)

Adobe Media Server uses two `Order` elements: one in the `HTTPCommands` container and another in the `User` container.

Specifies the order in which to evaluate the `Deny` and `Allow` commands.

Example

The sequence `Deny, Allow` means the HTTP command is allowed if the command is in the `Allow` list of commands or not in the `Deny` list.

```
<Order>Deny, Allow</Order>
```

The sequence `Allow, Deny` means the HTTP command is allowed if it is in the `Allow` list of commands and not in the `Deny` list.

```
<Order>Allow, Deny</Order>
```

See also

[Enable, Allow \(HTTPCommands\)](#), [Deny \(HTTPCommands\)](#)

Order (User)

Adobe Media Server uses two `Order` elements: one in the [HTTPCommands](#) container, and the other in the [User](#) container.

Specifies the sequence in which Adobe Media Server evaluates the `Allow` and `Deny` elements for an administrator.

Example

The default sequence `Allow, Deny` means that administrative access is allowed unless the user is specified in the `Allow` list of commands and not in the `Deny` list:

```
<Order>Allow, Deny</Order>
```

The alternative sequence `Deny, Allow` means that administrative access is allowed unless the user is specified in the `Deny` list of commands and not specified in the `Allow` list.

```
<Order>Deny, Allow</Order>
```

See also

[Password, Allow \(User\)](#), [Deny \(User\)](#)

Password

A salted hash of the password for vhost administrators.

Passwords cannot be empty strings (" "). Passwords are usually encrypted.

Example

```
<Password>e206a5e1b52dcb4eaf024ca6adbe321b86cf0079bb747b78134ddaf8375e10aff905da0b28e84c5a</Password>
```

See also

[Allow \(User\)](#), [Deny \(User\)](#), [Order \(User\)](#)

PasswordPolicy

You can use `PasswordPolicy` element to enable or disable minimum password length policy.

Example

```
<Root><PasswordPolicy enable="false"></PasswordPolicy></Root>
```

Root

Container element.

The `Root` element is a container for all the other elements. If the `Users.xml` file resides under a virtual host (to define administrators for that virtual host), then this tag must have its `name` attribute set to the name of the virtual host under which it resides.

Example

```
<Root name="_defaultVHost_">
```

User

This element identifies an administrator of the server.

You can identify multiple administrators of a virtual host by creating a profile for each administrator.

Example

Use the `name` attribute to identify the login name of a Adobe Media Server administrator:

```
<User name="jsmith"></User>
```

See also

[UserList](#)

UserList

Container element.

The `UserList` element defines and holds information about server administrators.

Contained elements

[User](#), [Password](#), [Allow \(User\)](#), [Deny \(User\)](#), [Order \(User\)](#)

Vhost.xml file

The `Vhost.xml` configuration file defines an individual virtual host. Each virtual host directory on the server contains its own `Vhost.xml` file.

The `Vhost.xml` file contains elements that define the settings for the virtual host. These settings include aliases for the virtual host, the location of the virtual host's application directory, limits on the resources the virtual host can use, and other parameters.

Each virtual host must have its own directory inside the adaptor directory. The name of the directory must be the actual name of the virtual host, such as `streaming.adobe.com`. Each defined virtual host must be mapped to a DNS (domain name server) entry or another name resolution, such as a WINS address or a hosts file, that specifies an IP address on the server computer.

Each adaptor must contain a `_defaultVHost_` directory in addition to the custom virtual hosts that you define. If a client application tries to connect to a virtual host that does not exist, the server attempts to connect to `_defaultVHost_`. If you are using a secure port for the adaptor that contains the virtual host, you can only define one virtual host for the adaptor, in addition to `_defaultVHost_`.

To see the element structure and default values in `Vhost.xml`, see the `Vhost.xml` file installed with Adobe Media Server in the `RootInstall/conf/_defaultRoot/_defaultVhost_` directory.

Access

Container element.

The elements nested within the `Access` container configure the Access log settings. The Access logs are located in the `RootInstall/logs` directory.

Contained elements

[Checkpoints](#)

AggregateMessages

Determines whether aggregate messages are delivered from the edge cache when the virtual host is configured as an edge server. The default value is `false`.

If the edge server receives aggregate messages from the origin when this setting is disabled, the messages will be broken up before being cached.

Example

```
<AggregateMessages enabled="true"
  <MaxAggMsgSize>65536</MaxAggMsgSize>>
</AggregateMessages>
```

See also

[EdgeAutoDiscovery](#), [RouteEntry](#)

Alias

The `Alias` element specifies the assumed name(s) of the virtual host.

An alias is an alternative short name to use when connecting to the virtual host. The `Alias` element lets you specify additional names to connect to this virtual host. Use the `Alias` element to shorten long host names, or if you want to be able to connect to this virtual host with different names.

Example

```
<Alias name="abc">abc.adobe.com</Alias>
```

If the name of this virtual host is “abc.adobe.com”, but you wish to connect by simply specifying “abc”, then specify the alias `abc`. Keep in mind that `abc` must still map to the same IP address as “abc.adobe.com”.

If more than one virtual host on the same adaptor has been defined with the same alias, then the first match that is found is taken. You can avoid unexpected behavior by specifying a unique alias for each virtual host.

See also

[AliasList](#)

AliasList

Container element.

The elements nested in this section list the alias(es) for this virtual host. You can specify an unlimited number of aliases by adding additional `Alias` elements. Each `Alias` must map to the IP address of the virtual host.

Contained elements

[AggregateMessages](#)

Allow

This element is a comma-delimited list of domains that are allowed to connect to this virtual host. The default value is `all`. If the `Allow` element is left empty, the only connections allowed are those coming from the same domain.

Examples

```
<Allow>adobe.com,yourcompany.com</Allow>
```

This example allows only connections from the `adobe.com` and `yourcompany.com` domains.

```
<Allow>localhost</Allow>
```

This example allows `localhost` connections only.

```
<Allow>all</Allow>
```

This example allows connections from all domains. Adobe does not recommend the use of `all`; it may create a security risk.

See also

[Anonymous](#)

AllowOverride

Specifies whether overriding edge autodiscovery is allowed by specifying the `rtmpd` protocol. If enabled, edge autodiscovery is performed by default.

Example

```
<AllowOverride>true</AllowOverride>
```

See also

[Enabled](#), [WaitTime](#)

Anonymous

Configures the virtual host as an anonymous proxy (also called an *implicit* or *transparent proxy*) or as an explicit proxy. The default value is `false`. Setting this element to `true` creates an implicit proxy to intercept the incoming URIs.

Both anonymous and explicit proxies intercept and aggregate the clients' requests to connect to the origin server. Here are some key differences between anonymous and explicit proxies:

- The identity (IP address and port number) of an anonymous server is hidden from the client.
- The anonymous proxy does not change or modify the routing information in the incoming URI before connecting the client(s) to the origin server.
- The URI for an explicit proxy specifies the edge server(s) that will intercept connection requests to the origin server.

You can create a chain of proxies by specifying them in the URI.

- Any anonymous proxy in the chain passes on, without modification, the routing information in the URI to the next edge server in the chain.
- The routing information in the URI for a chain of explicit proxies specifies the edge servers that are chained together to intercept connection requests to the origin server.
- The routing information in the URI for a chain of explicit proxies specifically identifies the sequence of edge servers in the chain.
- The URI for a chain of explicit proxies directs all clients' connection requests through a specific sequence of edge servers before making the connection to the origin server.
- The explicit proxy modifies the routing information in the URI by stripping off its token or identifier in the URI before passing the URI on to the next server in the chain.

Example

```
<Anonymous>false</Anonymous>
```

See also

[Mode](#), [CacheDir](#), [LocalAddress](#), [RouteTable](#), [EdgeAutoDiscovery](#), [AggregateMessages](#)

AppInstanceGC

Specifies how often to check for and remove unused resources for application instances, such as Shared Objects, Streams, and Script engines.

The default interval is 1 minute.

Example

```
<AppInstanceGC>1</AppInstanceGC>
```

See also

[MaxConnections](#), [MaxEdgeConnections](#), [MaxAppInstances](#), [MaxStreams](#), [MaxSharedObjects](#)

AppsDir

Specifies the Applications directory for this virtual host.

The Applications directory is the base directory where all applications for this virtual host are defined. You define an application by creating a subdirectory with the application name.

- In Windows, the default `AppsDir` location is `C:\Program Files\Adobe\Adobe Media Server 4\applications`.
- In Linux, the default location is `/opt/adobe/ams/applications`.

Note: If you use this tag to map to a network drive, see “[Mapping directories to network drives](#)” on page 50 for additional information.

Example 1

```
<AppsDir>C:\MyApps;D:\NewApps</AppsDir>
```

You can specify multiple applications directories by separating locations with a semicolon (;). You can specify two locations, each of which contains application subdirectories. If you change the default location of the `AppsDir` element, be sure to include a directory named `admin` in each directory. This ensures that the Administration Console (`ams_adminConsole.swf`) will be able to connect to the virtual host.

If no location is specified for this element, the applications directory is assumed to be located in the `vhost` directory.

Example 2

The following example shows a mapping to a network drive:

```
<AppsDir>\myNetworkDrive\share\amsapps</AppsDir>
```

See also

[AliasList](#), [ResourceLimits](#), [VirtualKeys](#)

AutoCloseIdleClients

Container element.

Determines whether or not to close idle clients automatically.

Set the `enable` attribute to `true` to close idle clients. If the `enable` attribute is omitted or set to `false`, the feature is disabled. The default value is `false`.

A client is active when it is sending or receiving data. Use `AutoCloseIdleClients` to specify how often the server should check for idle clients. When a client has been idle longer than the maximum idle time (60 seconds by default), the server sends a status message to the `NetConnectionObject` (the client). The server closes the client connection to the server and writes a message to the access log. The server also writes a message such as “Client *x* has been idle for *y* seconds” in the core and event logs.

To configure the closing of idle connections, you must enable the feature in the `Server.xml` file. Once you enable the feature in the `Server.xml` file, you can disable the feature for individual virtual hosts in the `Vhost.xml` files or for individual applications in `Application.xml`. The values defined in the `Vhost.xml` configuration file apply to all clients connected to the `Vhost`, unless values are defined in the `Application.xml` file. The `Application.xml` values override the `Vhost.xml` values. Subsequently, the values defined in the `Server.xml` configuration file apply to all clients connected to the server, unless the values are defined in the `Vhost.xml` file. The `Vhost.xml` values override the `Server.xml` values.

Example

```
<AutoCloseIdleClients enable="false">  
  <MaxIdleTime>600</MaxIdleTime>  
</AutoCloseIdleClients>
```

See also

[AppsDir](#), [MaxIdleTime](#)

CacheDir

Container element.

This element enables or disables writing recorded streams to disk. Set this element on an edge server or an intermediate origin server to control the caching behavior. The contents of the cache change. This element controls whether the cached streams are written to disk, in addition to being cached in memory.

The edge server caches content locally to aid performance, especially for vod (video on demand) applications. Caching static content can reduce the overall load placed on the origin server.

The default value of the `enabled` attribute is `false`. The `useAppDir` attribute determines whether to separate cache subdirectories by application. The default value is `true`.

If a server has multiple virtual hosts, each virtual host should point to its own cache directory.

Contained elements

[Mode](#), [Anonymous](#), [LocalAddress](#), [RouteTable](#), [Path](#), [MaxSize](#)

See also

[RequestTimeout](#)

Checkpoints

Enables logging checkpoint events. Checkpoint events log bytes periodically from the start to the end of an event. The following are available as checkpoint events: `connect-continue`, `play-continue`, and `publish-continue`.

This element contains the `enable` attribute which you can set to `true` or `false`. Set the `enable` attribute to `true` to turn on checkpoint events in logs. The default value is `false`.

You must enable checkpoint events at the server level in the `Server.xml` file. You can disable checkpoints at the vhost and application level in the `Vhost.xml` and `Application.xml` files. You can also override the logging interval at the vhost and application levels.

Contained elements

[CheckInterval](#), [LogInterval](#)

DNSSuffix

Specifies the primary DNS suffix for this virtual host.

If a reverse DNS look up fails to return the domain as part of the host name, then this element is used as the domain suffix.

See also

[AliasList](#), [AppsDir](#), [ResourceLimits](#), [VirtualKeys](#), [VirtualDirectory](#)

EdgeAutoDiscovery

Container element.

Contains elements that configure edge autodiscovery. An edge server may connect to another server that is part of a cluster. In this case, the edge server tries to determine which server in the cluster it should connect to (may or may not be the server specified in the URL).

Example

```
<EdgeAutoDiscovery>
  <Enabled>false</Enabled>
  <AllowOverride>true</AllowOverride>
  <WaitTime>1000</WaitTime>
</EdgeAutoDiscovery>
```

See also

[Enabled](#), [AllowOverride](#), [WaitTime](#)

Enabled

Specifies whether edge autodiscovery is enabled. If `Enabled` is set to `true`, the edge server tries to determine to which server in a cluster it should connect. The default value is `false`.

Example

```
<Enabled>false</Enabled>
```

See also

[AllowOverride](#), [WaitTime](#)

Key

When Flash Player connects to Adobe Media Server, it sends the server a string containing its platform and version information. You can add `Key` elements that map Flash Player information to keys. The keys can be any alphanumeric value. In the following example, the keys are A and B:

```
<VirtualKeys>
  <Key from="WIN 8,0,0,0" to="WIN 9,0,45,0">A</Key>
  <Key from="WIN 6,0,0,0" to="WIN 7,9,9,9">B</Key>
  <Key from="MAC 8,0,0,0" to="MAC 9,0,45,0">A</Key>
  <Key from="MAC 6,0,0,0" to="MAC 7,9,9,9">B</Key>
</VirtualKeys>
```

In the `VirtualDirectory` element, you map virtual directories used in URLs to physical directories containing streams. In the following example, if a client with key A requests a stream with the URL `NetStream.play("vod/someMovie")`, it is served the stream `c:\on2\someMovie.flv`. If a client with key B requests a stream with the URL `NetStream.play("vod/someMovie")`, it is served the stream `c:\sorenson\someMovie.flv`.

```
<VirtualDirectory>
  <Streams key="A">vod;c:\on2</Streams>
  <Streams key="B">vod;c:\sorenson</Streams>
</VirtualDirectory>
```

Note: You can also set these values in a server-side script. For more information, see the `Client.virtualKey` and `Stream.setVirtualPath()` entries in the *Server-Side ActionScript Language Reference*.

For more information, see the “Configuring content storage” section of the *Configuration and Administration Guide* at www.adobe.com/go/learn_ams_content_en.

See also

[VirtualKeys](#)

LocalAddress

This element binds an outgoing edge connection to a specific local IP address.

The `LocalAddress` element lets you allocate incoming and outgoing connections to different network interfaces. This strategy is useful when configuring an edge to either transparently pass on or intercept requests and responses.

If the `LocalAddress` element is not specified, then outgoing connections bind to the value of the `INADDR_ANY` Windows system variable.

See also

[Proxy](#)

Logging

Container element.

Contains elements that control logging.

Contained elements

LogInterval

Specifies how often to log a checkpoint, in seconds. This value should be larger than the value for `CheckInterval`. If the value is smaller, the server logs a checkpoint every check interval. The default value is 3600 seconds (60 minutes).

MaxAggMsgSize

Specifies the size in bytes of aggregate messages returned from the edge cache. (Aggregate messages must be enabled.) The default size is 65,536.

This setting only applies to messages retrieved from the disk cache. Aggregate messages received directly from the origin server are returned as is and their size is determined by the origin server settings for aggregate message size.

Example

```
<MaxAggMsgSize>66536</MaxAggMsgSize>
```

See also

[AggregateMessages](#)

MaxAppInstances

Specifies the maximum number of application instances that can be loaded into this virtual host.

A chat application, for example, might require more than one instance, because each chat room represents a separate instance of the application on the server. The default number is 15,000 application instances.

A Flash SWF file defines which application instance it is connecting to by the parameters it includes with its `ActionScript connect` call.

Example

```
<MaxAppInstances>15000</MaxAppInstances>
```

See also

[MaxConnections](#), [MaxEdgeConnections](#), [MaxStreams](#), [MaxSharedObjects](#), [AppInstanceGC](#)

MaxConnections

Specifies the maximum number of clients that can connect to this virtual host.

The maximum number of allowed connections is encoded in the license file. Connections are denied if the specified limit is exceeded. The default number is -1, which represents an unlimited number of connections.

Example

```
<MaxConnections>-1</MaxConnections>
```

See also

[MaxAppInstances](#), [MaxEdgeConnections](#), [MaxStreams](#), [MaxSharedObjects](#), [AppInstanceGC](#)

MaxEdgeConnections

Specifies the maximum number of connections that can remotely connect to this virtual host. This number is enforced by the license key.

Example

```
<MaxEdgeConnections>1</MaxEdgeConnections>
```

See also

[MaxConnections](#), [MaxAppInstances](#), [MaxStreams](#), [MaxSharedObjects](#), [AppInstanceGC](#)

MaxIdleTime

Specifies the maximum idle time allowed, in seconds, before a client is disconnected.

The default idle time is 3600 seconds (60 minutes). If you set a value lower than 60 seconds, the server uses the value 1200 seconds (20 minutes).

A different value can be set for each virtual host. If no value is set for this element in the Vhost.xml file, the server uses the value in the Server.xml file. The value for the `MaxIdleTime` element in the Vhost.xml file overrides the value of the `MaxIdleTime` element in the Server.xml file.

Example

```
<MaxIdleTime>3600</MaxIdleTime>
```

See also

[AutoCloseIdleClients](#)

MaxSharedObjects

Specifies the maximum number of shared objects that can be created. The default number of shared objects is 50,000.

Example

```
<MaxSharedObjects>50000</MaxSharedObjects>
```

See also

[MaxConnections](#), [MaxEdgeConnections](#), [MaxAppInstances](#), [MaxStreams](#), [AppInstanceGC](#)

MaxSize

Specifies the maximum allowed size of the disk cache, in gigabytes. The server does LRU (least recently used) cleanup of the cache to keep it under the maximum size. The default value is 32 gigabytes. A value of 0 disables the disk cache. A value of -1 specifies no maximum.

See also

[NumBuckets](#), [NumBucketsAtRisk](#)

MaxStreams

Specifies the maximum number of streams that can be created for live streams. The default number of streams is 250,000.

Note: This property is ignored for recorded streams.

Example

```
<MaxStreams>250000</MaxStreams>
```

See also

[MaxConnections](#), [MaxAppInstances](#), [MaxSharedObjects](#), [AppInstanceGC](#)

Mode

The `Mode` element configures whether the server runs as an origin server or as an edge server.

The `Mode` element can be set to `local` or `remote`. The default setting is `local`.

- When the `Mode` element is set to `local`, Adobe Media Server runs its applications locally and is called an origin server.
- When the `Mode` element is set to `remote`, the server behaves as an edge server that connects to the applications running on an origin server.
- If the `Mode` element is undefined, the virtual host is evaluated as an alias for the default virtual host and assumes its configuration.

Example

```
<Mode>local</Mode>
```

See also

[Anonymous](#), [CacheDir](#), [LocalAddress](#), [RouteTable](#), [EdgeAutoDiscovery](#)

NumBuckets

Specifies the number of buckets to divide the cache into. The aggregate cache size is defined by `MaxSize`. Can be any value from 2 to 128; the default value is 8. More buckets mean that a smaller portion of the disk content will be deleted at any one time, but also that files will need to be moved to new buckets more often, which results in more disk activity and lower performance.

You can define the number of buckets that contain contents that can be moved with `NumBucketsAtRisk`.

Example

```
<NumBuckets>10</NumBuckets>
```

See also

[MaxSize](#), [NumBucketsAtRisk](#)

NumBucketsAtRisk

Specifies the number of buckets considered to be “at risk” of deletion. Can be any value from 0 to `NumBuckets - 1`; the default is `NumBuckets/2`. To avoid moving files too frequently at the expense of disk performance, only segments that are in the oldest `NumBucketsAtRisk` buckets will be moved to the newest bucket when accessed. A value of 0 means that segments, once pulled from the origin, are never moved to a newer bucket, effectively turning the cache into an LRU (least recently used) cache.

The default value of `NumBuckets/2` means that only segments in the “older half” of content will be moved. The idea is that segments in the “newer half” of content are more likely to be accessed again before they are deleted, thus making it less important to move them.

Example

```
<NumBucketsAtRisk>1</NumBucketsAtRisk>
```

See also

[MaxSize](#), [NumBuckets](#)

Path

Specifies the physical location of the proxy cache. By default, the location is `RootInstall/cache/`. The value must be an absolute path. Relative paths are ignored and the server uses the default folder.

See also

Proxy

Container element.

The elements nested in this section configure this virtual host as an edge server that can forward connection requests from applications running on one remote server to another server.

Note: *Whenever a virtual host is configured as an edge server, it behaves locally as a remote server.*

If this virtual host is configured to run in `remote` mode and you want to configure the properties of an outgoing SSL connection to an upstream server, the SSL connection to upstream servers will use the default configuration specified in the `SSL` section of the `Server.xml` file.

Contained elements

[Mode](#), [Anonymous](#), [CacheDir](#), [LocalAddress](#), [RouteTable](#), [EdgeAutoDiscovery](#), [SSL](#), [AggregateMessages](#), [RequestTimeout](#)

RequestTimeout

The maximum amount of time, in seconds, the server waits for a response to a request from an upstream server. A request can be for metadata, content, and so on. This value -1 specifies an unlimited amount of time (no timeout). The default value is 2 seconds.

See also

[CacheDir](#)

ResourceLimits

Container element.

The elements in this section specify the maximum resource limits for this virtual host.

Contained elements

[MaxConnections](#), [MaxEdgeConnections](#), [MaxAppInstances](#), [MaxStreams](#), [MaxSharedObjects](#), [AppInstanceGC](#),

RouteEntry

Instructs the edge server to forward the connection request to one server's IP address and port number [host:port] to a different IP address and port number.

Edge servers are configured with the `RouteEntry` element to direct connections to another destination. The `RouteTable` element contains the `RouteEntry` elements that control where the edge server reroutes requests.

You can also add the `protocol` attribute to an individual `RouteEntry` element to specify how the edge server reroutes requests. If no protocol is specified, however, Adobe Media Server applies the protocol specified in the `RouteTable` element. Implicit proxies hide the routing information from the clients.

The connection syntax for this element is flexible, as demonstrated in the following examples.

Examples

This example shows how you can configure the edge to route all connections to the host `foo` to the host `bar`.

```
<Proxy>
  <RouteTable protocol="">
    <RouteEntry>foo:1935;bar:80</RouteEntry>
  </RouteTable>
</Proxy>
```

Use of the wildcard character `*` to replace host and port. The example shows how to route connections destined for any host on any port to port 1935 on the host `foo`.

```
<RouteEntry>*:*;foo:1935</RouteEntry>
```

The example instructs the server to route connections to any host on any port to the specified host on port 1936. For example, if you were to connect to `foo:1935`, the connection would be routed to `foo:1936`.

```
<RouteEntry>*:*;*:1936</RouteEntry>
```

The example instructs the server to use the values for host and port on the left side as the values for host and port on the right side, and to route connections destined for any host on any port to the same host on port 80.

```
<RouteEntry>*:*;*:80</RouteEntry>
```

The example instructs the server to route a host:port combination to `null`. Its effect is to reject all connections destined for `foo:80`.

```
<RouteEntry>foo:80;null</RouteEntry>
```

See also

[RouteTable](#)

RouteTable

Container element.

```
<RouteTable protocol="rtmp">
```

or

```
<RouteTable protocol="rtmps">
```

The `RouteEntry` elements nested under the `RouteTable` element specify the routing information for the edge server. Administrators use these elements to route connections to the desired destination. The `RouteTable` element can be left empty or it can contain one or more `RouteEntry` elements.

The `protocol` attribute specifies the protocol to use for the outgoing connection. The attribute is set to "" (an empty string), `rtmp` for a connection that isn't secure, or `rtmps` for a secure connection.

- Specifying "" (an empty string) means preserving the security status of the incoming connection.
 - If the incoming connection was secure, then the outgoing connection will also be secure.
 - If the incoming connection was not secure, the outgoing connection will not be secure.
- Specifying `rtmp` instructs the edge not to use a secure outgoing connection, even if the incoming connection was secure.
- Specifying `rtmps` instructs the edge to use a secure outgoing connection, even if the incoming connection was not secure.

You can override the security status for a connection mapping by specifying a `protocol` attribute in a `RouteEntry` element. By default, Adobe Media Server applies the protocol configured in the `RouteTable` list unless the mapping for a particular `RouteEntry` element overrides it.

Contained elements

[RouteEntry](#)

SSL

Container element.

If a virtual host is running in `remote` mode as an edge server and you want to configure the properties of an outgoing SSL connection to an upstream server, then you must enable this section and configure its SSL elements appropriately.

When Adobe Media Server acts as a client to make an outgoing SSL connection, the following sequence of events takes place:

- The SSL elements in the `Vhost.xml` file are evaluated first.
- If the SSL elements in the `Vhost.xml` file override the SSL elements in the `Server.xml` file, Adobe Media Server uses the SSL elements in the `Vhost.xml` file to configure the connection.

- If the `SSL` elements in the `Vhost.xml` file match the `SSL` elements in the `Server.xml` file, Adobe Media Server uses the default values for `SSL` in the `Server.xml` file to configure the connection.
- If the `SSL` elements in an edge's `Vhost.xml` file are not present, Adobe Media Server uses the default values specified in the `SSL` section of `Server.xml` to configure the `SSL` connection to upstream servers.

Note: When Adobe Media Server is running in local mode as an origin server, the `SSL` information in the `vhost.xml` file is not evaluated.

You can also override the configuration for outgoing `SSL` connections for an individual virtual host in `Vhost.xml` by copying the `SSL` elements in `Server.xml` to the corresponding `SSL` section in the `Vhost.xml` file.

For more information on the `SSL` elements in `Server.xml`, see [SSL](#).

Contained elements

[SSLCACertificateFile](#), [SSLCACertificatePath](#), [SSLCipherSuite](#), [SSLVerifyCertificate](#), [SSLVerifyDepth](#)

SSLCACertificateFile

Specifies the name of a file that contains one or more CA (Certificate Authority) digital certificates in PEM (Privacy Enhanced Mail) encryption format.

SSLCACertificatePath

Specifies the name of a directory containing CA certificates. Each file in the directory must contain only a single CA certificate. File names must be the hash with "0" as the file extension.

For Win32 only: If this element is empty, attempts are made to find CA certificates in the `certs` directory located at the same level as the `conf` directory. The Windows certificate store can be imported into this directory by running `AMSMaster - console - initialize` from the command line.

SSLCipherSuite

Specifies the suite of encryption ciphers that the server uses to secure communications.

This element is a colon-delimited list of encryption resources, such as a key-exchange algorithm, authentication method, encryption method, digest type, or one of a selected number of aliases for common groupings. Each item in the cipher list specifies the inclusion or exclusion of an algorithm or cipher. In addition, there are special keywords and prefixes. For example, the keyword `ALL` specifies all ciphers, and the prefix `!` removes the cipher from the list.

The default cipher list instructs the server to accept all ciphers, but block those using anonymous Diffie-Hellman authentication, block low-strength ciphers, block export ciphers, block MD5 hashing, and sort ciphers by strength from highest to lowest level of encryption.

Important: Contact Adobe Support before changing the default settings.

The cipher list consists of one or more cipher strings separated by colons. Commas or spaces are also acceptable separators, but colons are normally used.

The string of ciphers can take several different forms.

- It can consist of a single cipher suite, such as `RC4-SHA`.
- It can represent a list of cipher suites containing a certain algorithm, or cipher suites of a certain type.

For example, `SHA1` represents all cipher suites using the digest algorithm `SHA1`, and `SSLv3` represents all `SSL v3` algorithms.

- Lists of cipher suites can be combined in a single cipher string using the + character as a logical and operation. For example, SHA1+DES represents all cipher suites containing the SHA1 and DES algorithms.
- Each cipher string can be optionally preceded by the characters !, -, or +.
- If ! is used, then the ciphers are permanently deleted from the list. The ciphers deleted can never reappear in the list even if they are explicitly stated.
- If - is used, then the ciphers are deleted from the list, but some or all of the ciphers can be added again later.
- If + is used, then the ciphers are moved to the end of the list. This option doesn't add any new ciphers—it just moves matching existing ones.
- If none of these characters is present, then the string is just interpreted as a list of ciphers to be appended to the current preference list.
- If the list includes any ciphers already present, the server does not evaluate them.
- The cipher string @STRENGTH sorts the current cipher list in order of the length of the encryption algorithm key.

The components can be combined with the appropriate prefixes to create a list of ciphers, including only those ciphers the server is prepared to accept, in the order of preference.

Example

This cipher string instructs the server to accept all ciphers except those using anonymous or ephemeral Diffie-Hellman key exchange.

```
<SSLCipherSuite>ALL:!ADH:!EDH</SSLCipherSuite>
```

These cipher strings instruct the server to accept only RSA key exchange and refuse export or null encryption. The server evaluates both strings as equivalent.

```
<SSLCipherSuite>RSA:!NULL!EXP</SSLCipherSuite>
<SSLCipherSuite>RSA:LOW:MEDIUM:HIGH</SSLCipherSuite>
```

This cipher list instructs the server to accept all ciphers but place them in order of decreasing strength. This sequencing allows clients to negotiate for the strongest cipher that both they and the server can accept.

```
<SSLCipherSuite>ALL:+HIGH:+MEDIUM:+LOW:+EXP:+NULL</SSLCipherSuite>
```

This string instructs the server to accept only high- and medium-strength encryption, with the high being preferred, and reject export-strength versions.

```
<SSLCipherSuite>ALL:+HIGH:!LOW:!EXP:!NULL</SSLCipherSuite>
```

This string instructs the server to accept all ciphers but to order them so that SSLv2 ciphers come after SSLv3 ciphers.

```
<SSLCipherSuite>ALL:+SSLv2</SSLCipherSuite>
```

The following is the complete list of components that the server can evaluate.

Key exchange algorithm	Description
kRSA	Key exchange
kDhr	Diffie-Hellman key exchange with RSA key
kDHd	Diffie-Hellman key exchange with DSA key
RSA	Ephemeral Diffie-Hellman key exchange

Key exchange algorithm	Description
DH	RSA key exchange
EDH	Ephemeral Diffie-Hellman key exchange
ADH	Anonymous Diffie-Hellman key exchange

Authentication methods	Description
aNULL	No authentication
aRSA	RSA authentication
aDSS	DSS authentication
aDH	Diffie-Hellman authentication

Encryption methods	Description
eNULL	No encoding
DES	DES encoding
3DES	Triple-DES encoding
RC4	RC4 encoding
RC2	RC2 encoding
IDEA	IDEA encoding
NULL	No encryption
EXP	All export ciphers (40-bit encryption)
LOW	Low-strength ciphers (no export, DES)
MEDIUM	128-bit encryption
HIGH	Triple-DES encoding

Digest types	Description
MD5	MD5 hash function
SHA1	SHA1 hash function
SHA	SHA hash function

Additional aliases	Description
All	All ciphers
SSLv2	All SSL version 2.0 ciphers
SSLv3	All SSL version 3.0 ciphers
DSS	All ciphers using DSS authentication

SSLVerifyCertificate

Specifies whether the certificate returned by the server should be verified. Certificate verification is enabled by default. To disable certificate verification, specify `false`.

Note: Disabling certificate verification can result in security problems.

SSLVerifyDepth

Specifies the maximum depth of the certificate chain to accept. If a self-signed root certificate cannot be found within this depth, certificate verification fails. The default value is 9.

Example

```
<SSLVerifyDepth>9</SSLVerifyDepth>
```

Streams

Specifies the virtual directory mapping for recorded streams. The `Streams` element enables you to specify a virtual directory for stored stream resources used by more than one application. By using a virtual directory, you specify a relative path that points to a shared directory that multiple applications can access.

You can specify multiple virtual directory mappings for streams by adding additional `Streams` elements—one for each virtual directory mapping.

For more information, see the “Configuring content storage” section of the *Configuration and Administration Guide*.

Examples

The following configuration maps all streams whose names begin with `foo/` to the physical directory `c:\data`. The stream named `foo/bar` maps to the physical file `c:\data\bar.flv`.

```
<Streams>foo;c:\data</Streams>
```

If a stream is named `foo/bar/x`, the server tries to find a virtual directory mapping for `foo/bar`. If there is no virtual directory for `foo/bar`, the server checks for a virtual directory mapping for `foo`. Since a virtual directory mapping does exist for `foo`, the stream `foo.bar` maps to the file `c:\data\bar\x.flv`.

Note: If the virtual directory you specify does not end with a backslash, the server adds one.

The following configuration maps streams whose paths begin with `common/` to the folder `C:\flashmediaserver\myapplications\shared\resources`.

```
<Streams>common;C:\flashmediaserver\myapplications\shared\resources</Streams>
```

If the application “videoConference” refers to an item `common/video/recorded/june5` and the application “collaboration” refers to `common/video/recorded/june5`, they both point to the same item `C:\flashmediaserver\myapplications\shared\resources\video\recorded\june5`.

See also

[VirtualDirectory](#)

VirtualDirectory

Specifies virtual directory mappings for resources such as recorded streams.

Virtual directories let you share resources among applications. When the beginning portion of a resource's URI matches a virtual directory, Adobe Media Server serves the resource from the physical directory. For detailed information on mapping virtual directories, see [“Mapping virtual directories to physical directories”](#) on page 51.

You can use the `VirtualDirectory` element in conjunction with the `VirtualKeys` element to serve content based on Flash Player version information. For more information, see [VirtualKeys](#).

Note: *If you are mapping a virtual directory to a drive on another computer, make sure that the computer running Adobe Media Server has the right permissions to access the other computer. For more information, see [“Mapping directories to network drives”](#) on page 50.*

Example

For example, using the following `VirtualDirectory` XML, if a client called `NetStream.play("vod/myVideo")`, the server would play the file `d:\sharedStreams\myVideo.flv`:

```
<VirtualDirectory>
  <Streams>vod;d:\sharedStreams</Streams>
</VirtualDirectory>
```

Contained elements

[Streams](#)

See also

[VirtualKeys](#)

VirtualHost

Root element of the `Vhost.xml` file.

This element contains all the configuration elements for the `Vhost.xml` file.

VirtualKeys

Lets you map Flash Player versions to keys. The keys are used in the [VirtualDirectory](#) element to map URLs to physical locations on a server. Use these elements to deliver streams to clients based on Flash Player version.

Contained elements

[Key](#)

WaitTime

Specifies length to wait in milliseconds for edge autodiscovery. The number must be long enough to establish a TCP connection, perform a UDP broadcast, collect the UDP responses, and return an XML response. Do not set this number too low.

Example

```
<WaitTime>1000</WaitTime>
```

See also

[Enabled](#), [AllowOverride](#)

Chapter 8: Diagnostics Log Messages

Message IDs in diagnostic logs

The IDs of messages that appear in the diagnostic log files (master.xx.log, edge.xx.log, core.xx.log, admin.xx.log, and httpcache.xx.log) record information about server operations. Message IDs can be useful for administrators who want to write error-handling scripts. For status codes related to applications, instances, or users that you can use for debugging, see “[Access logs](#)” on page 98 and “[Application logs](#)” on page 105.

Message ID	Description
1000	Received termination signal; server shutdown in progress.
1001	Received interrupt signal; server shutdown in progress.
1002	Server initialization failed; service will be stopped.
1003	Error during shutdown process; process will be terminated.
1004	Reinitializing server.
1005	Failed to start the following listeners for adaptor %1\$: %2\$.
1006	Failed to stop %1\$ listeners for adaptor %2\$.
1007	Failed to create thread (%1\$).
1008	Asynchronous I/O operation failed (%1\$: %2\$).
1009	Service Control Manager failed (%1\$: %2\$).
1010	Service Control Manager reported (%1\$: %2\$).
1011	Server starting...
1012	Server stopped %1\$.
1013	Failed to create listener for adaptor %1\$, IP %2\$, port %3\$: %4\$.
1014	Command name not found in the message.
1015	Method not found (%1\$).
1016	Failed to execute method (%1\$).
1017	Failed to stop virtual host (%1\$).
1018	The call method failed, invalid parameters: call(methodName[, resultObj, p1, pn]).
1019	Dropping application (%1\$) message. Clients not allowed to broadcast message.
1020	Response object not found (%1\$).
1021	Missing unlock for shared object %1\$, lock count %2\$.
1022	Nested lock for shared object %1\$, lock count %2\$.
1023	Unlock called without matching lock for shared object %1\$.
1024	Invalid application; rejecting message (%1\$).
1025	Ignoring message from client during authentication.

Message ID	Description
1026	Connection to %1\$S lost.
1027	Unknown %1\$S command issued for stream %2\$S (application %3\$S).
1028	Exception while processing message.
1029	Bad network data; terminating connection: %1\$S.
1030	Illegal subscriber: %1\$S cannot subscribe to %2\$S.
1031	Failed to start virtual host (%1\$S).
1032	Failed to open configuration file: %1\$S.
1033	Parse error at line %1\$S: %2\$S.
1034	Connect failed (%1\$S, %2\$S): %3\$S.
1035	Invalid proxy object; connection may be lost (%1\$S).
1036	Connect from host (%1\$S) not allowed.
1037	No adaptors defined.
1038	Adaptor already defined with the name %1\$S.
1039	Rejecting connection from %1\$S to %2\$S.
1040	Failed to create administrator: %1\$S.
1041	Failed to remove administrator: %1\$S.
1042	Failed to change password: %1\$S.
1043	Resource limit violation. Unable to create stream: %1\$S.
1044	Resource limit violation. Unable to create shared object: %1\$S.
1045	Script execution is taking too long.
1046	Reserved property (%1\$S).
1047	Admin request received from an invalid Administration Server.
1048	Administrator login failed for user %1\$S.
1049	Failed to start server.
1050	Write access denied for shared object %1\$S.
1051	Read access denied for shared object %1\$S.
1052	Write access denied for stream %1\$S.
1053	Read access denied for stream %1\$S.
1054	Virtual host %1\$S is not available.
1055	Invalid parameters to %1\$S method.
1056	Alive.
1057	NetConnection.Call.Failed
1058	Invalid application name (%1\$S).
1059	Invalid user ID (%1\$S).

Message ID	Description
1060	NetConnection.Admin.CommandFailed
1061	Invalid parameters to %1\$\$ method.
1062	Failed to unload application %1\$\$.
1063	Failed to load application %1\$\$.
1064	%1\$\$ applications unloaded.
1065	Admin user requires valid user name and password.
1066	Invalid virtual host alias : %1\$\$.
1067	Error registering class: name mismatch (%1\$\$, %2\$\$).
1068	Connection rejected: maximum user limit reached for application instance %1\$\$.
1069	(%2\$\$, %3\$\$) : Failed to load application instance %1\$\$.
1070	(%2\$\$, %3\$\$) : Connection rejected to application instance %1\$\$. Client already connected to an application.
1071	Illegal access property (%1\$).
1072	%1\$\$ is now published.
1073	%1\$\$ is now unpublished.
1074	Stopped recording %1\$\$.
1075	Stream %1\$\$ has been idling for %2\$\$ second(s).
1076	Playing and resetting %1\$\$.
1077	Pausing %1\$\$.
1078	Unpausing %1\$\$.
1079	Started playing %1\$\$.
1080	Stopped playing %1\$\$.
1081	Recording %1\$\$.
1082	Failed to record %1\$\$.
1083	New NetStream created (stream ID: %1\$).
1084	NetStream deleted (stream ID: %1\$).
1085	Publishing %1\$\$.
1086	Failed to publish %1\$\$.
1087	Failed to restart virtual host (%1\$).
1088	Connection to Adobe Media Server has been disconnected.
1089	Failed to play (stream ID: %1\$).
1090	Failed to play %1\$\$ (stream ID: %2\$).
1091	Play stop failed, stream ID: %1\$.
1092	Audio receiving enabled (stream ID: %1\$).
1093	Audio receiving disabled (stream ID: %1\$).

Message ID	Description
1094	Failed to enable audio receiving (stream ID: %1\$S).
1095	Failed to stop playing (stream ID: %1\$S).
1096	Video receiving enabled (stream ID: %1\$S).
1097	Video receiving disabled (stream ID: %1\$S).
1098	Set video fps to %1\$S (stream ID: %2\$S).
1099	Failed to receive video (stream ID: %1\$S).
1100	Seeking %1\$S (stream ID: %2\$S).
1101	Failed to seek (stream ID: %1\$S).
1102	Failed to seek %1\$S (stream ID: %2\$S).
1103	Invalid schedule event format (%1\$S).
1104	Invalid method name (%1\$S).
1105	(%2\$S, %3\$S): Invalid application name (%1\$S).
1106	Connection succeeded.
1107	Connection failed.
1108	Invalid shared object (%1\$S).
1109	Unknown exception caught in %1\$S.
1110	Invalid stream name (%1\$S).
1111	Server started (%1\$S).
1112	JavaScript runtime is out of memory; server shutting down instance (Adaptor: %1\$S, VHost: %2\$S, App: %3\$S). Check the JavaScript runtime size for this application in the configuration file.
1113	JavaScript engine runtime is low on free memory. Take action.
1114	Failed to start listeners for adaptor %1\$S.
1115	Configuration error for adaptor %1\$S: IP %2\$S and port %3\$S are already in use.
1116	Failed to create adaptor: %1\$S.
1117	Failed to play %1\$S; stream not found.
1118	Insufficient admin privileges to perform %1\$S command.
1119	Failed to initialize listeners for adaptor %1\$. Adobe Media Server is already running or other processes are using the same ports.
1120	Configuration file not found: %1\$S
1121	Invalid configuration file: %1\$S
1122	Server aborted.
1123	Invalid NetStream ID (%1\$S).
1124	Failed to open shared object file (%1\$S) for write.
1125	Failed to open shared object file (%1\$S) for read.
1126	Failed to flush shared object (%1\$S).

Message ID	Description
1127	Failed to initialize shared object from persistent store (%1\$S).
1128	Invalid shared object file (%1\$S).
1129	Failed to play %1\$S; index file not found or mismatch.
1130	(%2\$S, %3\$S): Application (%1\$S) is not defined.
1131	(%2\$S, %3\$S): Resource limit violation. Unable to load new application: %1\$S.
1132	(%2\$S, %3\$S): Resource limit violation. Unable to create new application instance: %1\$S.
1133	(%2\$S, %3\$S): Resource limit violation. Rejecting connection to: %1\$S.
1134	Failed to load admin application.
1135	Preload application aborted.
1136	(%2\$S, %3\$S): Application (%1\$S) is currently offline.
1137	Admin command setApplicationState failed for %1\$S.
1138	Command successful.
1139	Script is taking too long to process the event. Shutting down instance: %1\$S.
1140	NetConnection.Call.Success
1141	Unable to locate server configuration file during startup.
1142	Unable to locate script file: %1\$S.
1143	NetConnection.Call.AccessDenied
1144	NetConnection.Call.BadValue
1145	Publish %1\$S failed, invalid arguments.
1146	Pause %1\$S failed, invalid arguments.
1147	Unable to create directory %1\$S.
1148	Server shutdown failed.
1149	Invalid admin command: %1\$S.
1150	Beta expired.
1151	Invalid object name (stream ID: %1\$S).
1152	Breaking potential deadlock, shared object(%1\$S) lock reset to unlocked.
1153	Potential deadlock, shared object (%1\$S) has been locked for %2\$S sec.
1154	Invalid license key: %1\$S
1155	License key specified does not allow multiple adaptor support.
1156	License key specified does not allow multiple virtual host support.
1157	(%2\$S, %3\$S/%1\$S): Current server bandwidth usage exceeds license limit set. Rejecting connection.
1158	(%2\$S, %3\$S/%1\$S): Current virtual host bandwidth usage exceeds max limit set. Rejecting connection.
1159	Multiprocessor support available only in Enterprise Edition.
1160	Trial run expired. Server shutting down.

Message ID	Description
1161	License key has expired.
1162	Invalid shared object name (%1\$S).
1163	Failed to record %1\$S, no space left on device.
1164	Unknown exception occurred. Instance will be unloaded: %1\$S.
1165	Failed login attempt from %1\$S at %2\$S.
1166	Attempt to reconnect to Adobe Media Server.
1167	Failed to remove application: %1\$S.
1168	Exception while processing message: %1\$S.
1169	Failed to execute admin command: %1\$S.
1170	Unloaded application instance %1\$S.
1171	System memory load (%1\$S) is high.
1172	System memory load (%1\$S) is now below the maximum threshold.
1173	<i>Generic message code.</i>
1174	Listener started (%1\$S): %2\$S.
1175	Restarting listener (%1\$S): %2\$S.
1176	Out of memory: %1\$S.
1177	Adaptor (%1\$S) has an SSL configuration error on port %2\$S.
1178	Error from %1\$S:%2\$S.
1179	Warning from %1\$S:%2\$S.
1180	Info from %1\$S:%2\$S.
1181	Exception caught in %1\$S while processing streaming data inside %2\$S.
1182	(%2\$S, %3\$S): Max connections allowed exceed license limit. Rejecting connection to: %1\$S.
1183	An internal version control error has occurred.
1184	Invalid cryptographic accelerator: %1\$S.
1185	Failed to initialize cryptographic accelerator: %1\$S.
1186	Failed to seed the pseudorandom number generator.
1187	Application directory does not exist: %1\$S
1188	Using default application directory: %1\$S
1189	Application instance is not loaded: %1\$S
1190	Error: command message sent before client connection has been accepted.
1191	Failed to play %1\$S; adaptor not found: %2\$S.
1192	Invalid value set for configuration key: %1\$S = %2\$S, using %3\$S.
1193	Pending queue size limit %1\$S reached. Rejecting connection request Host: %2\$S:%3\$S.
1194	Client to server bandwidth limit exceeded. [Virtual host (%1\$S), Max Allowed %2\$S, Current %3\$S]

Message ID	Description
1195	Server to client bandwidth limit exceeded. [Virtual host (%1\$S), Max Allowed %2\$S, Current %3\$S]
1196	Adaptor (%1\$S) does not exist.
1197	Virtual host (%1\$S) does not exist.
1198	Message queue is too large. Server memory usage too high. Disconnecting client.
1199	Duplicate license key: %1\$S
1200	Expired license key: %1\$S
1201	No primary license key found. Switching to Developer Edition.
1202	Commercial and Educational licenses cannot be mixed. Switching to Developer Edition.
1203	Personal and Professional licenses cannot be mixed. Switching to Developer Edition.
1204	NFR licences cannot be mixed with any other kind. Switching to Developer Edition.
1205	OEM licences cannot be mixed with any other kind. Switching to Developer Edition.
1206	Too many trial licenses detected. Switching to Developer Edition.
1207	Shared object %1\$S has changed and is not being saved, as auto commit is set to false. Current version %2\$S, Last saved version %3\$S.
1208	%1\$S failed. Invalid argument %2\$S.
1209	File operation %1\$S failed. %2\$S
1210	File operation %1\$S failed. File is in closed state (%2\$S).
1211	File operation %1\$S failed. Object is not a file (%2\$S).
1212	File object creation failed (%1\$S).
1213	Connection rejected by server. Reason: %1\$S.
1214	Invalid substitution variable: %1\$S
1215	Resetting service failure action from %1\$S to %2\$S.
1216	Administrator (%1\$S) already exists.
1217	Failed to open log file. Log aborted.
1218	Failed to play stream %1\$S: Recorded mode not supported.
1219	Missing arguments to %1\$S method.
1220	Invalid admin stream: %1\$S .
1221	Core (%1\$S) started, arguments: %2\$S.
1222	Failed to start core: %1\$S %2\$S.
1223	Core (%1\$S) is no longer active.
1224	Edge (%1\$S) started, arguments: %2\$S.
1225	Failed to start edge: %1\$S %2\$S.
1226	Edge (%1\$S) is no longer active.
1227	Shared memory heap (%1\$S) has exceeded 90 usage. Consider increasing the heap size to prevent future memory allocation failures.

Message ID	Description
1228	Failed to create process mutex.
1229	Process (%1\$S): shared memory (%2\$S) init failed.
1230	Process (%1\$S): failed to map view of shared memory (%2\$S).
1231	Core (%1\$S) connected to admin.
1132	Core (%1\$S) failed to connect to admin.
1233	Core (%1\$S) disconnecting from admin.
1234	Core (%1\$S) connection to admin accepted.
1235	Core (%1\$S) connection to admin failed.
1236	Core (%1\$S) received close command from admin.
1237	Starting admin app on core (%1\$S).
1238	Core (%1\$S) connecting to admin.
1239	Core (%1\$S): Failed to initiate connection to admin.
1240	Core (%1\$S) shutdown failed.
1241	Connection to admin received.
1242	Core (%1\$S) disconnected: %2\$S.
1243	Connection from core %1\$S received.
1244	Connection from core %1\$S accepted.
1245	Failed to send connect response to core %1\$S.
1246	Core (%1\$S) sending register command to edge.
1247	Core (%1\$S) disconnected from edge.
1248	Core (%1\$S) failed to establish proxy to edge.
1249	Core (%1\$S) socket migration failed.
1250	Edge disconnected from core (%1\$S).
1251	Proxy to core (%1\$S) failed.
1252	Registering core (%1\$S).
1253	Socket migration to core (%1\$S) failed.
1254	Recovering edge %1\$S with %2\$S failure[s] after %3\$S seconds!
1255	Edge (%1\$S) %2\$S experienced %3\$S failure[s]!
1256	Core (%1\$S) %2\$S experienced %3\$S failure[s]!
1257	Core (%1\$S) %2\$S is not responding and is being restarted!
1258	Core (%1\$S) is no longer active; create a new one.
1259	Recovering core %1\$S with %2\$S failure[s] after %3\$S seconds!
1260	Core (%1\$S) did not shut down as expected. Killing core now.
1261	Command (%1\$S) timed out.

Message ID	Description
1262	OpenProcess(PROCESS_TERMINATE) failed with %1\$.
1263	OpenProcess(PROCESS_QUERY_INFORMATION) failed for pid (%1) with %2\$.
1373	SWF verification failed, disconnecting client.
1374	SWF verification timeout, disconnecting client.
1375	SWF verification unsupported by client, disconnecting client.