ADOBE® FLASH® MEDIA SERVER

CONFIGURATION AND ADMINISTRATION GUIDE

Adobe®

Adobe® Flash® Media Server Configuration and Administration Guide

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

Chapter 8: XML configuration files reference

Chapter 9: Diagnostic Log Messages

Chapter 1: Before you begin

Overview of Flash Media Server

Adobe® Flash® Media Server provides streaming media capabilities and a scripting engine that enable you to create and deliver a wide range of interactive media applications. Use Flash Media Server to create traditional media delivery applications, such as video on demand, live web event broadcasts, or MP3 streaming. You can also design media applications like video blogging, video messaging, and multimedia chat environments. Flash Media Server is part of Adobe's complete solution for database connectivity, directory systems, presence services, and audio and video delivery to Flash Player.

About the documentation

All documents are available in LiveDocs and PDF formats. Some documents are available for installation in the Flash and Flex Help panels.

Flash Media Server includes the following documentation:

- *Adobe Flash Media Server Installation Guide* describes system requirements, server editions, and installation profiles and explains how to install the server as either an origin or an edge server.
- *Adobe Flash Media Server Technical Overview* describes the server architecture including new features, the client-server relationship, edge servers, and security features.
- *Adobe Flash Media Server Configuration and Administration Guide* (this manual) describes how to deploy, configure, and tune the server, how to use the Administration Console to monitor the server, and how to use the Administration application programming interface (API) to monitor and configure the server.
- *Adobe Flash Media Server Developer Guide* explains how to set up your development environment. It also describes how to use the Flash authoring environment, the Flex authoring environment, and the Flash Media Server API to create media applications.
- *Adobe Flash Media Interactive Server Plug-in Developer Guide* documents how to create Access, Authorization, and File plug-ins in C++ that extend the capabilities of the server.
- *ActionScript 3.0 Language and Components Reference* documents the version 3.0 ActionScript™ you can use to create client-side functionality. This document is part of the Flash or Flex documentation set, depending on which authoring tool you use.
- *Adobe Flash Media Server ActionScript 2.0 Language Reference* documents the version 2.0 ActionScript you can use to create client-side functionality. This document contains additional APIs and information about calling Flash Media Server resources from a Flash Player client. You may also need to use the Flash ActionScript documentation to create Flash Media Server client applications.
- *Server-Side ActionScript Language Reference for Adobe Flash Media Server* documents the Server-Side ActionScript you can use to write scripts on the server. Server-Side ActionScript is JavaScript 1.5 with additional classes that work only in the Flash Media Server host environment.
- *Adobe Flash Media Server Administration API Reference* documents the ActionScript API you can use to extend the Flash Media Server Administration Console or to make your own administration and monitoring tools.

- *Adobe Flash Media Interactive Server Plug-in API Reference* documents the C++ APIs you use to create plug-ins.

Support

You may want to explore these other sources of support for Flash Media Server:

- The Flash Media Server Support Center at www.adobe.com/go/flashmediaserver_support_en provides TechNotes and up-to-date information about Flash Media Server.
- The Flash Media Server Developer Center at www.adobe.com/go/flashmediaserver_desdev_en provides tips and samples for creating Flash Media Server applications.
- The Flash Media Server Online Forum at www.adobe.com/go/flashmediaserver_forum_en provides a place for you to chat with other Flash Media Server users.
- For late-breaking information and a complete list of issues that are still outstanding, read the Flash Media Server release notes at www.adobe.com/go/flashmediaserver_releasenotes_en.

Third-party resources

Adobe recommends several websites with links to third-party resources on Flash Media Server, including the following:

- Adobe Flash community sites
- Adobe Flash books
- Object-oriented programming concepts

You can access these websites at www.adobe.com/go/flashmediaserver_resources_en.

Typographical conventions

The following typographical conventions are used in this manual:

- `Code font` indicates ActionScript statements, HTML tag and attribute names, and literal text used in examples.
- *`Italic`* indicates placeholder elements in code or paths. For example, `attachAudio(`*`source`*`)` means that you should specify your own value for *`source`*; /settings/*myPrinter*/ means that you should specify your own location for *myPrinter*.
- Directory paths are written with forward slashes (/). If you are running Flash Media Server on a Windows operating system, replace the forward slashes with backslashes. When a path is specific to the Windows operating system, backslashes (\) are used.

Chapter 2: Deploying the server

Deploying servers in a cluster

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.

1. **Install Flash Media Server and verify the installation on each computer.**

   Ensure that you deploy all servers on computers that meet the minimum system requirements. For information about installing and verifying installation, see the *Installation Guide*.

   Note: *Use the same operating system (Linux or Windows) on all computers to avoid conflicts with filenames. Filenames in Linux are case-sensitive; filenames in Windows are case-insensitive.*

2. **Configure a load balancer to see the servers hosting Flash Media Interactive Server or Flash Media Streaming Server.**

   See Clustering multiple servers behind a load balancer.

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 tunneling 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 tunneling connections, cookies are currently supported only on Flash Player 9.0.28 or later in Windows only.*

For more information, see the following articles:

- www.adobe.com/go/learn_fms_redundancy_en
- www.adobe.com/go/learn_fms_clustering_en
- www.adobe.com/go/learn_fms_scalable_en

Deploying edge servers

Note: *Only Flash Media Interactive Server can be configured as an edge server. Flash Media Streaming Server and Flash Media Development Server cannot be configured as edge servers.*

The default configuration for Flash Media Server is that of an origin server. Typically, to deploy edge servers, you would run Flash Media Server as an origin on one instance of the server, and deploy an edge server on another instance.

While it is possible for development scenarios to run the server in hybrid mode—that is, to configure some virtual hosts to run as an edge and another to run as an origin—this is not a typical production scenario.

Workflow for deploying edge servers

1. **Install Flash Media Interactive Server 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: *Use the same operating system (Linux or Windows) on all computers to avoid conflicts with filenames. Filenames in Linux are case-sensitive; filenames in Windows are case-insensitive.*

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. For more information, see Configure edge servers.

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. See Connect to an edge server.

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 installed with Flash Media Server, or see the description of the `RouteEntry` element in Configure edge servers.

Configure edge servers

To configure Flash Media Interactive 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. 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*.

Note: *For information about virtual hosts, see Configuring adaptors, virtual hosts, and applications. For information about using configuration files, see Working with configuration files. For detailed information about XML elements in the configuration files, see XML configuration files reference.*

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>local</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 `local` to configure Flash Media Interactive Server to run as an origin server. Enter `remote` to configure Flash Media Interactive Server to run as an edge server.
`Anonymous`	Optional	A Boolean value specifying whether the edge server connection is implicit (`true`) or explicit (`false`). The default value is `false`. For more information, see Connect to an edge server.
`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: `enabled` and `appname`. To enable caching, set the `enabled` attribute to `"true"`. When enabled, streams are placed by default in the *RootInstall*/cache/*appName* directory. You can specify a different cache location in this tag. The `useAppName` attribute indicates 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.

Element	Required/optional	Description
`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 RouteEntry element, how to route connections from the origin to the edge. There is one attribute, `protocol`, that indicates the protocol of the outgoing connection. Set this attribute to either `"rtmp" or "rtmps"`. To override the `RouteTable` protocol for a specific `RouteEntry` element, add a `protocol` attribute to the `RouteEntry` element you want to change.
`RouteEntry`	Optional	Each `RouteEntry` element maps a host/port pair to a different host/port pair: *`host1:port1;host2:port2`* Connections to *`host1:port1`* are routed to *`host2:port2`*. Typically, *`host1:port1`* is your origin server and *`host2:port2`* is your edge server. For example: `<RouteEntry>edge:1935;origin:80</RouteEntry>` routes connections destined for host "edge" on port 1935 to host "origin" on port 80. You can specify a wildcard character (*) for any host or port. For example: `<RouteEntry>*:*;origin:1935</RouteEntry>` routes connections destined for any host on any port to host "origin" on port 1935. You can also specify a wildcard for the host/port to which connections are being routed. For example: `<RouteEntry>*:*;*:80</RouteEntry>` routes connections destined for any host on any port to the same host on port 80. To reject connections, you can specify that a host/port combination be routed to null: `<RouteEntry>edge:80;null</RouteEntry>` The RouteEntry element has a `protocol` attribute. This attribute overrides the `RouteTable` protocol for a specific `RouteEntry` element. For example, `RouteTable` may have one `RouteEntry` element that specifies an encrypted outgoing RTMPS connection, and another `RouteEntry` tag that specifies the regular RTMP connection. If a protocol 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 Flash Media Interactive Server.

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, if your applications were running on fms.foo.com, instead of clients connecting to an application with a connection string such as `rtmp://fms.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://fmsedge.foo.com/?rtmp://fms.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 Flash Media Server.

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

```
rtmp://origin/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);
}
```

Chapter 3: Configuring the server

Configuring adaptors, virtual hosts, and applications

Adaptors and virtual hosts

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.

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*.

Applications

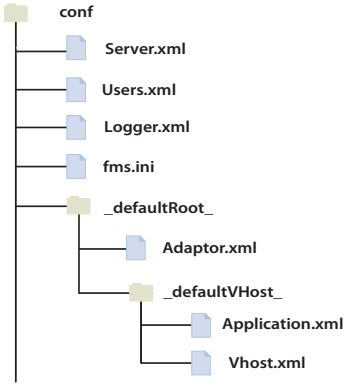
Application files (SWF, HTML, FLA) must be stored in an applications folder. The applications folder registers applications with the server.

By default, the location of the applications folder is *RootInstall*/applications. For example, the live and vod applications that come with Flash Media Server are installed at *RootInstall*/applications/live and *RootInstall*/applications/vod, respectively. You can change the default location of the applications folder and of the live and vod applications in particular; see Setting the location of application files.

You create instances of applications by creating subfolders within the application's folder. For example, to create an instance of the vod application called room1, create a *RootInstall*/applications/vod/room1 folder.

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 fms.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 fms.ini, you also must restart Flash Media Administration Server. For more information, see Working with configuration files.

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, fms.ini, in the *RootInstall*/conf directory. This file contains commonly used settings, including the administrator user name and password and the settings you chose during installation.
- 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 adaptor 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">:1936</HostPort>
```

The `name` attribute must be unique on the server. The control channel (`ctl_channel`) attribute and the `HostPort` value specify the ports to which an IP address should bind. If an IP address is not specified, the adaptor can listen 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 fmsedge 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 interoperate. Ensure that the control channels on which separate adaptors listen are different from each other, as in the following example:

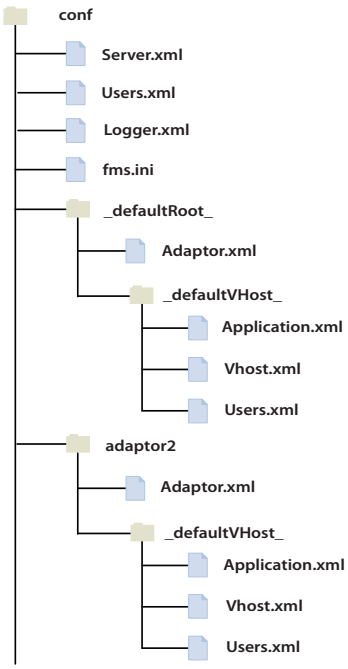
```
<HostPort name="edge1" ctl_channel=":19350">xx.xx.xx.xx:1935</HostPort>
<HostPort name="edge2" ctl_channel=":19351">yy.yy.yy.yy:1935</HostPort>
```

5 Restart the server.

6 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.

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>
```

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.

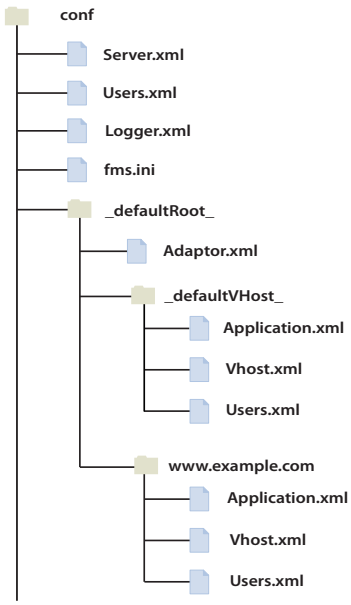
Note: *You can call the* startVHost() *Administration API or log in to the Administration Console without restarting the server.*

6 Log in to the Administration Console.

For information about logging in to the Administration Console, see Connecting to the Administration Console.

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.

Working with configuration files

Editing configuration files

Note: *For information about configuration file names, locations, and hierarchy, see Configuration folder structure.*

To edit a configuration file, open it in a text editor, modify and save it, and restart the server. If you modify Users.xml, you also must restart Flash Media Administration Server.

It's a good idea to check that the XML is valid after editing an XML configuration file.

Edit the fms.ini file

1 Open *RootInstall*/conf/fms.ini in a text editor.

2 Save a copy to another location as a backup.

3 Enter a new value for the `SERVER.ADMIN_PASSWORD` parameter.

4 Save the file and restart the server.

5 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 fms.ini file. When the server starts, it looks for the fms.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 FMSMaster.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 fms.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

Configure the recorded media cache

In some scenarios, you might want to increase the size of the recorded media cache. When a stream is requested from the server, segments of the stream are stored in a memory 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.

1 Open the *RootInstall*/conf/fms.ini file.

2 Edit the `SERVER.FLVCACHE_MAXSIZE` parameter.

This is the maximum size of the cache, in megabytes. The default value is 500. This value shares memory with the running process and has a limit of 2 GB in Windows and 3 GB in Linux.

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.

Configure the size of stream chunks

In some scenarios, you might want to change the size of stream chunks. Stream content breaks into chunks as it's written to the network. You can specify the size of an RTMP chunk. Larger values reduce CPU usage, but also commit to larger writes that 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.

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.

3 Restart the server.

Configure the size of stream chunks for the vod service

In some scenarios, you might want to change the size of stream chunks. Stream content breaks into chunks as it's written to the network. You can specify the size of an RTMP chunk for the vod service. Larger values reduce CPU usage, but also commit to larger writes that can delay other content on lower bandwidth connections.

1 Open the Application.xml file for the vod application.

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 the Application.xml file.

3 Restart the server.

Send aggregate messages

In some scenarios, you might want to enable the server to aggregate messages before sending them. An application can be configured 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. Sending aggregate messages reduces CPU usage and increases server capacity.

1 Open the Application.xml file.

2 Locate the following XML:

```
<Client>
    ...
    <AggregateMessages enabled="true"></AggregateMessages>
<Client>
```

3 Make sure that the `enabled` attribute to `true` (the default).

4 Validate the XML and save the Application.xml file.

5 Restart the server.

Combine audio samples

In some scenarios, to handle more connections while broadcasting a live stream, combine audio samples.

Important: *Do not combine audio samples if you are also using the live aggregation message feature.*

1 Open the *RootInstall*/conf/fms.ini file.

2 Edit the following parameters:

- `APP.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.
- `APP.COMBINESAMPLES_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.
- `APP.COMBINESAMPLES_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.
- `APP.COMBINESAMPLES_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

In some cases, 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>100</MaxConnectionRate>
```

Element	Description	Impact
`MaxConnectionRate`	The maximum number of incoming connections per second the server accepts, per listener. Listeners are defined in the `HostPort` element in the Adaptor.xml 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. 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 `MaxConnectionRate` is set to 10, the server imposes a maximum total combined rate of 30 connections per second.	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 users abandon their connections to an application. To reclaim these resources for new and active clients, the server can close the idle clients.

A client is active when it is sending (e.g., publishing) or receiving (e.g., subscribing to) data. 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, 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 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
`AutoCloseIdleClients`	Set the `enable` attribute to `true` to close idle clients. If the `enable` attribute is omitted or not set to `true`, the feature is disabled. The default value is `false`.	
`CheckInterval`	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 `MaxIdleTime` value. A shorter interval results in more reliable disconnection times, but can also result in decreased server performance.
`MaxIdleTime`	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 the type of applications running on the server. For example, if you have an application allowing users to watch short video clips, a user might leave the window to idle out.

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 `enable` attribute to `false`. If this element is disabled in Server.xml, the feature is disabled for all virtual hosts, even if you specify `true` in the Vhost.xml 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 Server.xml. file. The value of the `MaxIdleTime` element in the Vhost.xml file overrides the value of the `MaxIdleTime` element in the Server.xml 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 `enable` attribute to `false`. If this element is disabled in Server.xml, the feature is disabled for all applications, even if you specify `true` in the Application.xml 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 Vhost.xml. file. 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 of the `MaxIdleTime` element in the Vhost.xml file overrides the value of the `MaxIdleTime` element in the Server.xml file.

3 Restart the server.

Configure how applications are assigned to server processes

Note: *This section is only applicable to Flash Media Interactive Server and Flash Media Development Server. Flash Media Streaming Server 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 FMSMaster.exe (Windows) or fmsmaster (Linux). Application instances run in processes called FMSCore.exe (Windows) and fmscore (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. However, you shouldn't run more than 100, and you probably won't need more than 20.*

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 Distribute numprocs 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 one core process (<Scope>app</Scope>) or three 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.

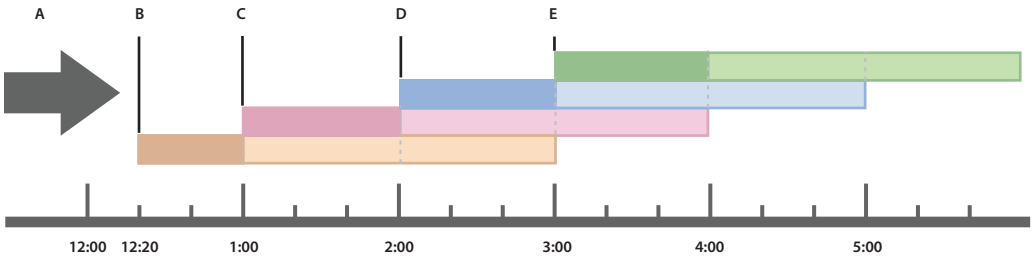
Value	Description
vhosts	All instances of applications in a virtual host run together in a process.
apps	All instances of an application run together in a process.
insts	Each application instance runs in its own process. This is the default value. If you choose this value, you must also set the Distribute numprocs attribute to a value greater than 1.
clients	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 client, 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, the 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 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 consume the 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.*

Configuring security features

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/fms.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.

3 Restart the server.

Verify SWF files

If desired, you can configure the server to verify client SWF files before allowing them to connect to an application. Verifying SWF files prevents someone from creating their own SWF files that attempt to stream your resources.

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

In the Application.xml file, specify one or more folders on the server to hold a copy of an application's client SWF file (this is the *verifying SWF file*). 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. You can also configure the length of time the verification data is held in the server's cache and how often the server checks for updated verifying SWF files.

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

Configure SWF verification

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

```
<Application>
    ...
    <SWFVerification enabled="false">
        <SWFFolder></SWFFolder>
        <MinGoodVersion></MinGoodVersion>
        <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 `enabled` attribute to `"true"` or `"false"` to turn this feature on or off. The default value is `"false"`.
`SWFFolder`	None.	A single folder or a semicolon-delimited list of folders that contain copies of client SWF files for an application. These SWF files are used to verify connecting SWF files. The default value is the application's folder appended with /SWFs. For example, for an application called myApplication, if there isn't a value set for this element, verifying SWF files should be placed in the applications/myApplication/SWFs folder.
`MinGoodVersion`	None.	Specifies the minimum version of this feature to accept. The default value is `0`, which allows this and all future versions.
`UserAgentExceptions`	None.	Container.
`Exception`	`from` `to`	A user agent to except from verification. Use the `from` and `to` attributes to indicate the lowest and highest versions to except. This a string comparison, with editing to make all numeric fields equal in length. For more information, see the comments in the Application.xml file.
`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 Encoder—don't 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 Flash Media Encoder, to bypass SWF verification, as in the following:

```
<SWFVerification enabled="true">
    ...
    <UserAgentExceptions>
        <Exception to="FME/1.0" from="FME/1.0"/>
    </UserAgentExceptions>
</SWFVerification>
```

Verify administrative clients

In the `Root/Server/SWFVerification/SWFFolder` tag in the Server.xml file, you can specify folders that hold SWF files to verify SWF files trying to connect to any application or instance on the server. You can also specify the cache values for these SWF files.

These directories are intended for administrative purposes; SWF files placed in these directories can be verified to view any application instance on the server. For example, if you created a recorded file viewer application that lets you view files from any application on the server, you could place a verifying SWF file in directory specified in the `SWFFolder` tag.

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, 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.*

Limit access to Flash Media Administration Server

By default, a client can connect to Flash 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 Flash 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.

Disable RTMPE

By default, RTMPE is enabled in the server's Adaptor.xml file. In some scenarios, 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 Flash Media Server and you don't want them to use RTMPE, you might want to disable RTMPE at the server level.

To request an encrypted or encrypted tunneling channel, applications specify `rtmpe` or `rtmpte`, 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).

1 Open the fms.ini file (located in *RootInstall*/conf).

2 Set the `ADAPTOR.RTMPE_ENABLED` parameter to `"off"`.

3 Restart the server.

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

See also

XML configuration files reference

Configure SSL

Secure Sockets Layer (SSL) is a protocol for enabling secure communications over TCP/IP. Flash 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.

When SSL is configured, to use SSL, applications must specify the RTMPS protocol in the `NetConnection.connect()` URL; for example, `nc.connect("rtmps://www.example.com/myMediaApplication")`.

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.

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 passphrase must be specified in the `SSLPassPhrase` tag.
`SSLPassPhrase`	The passphrase 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 in the `SSLCipherSuite` entry in XML configuration files reference.
`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/fms.ini file, as follows:

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

This tells the server to listen on ports 1935 and 443, and that 443 is a secure port that receives only RTMPS connections. So a RTMPS connection to port 1935 will fail: the client attempts to perform a SSL handshake that the server fails to complete. A RTMPS connection to port 443 will succeed: the client performs a SSL handshake that the server completes. Similarly, a RTMP connection to port 443 will also fail: 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>
    ...
    <Ho4stPortList>
        <HostPort name="edge1" ctl_channel=":19350">${ADAPTOR.HOSTPORT}</HostPort>
    </HostPortList>
```

```
    ...
</Adaptor>
```

2 Create new `HostPort` elements with unique `name` and `ctl_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" ctl_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 pseudorandom 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 (`true`) or not (`false`). The default value is `true`. 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 files must be named by the subject name's 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 *RootInstall*\certs directory. You can import the Windows certificate store to the certs directory by running `FMSMaster -console -initialize` 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 in the `SSLCipherSuite` entry in XML configuration files reference.

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>
                <SSLCACertifcateFile></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.

Performing general configuration tasks

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/fms.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 Users.xml file.

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 of the virtual host or application you want to configure.

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.

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

1. 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.

2. **Allow the server to listen on IPv6 sockets.**

   Open the *RootInstall*/conf/fms.ini file and set the `SERVER.NetworkingIPV6 enable` attribute to `"true"`. Restart the server.

3. **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.

4. **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:

   ```
   FMS detected IPv6 protocol stack!
   FMS config <NetworkingIPv6 enable=true>
   FMS running in IPv6 protocol stack mode!
   Host: fmsqewin2k3-02 IPv4: 10.133.192.42 IPv6: fe80::204:23ff:fe14:da1c%4
   Listener started ( _defaultRoot__? ) : 19350/v6
   Listener started ( _defaultRoot__? ) : 19350/v4
   Listener started ( _defaultRoot__? ) : 1935/v6
   Listener started ( _defaultRoot__? ) : 1935/v4
   ```

Note: *On IPv6-enabled Linux, if you are using an IPv4 host name (a host name that resolves to IPv4) on an RTMPT or RTMPTE connection, you should configure the Adaptor.xml appropriately to resolve connections quickly. See the HTTPIdent2 tag in Adaptor.xml.*

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 `JSEngine` 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>
    <JSEngine>
            <config>
                <user_name>jdoe</user_name>
                <dept_name>engineering</dept_name>
            </config>
    </JSEngine>
</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>
    <JSEngine>
            <config>
                <compName>${%COMPUTERNAME%}</compName>
                <dept>${DEPT}</dept>
            </config>
    </JSEngine>
</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: *In server-side code,* `trace()` *statements are displayed in the Live Log panel of the Administration Console.*

See also

Using symbols in configuration files

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 Locate the following code in the Application.xml file:

```
...
    ...
        <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.*

2 Edit the following elements.

Element	Description
`BandwidthDetection`	Set the `enabled` 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 `MaxWait` 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.

3 Save and validate the Application.xml file.

4 Restart the server.

Configuring content storage

About content storage

To improve server performance, configure storage correctly. The server can use local or network storage to serve media files. If desired, you can change the default location where streams and shared objects are stored and also map virtual directories to physical directories on local or network storage to organize content.

Note: *When media files are used in an application, they are cached in local RAM.*

Setting the location of application files

The applications folder registers applications with the server; that is, the presence of an application within the applications folder tells the server that the application exists. It is by default located at *RootInstall*/applications.

Within the applications folder, you create subfolders for your applications. Within each individual application folder, you create subfolders to create instances of applications, for example:

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 fms.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 Flash Media Server and Flash Media Administration Server.

2 Make the changes to the config.

3 Check that the the server user has appropriate access rights to map to the network drive (system account rights are usually not sufficient.)

4 Restart Flash Media Server and Flash 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 sharedobjects folder in the application directory.

Using the `<storageDir>` element in the Application.xml file, you can 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.

Note: *If you use this tag to map to a network drive, see Mapping directories to network drives for additional information.*

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. Flash 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. See Mapping virtual directories to physical directories.

Mapping virtual directories to physical directories

Flash 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 Flash Media Server's application directory, and you can retain your existing classification and categorization of resources.

Note: *If you use this tag to map to a network drive, see Mapping directories to network drives for additional information.*

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.
- 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 the the Vhost.xml file instead of the virtual-host Application.xml file.

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)

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 make them instantly streamable. 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. A connecting client will be able to play a file in the virtual directory, such as **C:\my_videos\sample.flv**, by connecting to the vod application and issuing a NetStream `play()` call:

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

or by passing `"rtmp://myDomain/VOD/sample.flv"` to the `source` property of a call to `FLVPlayback.play()`.

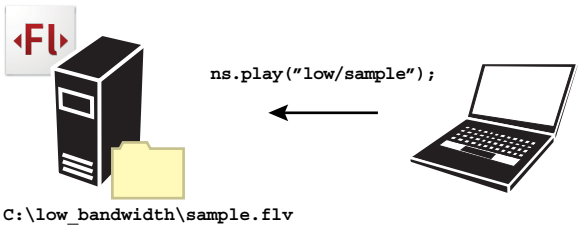
The `<VirtualDirectory>` element in the application-specific Application.xml file affects only that particular application, protecting your streams from being accessed by other applications on the same virtual host. It has a higher precedence than the virtual directory mapping in the Vhost.xml file, so it will always be checked first.

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 a specific directory, C:\low_bandwidth, and all streams that start with *high* are stored in a different directory:

```
<VirtualDirectory>
    <Streams>low;c:\low_bandwidth</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 folder.



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.

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 <VirtualDirectory><Streams> tag	**URI in NetStream call**	**Location of published stream**
low;e:\fmsstreams	"myStream"	c:\...*RootInstall*\applications\yourApp\streams_definst_\myStream.flv
low;e:\fmsstreams	"low/myStream"	e:\fmsstreams\myStream.flv
low;\\mynetworkDrive\share\fmsstreams	"low/myStream"	\\mynetworkDrive\share\fmsstreams\myStream.flv

Chapter 4: Using the Administration Console

Use the Adobe Flash 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 (fms_adminConsole.swf) that lets you manage the server and view information about applications running on the server.

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

By default, the Administration Server in installed on port 1111 (the default value at installation time). You can change the port number of the Administration Server after installation by editing the fms.ini file.

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

Connect to the Administration Console

When you log in to the Administration Console as a virtual host administrator (not a server administrator), your session is specific to a virtual host, and you can only manage applications running on that virtual host. To manage applications running on a different virtual host, you need to log in to that virtual host. You cannot access applications running on different virtual hosts in the same login session.

1. Do one of the following:

   - On Windows, select Start > Programs > Adobe > Flash Media Server 3 > Flash Media Administration Console.
   - On Linux, open the fms_adminConsole.htm and fms_adminConsole.swf files in a browser with Flash Player.
   - On Mac, copy the fms_adminConsole.htm and fms_adminConsole.swf file to the Mac. Open the fms_adminConsole.htm file in a web browser that has Flash Player installed.

   The fms_adminConsole.swf and fms_adminConsole.htm files are located in the root installation folder.

2. Enter the name and address of the server or virtual host to which you want to connect.

   - If you want, specify a server name; the server name is simply 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.
   - 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**. This 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. The host name must be mapped to a valid network IP address.
- If you are connecting remotely by running the Administration Console on another computer, enter the server's name (FlashMediaServer.myCompany.com) or the IP address and port number (12.34.56.78:1112) 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.

3 Enter the administrator user name and password you entered during the Flash Media Server installation. If you changed the administrator user name and password using the Administration Console or manually in the Users.xml file, enter the new user name and password.

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 to a virtual host on the adaptor `_secondAdaptor_`, the administrator `JLee` would enter the following information in the Name box: `_secondAdaptor_/JLee`.

4 (Optional) Select the Remember My Password option.

5 (Optional) Select the Automatically Connect Me option.

6 (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.)

7 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 Flash Media Server website and related resources. Click the question mark icon to display links to Flash Media Server Help.

To run the Administration Console from a computer other than the one in which the server is installed, copy fms_adminConsole.htm and fms_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 is refreshed 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 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.

Access Help

Access Flash Media Server LiveDocs

❖ From the Administration Console, select Help/Documentation.

Access locally installed help

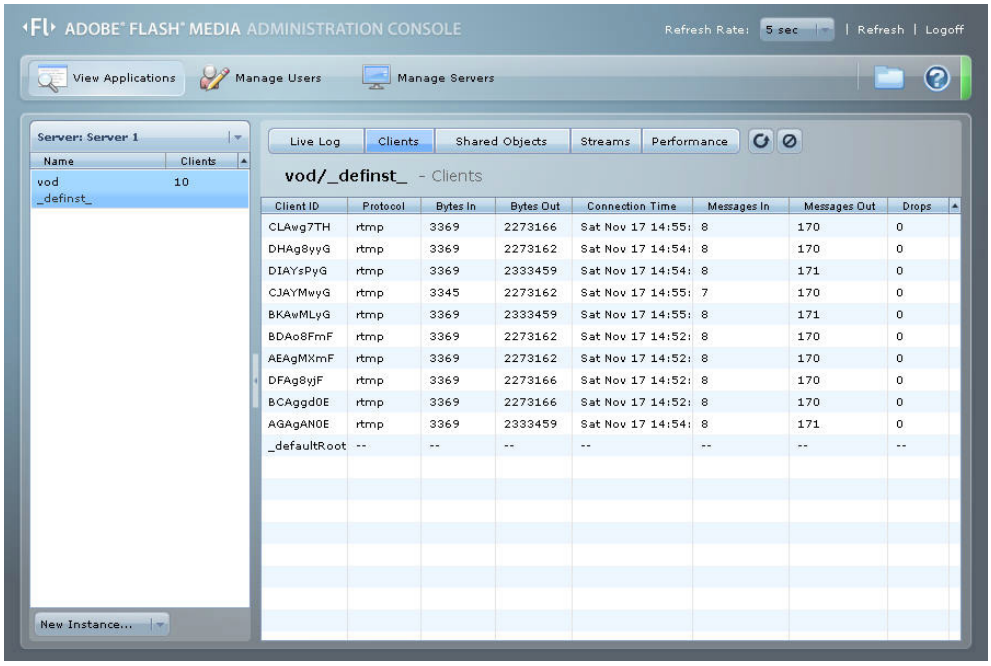
❖ On Windows, select Start > Programs > Adobe > Flash Media Server > Documentation.

❖ On Linux or Mac OS, open the documentation folder in the installation directory.

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, and 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 36.

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 `suffix _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 36.

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 36.

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 application list

1 Follow the steps in "Connect to the Administration Console" on page 36.

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 36.

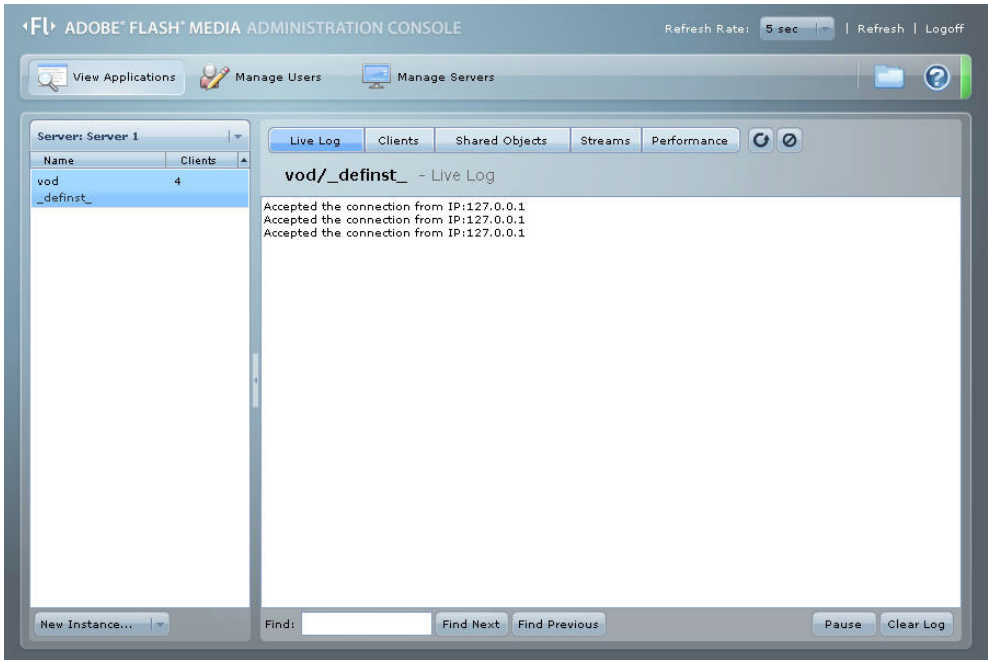
2 Click View Applications.

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 36.

2 Click View Applications.

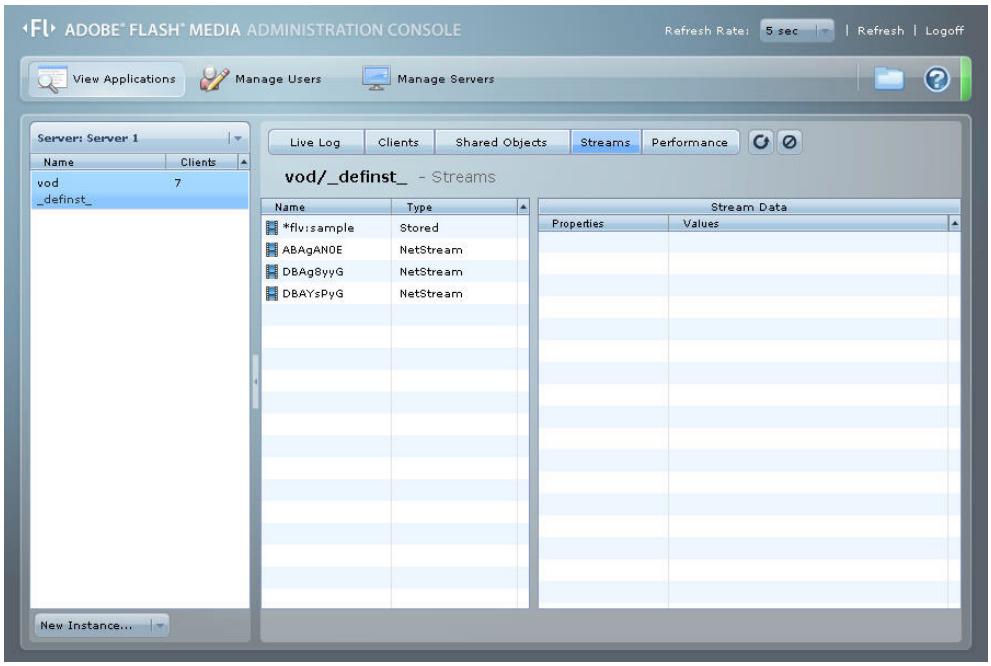
3 Click Live Log.

4 Select an application from the list.

5 Type string 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 36.

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:

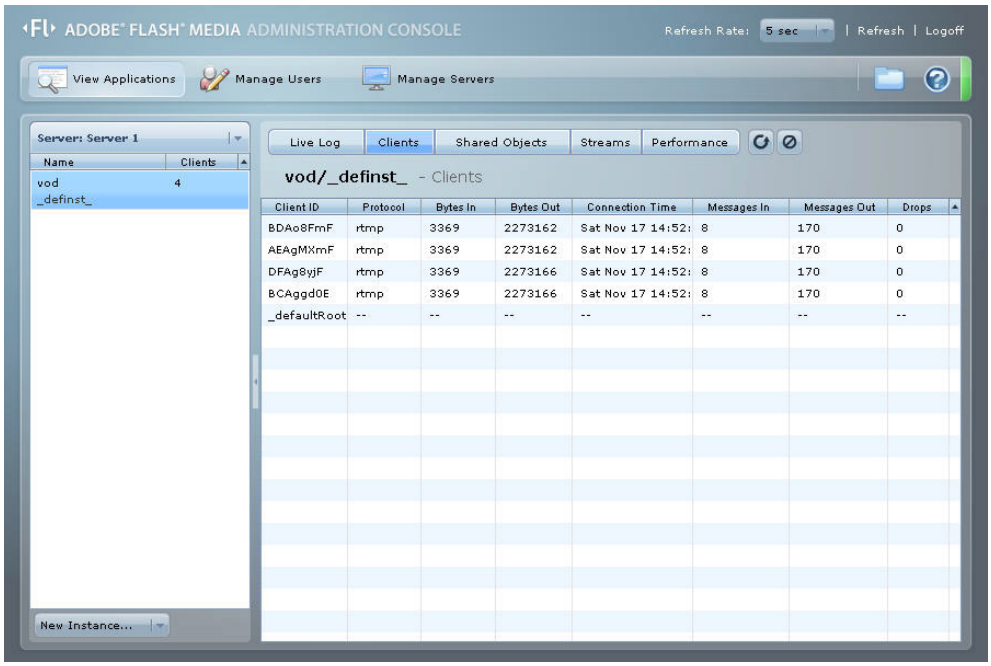
- Name: The actual stream name, not streamID.
- Status: States if the stream is publishing, playing live, or playing recorded.
- Client: The client ID playing the stream.
- Time: The time that the client started playing the stream.

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.



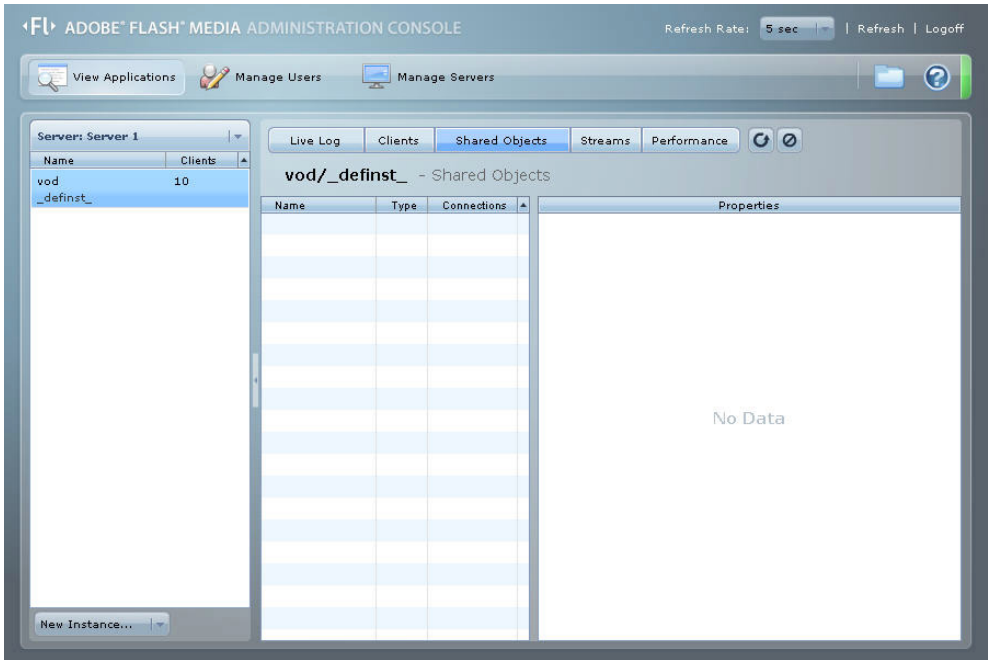
Clients panel

1. Follow the steps in "Connect to the Administration Console" on page 36.
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 Flash 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 (this duration is configurable), 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

1 Follow the steps in "Connect to the Administration Console" on page 36.

2 Click View Applications.

3 Click Shared Objects.

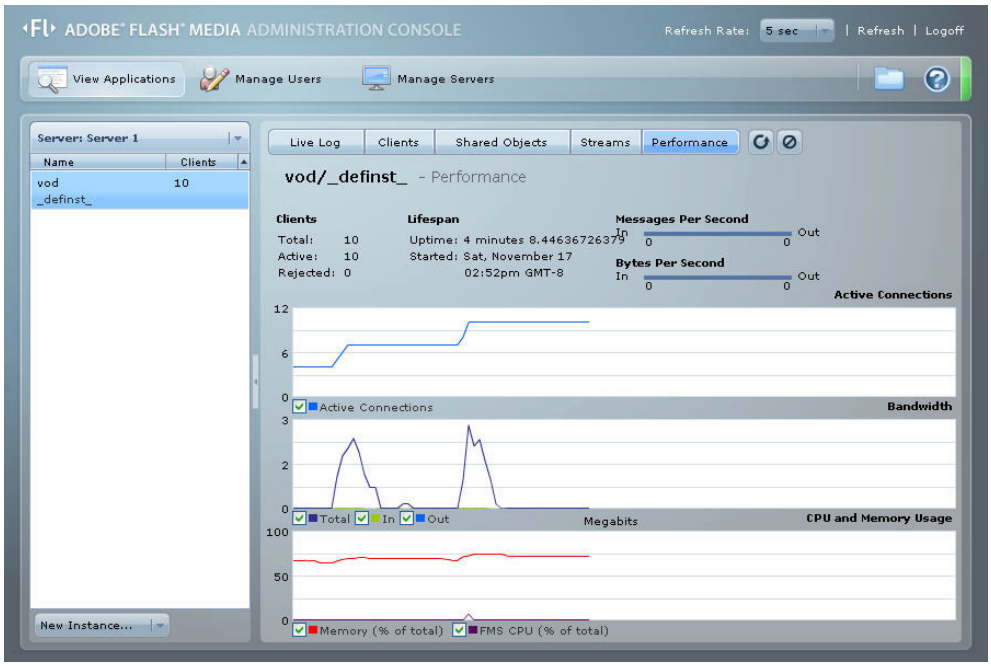
4 Select an application from the list.

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.

Note: *If the shared object is available for debugging the application, the Administration Console displays its properties. If the shared object is not available for debugging, an error message is displayed.*

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 (this duration is configurable), or click Refresh to refresh at any time.



Performance panel

1 Follow the steps in "Connect to the Administration Console" on page 36.

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 Flash 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.

Managing 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

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.

5 Restart Flash Media Administration Server.

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 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="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.

5 Restart Flash Media Administration Server.

Managing server administrators

You must be a server administrator (not a virtual host administrator) to perform operations on the Manage Users tab.

Add a server administrator

1 Follow the steps in "Connect to the Administration Console" on page 36.

2 Click Manage Users.

3 Click New User.

4 Type a user name and password.

5 Click Save or Save And Add Another.

Reset user password

1 Follow the steps in "Connect to the Administration Console" on page 36.

2 Click Manage Users.

3 Click Reset The Password For This User.

4 Type a new password.

Delete user account

1 Follow the steps in "Connect to the Administration Console" on page 36.

2 Click Manage Users.

3 Click Delete This User Account On The Server.

4 Confirm the action.

Managing the server

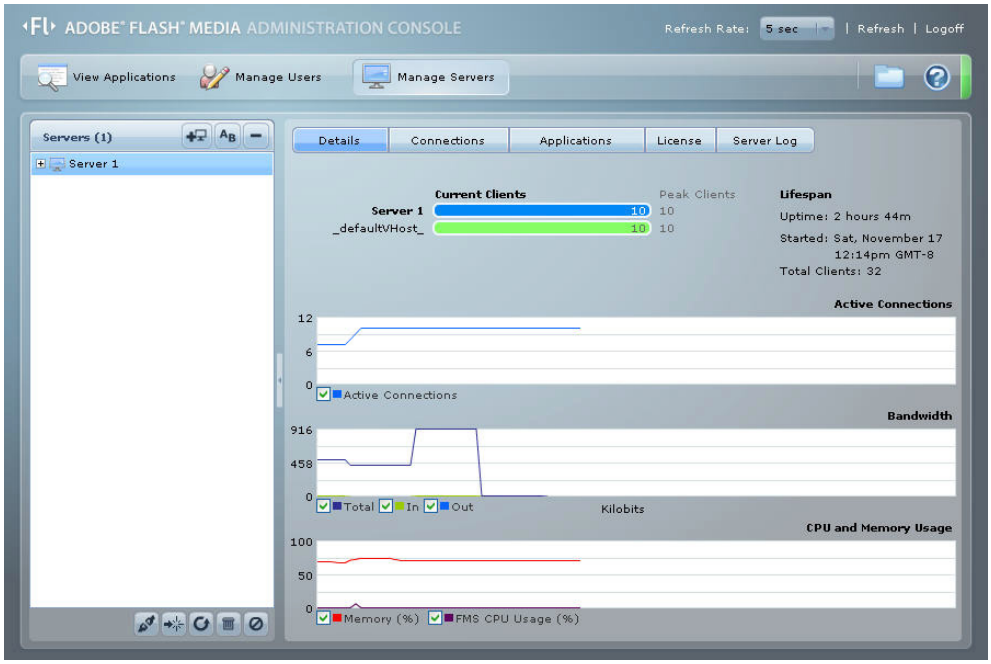
Monitoring server performance

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 36.

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 36.

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 36.

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 and 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 Flash 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 36.
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 36.
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.

Note: *Serial numbers that are added manually (that is, added by editing configuration files directly) to either fms.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 36.

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 5: Monitoring and Managing Log Files

Adobe Flash 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.

Working with log files

Managing log files

Flash Media Server maintains several different types of logs. The server outputs statistics about client connections and stream activity to access logs. Flash Media Server also maintains diagnostic logs and application logs for application activities. (The application and diagnostic logs are an addition to operating system logs that track error and informational messages about Flash Media Server operations.)

access.log Tracks information about users accessing the server.

application.log Tracks information about activities in application instances.

diagnostic logs Tracks information about server operations.

Note: *In Adobe Flash Player 9 Update 3, Flash Player no longer notifies the server about pause events.*

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 a sample, see the Logger.xml file installed with Flash Media Server in the /conf directory.

Note: *Log file rotation cannot be disabled. To effectively turn off rotation, however, you can choose a large maximum size and a long maximum duration.*

You can write an operating system script to delete or back up the log regularly. For important log files, you must move the log directory to a backup location. The current active file can be moved; the server creates a new file on the next log event.

Verifying IPv6 in log files

IPv6 (Internet Protocol version 6) is a new version of the Internet Protocol that supports 128-bit addresses. To use IPv6, you need to activate IPv6 on the network interface card, enable Flash Media Server to listen on IPv6 sockets, and enclose numeric IPv6 addresses in URLs within brackets.

After following those steps, Flash Media Server (when it starts) 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. 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 Flash Media Server host has dual addresses and is listening on both interfaces;

```
FMS detected IPv6 protocol stack!
FMS config <NetworkingIPv6 enable=true>
FMS running in IPv6 protocol stack mode!
Host: fmsqewin2k3-02 IPv4: 10.133.192.42 IPv6: fe80::204:23ff:fe14:da1c%4
Listener started ( _defaultRoot__? ) : 19350/v6
Listener started ( _defaultRoot__? ) : 19350/v4
Listener started ( _defaultRoot__? ) : 1935/v6
Listener started ( _defaultRoot__? ) : 1935/v4
```

Note: *In 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 example above, in Linux, only the two /v6 entries would be displayed.*

For more information about using IPv6, see Configuring IPv6.

Access logs

Reading access logs

The access log records information about requests by Flash Player and Flash 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 Flash Media Server logs directory. The default configuration for Flash Media Server creates a single access log per server. You can also configure Flash Media 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.)

Note: *The access logs are in W3C format. Administrators can use standard parsing tool to parse the log files.*

Flash 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.

Access events defined in access logs

Event	Category	Description
connect-pending	application	Client connects to the server, waiting for the client to be authenticated.
connect	application	Client connects to the server.
disconnect	application	Client disconnects.
publish	application	Client publishes a live stream.
unpublish	application	Client unpublishes a live stream.
play	application	Client plays a stream.
pause	application	Client pauses stream.
unpause	application	Client resumes playing stream.
seek	application	Client jumps to a new location within a recorded stream.
stop	application	Client stops playing or publishing a stream.
record	application	Client begins the recording of a stream.
recordstop	application	Client stops the recording of a stream.
server-start	application	Server has started.
server-stop	application	Server has stopped.
vhost-start	application	A virtual host has started.
vhost-stop	application	A virtual host has stopped

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	Category	Description
x-event	application	Type of event.
x-category	application	Event category.
date	application	Date of the event.
time	application	Time the event occurred.
tz	application	Time zone information.
x-ctx	application	Event-dependent context information.
x-pid	application	Server process ID.

Field	Category	Description
x-cpu-load	application	CPU load.
x-mem-load	application	Memory usage (as reported by the `getServerStats()` method).
x-adaptor	application	Adaptor name.
x-vhost	application	Virtual host name.
x-app	application	Application names.
x-appinst	application	Application instance names.
c-ip	application	Client IP address.
c-proto	application	Connection protocol: RTMP or RTMPT.
s-uri	application	URI of the Flash Media Server application.
c-referrer	application	URI of the referrer.
c-user-agent	application	User agent.
c-client-id	application	Client ID.
cs-bytes	application	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	application	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-service-name		Name of the service providing the connection (only applicable to certain connection types).
x-sc-qos-bytes		Number of bytes sent to client for quality of service.
x-comment		Comments.
x-sname	application	Stream name.
x-file-size	application	Stream size, in bytes.
x-file-length	application	Stream length, in seconds.
x-spos	application	Stream position.
cs-stream-bytes	application	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.

Field	Category	Description
sc-stream-bytes	application	This field shows the number of bytes transferred from the server to the client per stream. To calculate the bandwidth usage per stream, subtract sc-stream-bytes in the "play" event from sc-stream-bytes in the "stop" event. Note: sc-stream-bytes can be greater than x-file-size after streaming files not encoded in FLV format, such as MP3 files.
cs-uri-stem	application	Stem portion of s-uri (omitting query) field.
cs-uri-query	application	Query portion alone of s-uri.
x-sname-query	application	Query portion of stream URI specified in play or publish.
x-file-name	application	Full path of the file representing the x-sname stream.
x-file-ext	application	Stream type (currently, this can be FLV or MP3).
s-ip	application	IP address or addresses of the server.
x-duration	application	Duration of a stream or session event.
x-suri-query	application	Same as x-sname-query.
x-suri-stem	application	This is a composite field: cs-uri-stem + x-sname + x-file-ext.
x-suri	application	This is a composite field: cs-uri-stem + x-sname + x-file-ext + x-sname-query.
x-status	application	For a complete description of the x-status codes and descriptions, see Fields in diagnostic logs.

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 shut-down (or application unloaded).
connect, publish, unpublish, play, record, record stop, stop	status_OK	200	Successful.
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.

Field	Symbol	Status Code	Description
play	client_disconnect	408	Stream stopped because client disconnected.
connect, publish	status_conflict	409	Resource limit exceeded. (In authorization, a change has been made by the Authorization plug-in.) Or, Stream is already being published.
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.
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	Flash 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, play	server_internal_error	500	Server internal error.

Field	Symbol	Status Code	Description
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.
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 Flash Media Server logs directory. Flash 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.

Field	Event(s)	Description
x-pid	All	Server process ID.
x-status	All	Status code: The code is a 10-character string that represents the severity, category, and message ID. The first three characters represent severity, as follows: (w) = warning (e) = error (i) = information (d) = debug (s) = trace from server-side script (_) = unknown The next three characters represent category. All categories are listed in Status categories in diagnostic logs. The last four characters represent message ID. All message IDs are listed in Diagnostic Log Messages.
x-ctx	All	Event-dependent context information.

Diagnostic logs

Diagnostic log file

The diagnostic log records information about Flash 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 Flash 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 Flash Media Server logs directory. Flash 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 Log Messages.

Fields in diagnostic logs

Field	Event(s)	Description
date	All	Date on which the event occurred.
time	All	Time at which event occurred.

Field	Event(s)	Description
x-pid	All	Server process ID.
x-status	All	Status code: The code is a 10-character string that represents the severity, category, and message ID. The first three characters represent severity, as follows: (w) = warning (e) = error (i) = information (d) = debug (s) = trace from server-side script (_) = unknown The next three characters represent category. All categories are listed in Status categories in diagnostic logs. The last four characters represent message ID. All message IDs are listed in Diagnostic Log Messages.
x-stx	All	Event-dependent context information.

Status categories in diagnostic logs

Category	Description
257	TCService
258	TCServer
259	Presence
260	Storage
261	Stream
262	SMTP
263	Adaptor
264	JavaScript
265	TCApplication
266	TCConnector
267	Admin
268	SharedObject
269	Configuration
270	VirtualHost
271	SSL

Configuration files for logging

Flash Media Server logging is configured through the Server.xml and Logger.xml configuration files. Server.xml contains a `Logging` section that controls the overall logging behavior. This section includes an `Enable` tag that determines whether logging takes place, and a `Scope` tag that determines whether Flash Media 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 file(s).

Logger.xml files may be provided at the configuration root folder right next to Server.xml, and optionally for each virtual host right next to VHost.xml. The root Logger.xml file determines the logger configuration when the logging scope is server-wide. Optionally, a specific virtual host Logger.xml controls the logging behavior for a given virtual host. (If the scope is server-wide, virtual host Logger.xml files are not applicable.) The virtual host-specific Logger.xml configuration file is relevant only when the activities for each virtual host are being logged in a separate log file.

Note: *The root Logger.xml controls the logging behavior when the scope is set to vhost if the optional virtual host Logger.xml does not exist.*

For more information, see the Server.xml and Logger.xml files installed with Flash Media Server in the *RootInstall*/conf directory.

Note: *Log file rotation cannot be disabled. To effectively turn off rotation, however, you can choose a large maximum size and a long maximum duration.*

Chapter 6: Administering the server

Perform regular administrative tasks to keep the server running smoothly.

Start and stop the server

Start and stop the server in 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 > Flash Media Server 3 > Start Flash Media Server 3
- Choose Start > All Programs > Adobe > Flash Media Server 3 > Start Flash Media Administration Server 3

Stop the server from the Start menu

Do one of the following:

- Choose Start > All Programs > Adobe > Flash Media Server 3 > Stop Flash Media Administration Server 3
- Choose Start > All Programs > Adobe > Flash Media Server 3 > Stop Flash Media Server 3

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 Flash Media Server (FMS) from the Services list and click Stop, Start, or Restart.
   - Select Flash Media Administration Server from the Services list and click Stop, Start, or Restart.

Start and stop the server in Linux

Start, stop, or restart Flash Media Server

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:
   - `./fmsmgr server fmsstart`
   - `./fmsmgr server fmsstop`
   - `./fmsmgr server fmsrestart`

Start, stop, or restart the Administration Server

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:

- `./fmsmgr adminserver start`
- `./fmsmgr adminserver stop`
- `./fmsmgr adminserver restart`

Start, stop, or restart Flash Media Server using the command line

1 * cd /<the directory where FMS is installed>.

2 Enter `./server [start | stop | restart]`.

Start, stop, or restart the Administration Server using the command line

1 * cd /<the directory where FMS is installed>.

2 Enter `./adminserver [start | stop | restart]`.

Checking server status

View server events in the Windows Event Viewer

The Windows Event Viewer can be used for tracking Flash 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 Flash Media Server to view details.

Check server health

FMSCheck 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, FMSCheck is completely scriptable using the language of your choice, such as Cscript, bash, C shell, or Python. FMSCheck provides information about whether the server is running or not, what the response time is, and which fmscore 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 Flash 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, FMSCheck only supports RTMP connections and does not check for shared objects.*

FMSCheck 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 the application. Administrator must specify the application. Example: `--app app1/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 FMSCheck.
`--logfile <file>`	Allows the program to output 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 at 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.
`fmscheck -v`	Prints a version string.

Option	Description
`--auser <username>`	Specifies a user name for the Administration Server user. Example: `--auser admin`
`--apswd <password>`	Specifies a password for the Administration Server. Example: `--apswd admin`
`--ahostport <port>`	Specifies the Administration Server port number. If the port number is not specified in the command line, the default port is 1111. Example: `--ahostport 1111`

Usage examples for Windows:

- `* fmscheck.exe --host localhost --app app1 --logfile output.txt`
- `* fmscheck.exe --host localhost --app app1 --play foo 0 10 --logfile output.txt`
- `* fmscheck.exe --host localhost --app app1 --pubfile foo.flv --publish bar 10 --logfile output.txt`
- `* fmscheck.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 `* ./fmscheck` instead of `* fmscheck.exe`.

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 Flash 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 Flash 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

Flash Media Server supports playback 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. Any file with the Adobe extension .f4v is part of the supported subset and can be delivered using Flash Media Server.

For MPEG-4–based container formats with extensions other than .f4v, 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 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 --file <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 "\" character; try the "/" character instead.)
`-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 `metadata_duration` and `actual_duration` 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 `-d 0`.
`-q [ --quiet ]`	Specifies that only the status code, not the text output, be returned. The `--help` option overrides this option.
`-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 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); errors and warnings are described in the sections FLVCheck errors and FLVCheck warnings.

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 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. Flash 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.
-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. For a list of supported codecs, see Streaming media.
-15	MP4	Too many tracks. Max is 64.
-16	MP4	Only one sample type allowed per track.
-17	MP4	Box is too large.
-18	MP4	Truncated box.
-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.

FLVCheck warnings

Generally, warnings are informative and are not fatal errors; Flash 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 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. Flash Media Server ignores (will not play back) tracks that are encoded with unsupported codecs. For a list of supported codecs, see Streaming media.
-108	MP4	Invalid video codec. This warning indicates that a track has an invalid video codec. Flash Media Server cannot play back the track. For a list of supported codecs, see Streaming media.
-109	MP4	Invalid audio codec. This warning indicates that a track has an invalid audio codec. Flash Media Server cannot play back the track. For a list of supported codecs, see Streaming media.
-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.

Clearing the edge server cache

Deleting files from the edge server cache

Edge servers do not delete content automatically; you must delete unused files to manage disk usage. Edge servers update the timestamp of a file each time the file is used. You can use the timestamp to determine when a cached file was used.

Manage the edge server cache in Windows

You can create a weekly scheduled task to clear the edge server cache.

1. Create a cache.bat file to empty the cache directory. The entry in the cache.bat file has the following syntax:

   ```
   del /Q /S <cache_directory>\*.*
   ```

2. Run the cache.bat file and verify that it deletes files in the cache directory.

The directory structure remains and any files currently locked by the edge server are not deleted; this is expected behavior.

3 Select Control Panel > Scheduled Tasks > Add Scheduled Task.

4 Select cache.bat as the new file to run.

5 Replicate this procedure on each edge server.

Manage the edge server cache in Linux

1 Create a shell script named cache.sh to empty the cache directory. The cache.sh script has the following syntax:

```
find <cache_directory> -name "*.flv" -exec rm-f{}\;
find <cache_directory> -name "*.mp3" -exec rm-f{}\;
find <cache_directory> -name "*.mp4" -exec rm-f{}\;
```

Note: *You can add any additional file types to the script.*

2 Ensure the script is executable by running the following command:

```
$ chmod 700 cache.sh
```

3 Run cache.sh to verify that it deletes the correct files in the cache directory.

4 Use the cron utility to schedule cache.sh to run. (For details about the cron utility, see the documentation for your Linux distribution.)

Managing the server on Linux

Use the `fmsmgr` utility to perform basic management tasks for the Flash Media Administration Server running on Linux systems, such as starting and stopping the server and services. You must be a root user to use the `fmsmgr` utility.

For tasks not listed in the following table, such as adding users or checking the status of applications, use the Administration Console. Although you do not need to be a root user to use the Administration Console, the Administration service itself does need to be started by a root user using the `fmsmgr` utility before anyone can use the Administration Console.

Note: *Running multiple Flash Media Server services concurrently is not supported.*

Syntax

```
fmsmgr server <service_name> <cmd>
```

Command	Description
`fmsmgr adminserver start|stop|restart`	Starts, stops, or restarts the Flash Media Administration Server.
`fmsmgr clearautostart`	Sets the Flash Media Administration service to start manually. The *`service_name`* is the name of the server you selected during installation. If no name is specified, the action is performed on the default server. If the default *`service_name`* does not exist, the command fails.
`fmsmgr list`	Lists all the services installed, including Administration services, with additional information about services that are currently running.

Command	Description
`fmsmgr remove`	Removes the Flash Media Server service from the fmsmgr 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.
`fmsmgr add service_name install_dir`	Add a Flash Media Server service to the fmsmgr tables. *`service_name`* is the name of the server you select. If *`service_name`* already appears in the fmsmgr tables, the old entry is updated with the new. *`install_dir`* is the absolute directory path where you installed Flash Media Server.
`fmsmgr server service_name restart`	Stops a running Flash Media Server service and restarts it. If no name is specified, the action is performed on the default server. If the default *`service_name`* doesn't exist, the command fails.
`fmsmgr server service_name stop`	Stops the specified Flash Media Server service. *`service_name`* is the name of the server you selected during installation. If no name is specified, the action is performed on the default server. If the default *`service_name`* doesn't exist, the command fails.
`fmsmgr server service_name start`	Starts the Flash Media Server service. *`service_name`* is the name of the server you selected during installation.
`fmsmgr setadmin service_name`	Changes the default Administration service. *`service_name`* is the name of the server you selected during installation. The Administration service name is the same as the Flash 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.
`fmsmgr getadmin`	Gets the name of the default Administration service.
`fmsmgr setautostart service_name`	Sets the Flash Media Server service to start automatically when the system is started. *`service_name`* is the name of the server you selected during installation. If no name is specified, the action is performed on the default server. If the default *`service_name`* doesn't exist, the command fails.
`fmsmgr clearautostart service_name`	Sets the Flash Media Administration service to start manually. The *`service_name`* is the name of the server you selected during installation. If no name is specified, the action is performed on the default server. If the default *`service_name`* does not exist, the command fails.
`fmsmgr suggestname`	Suggests a service name that does not already appear in the fmsmgr tables.

Chapter 7: 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 Flash Media Server Administration Console was built using the Administration API. The API is described in detail in *Adobe Flash Media Server Administration API Reference*.

Here are a few important tips for working with the Administration API:

- Both Adobe Flash Media Server and Flash 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 Flash 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 fms.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.*

5. Save the file and restart Flash Media Administration Server.

Call an Administration API method over HTTP

1 Verify that Flash 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 Flash Media Server, use the server address instead of `localhost`. Substitute your administrator user name and password, which are located in the fms.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 `unload App()` to work properly.*

- Arrays are passed as comma-separated values enclosed by 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 the Flash 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.

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 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: *Since this is a Flash Media Server application, you must create an application directory with the application name in the* RootInstall*applications directory. <verify then add this back -sr>*

2 Enter the following code on frame 1 of a Flash file:

```
/** Establishes the connection to Flash 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 Flash 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 Flash Media Server application you started in step 3 in the input text field.

Method summary

Methods for monitoring the server

Queries let you monitor Flash Media Server, its applications, and specific instances of its applications. The following table lists the methods you can use to monitor the server.

Method	Description
`approveDebugConnection()`	Approves a pending debug session's request to connect to a selected application.
`getActiveInstances()`	Returns an array of strings that contains the names of all running application instances on the connected virtual host.
`getActiveVHosts()`	Returns an array of active virtual hosts.

Method	Description
`getActiveVHostStats()`	Returns performance statistics for the active virtual hosts.
`getAdaptors()`	Returns an array of adaptor names.
`getAdminContext()`	Returns the administrative context for the administrator (administrator type, name of adaptor, and name of the virtual host).
`getAdmins()`	Returns all the administrators on the Flash Media Server.
`getApps()`	Returns an array of strings that contains the names of all the applications that are installed.
`getAppStats()`	Returns aggregate information of all instances for an application.
`getFileCacheStats()`	Returns data about the file cache.
`getGroupMembers()`	Returns a list of the group members for a particular group.
`getGroupStats()`	Returns statistics for a particular group connection.
`getGroups()`	Returns a list of the group connections for a particular application instance.
`getInstanceStats()`	Returns detailed information about a single running instance of an application.
`getIOStats()`	Returns the I/O information: bytes in, bytes out, and so on.[a]
`getLicenseInfo()`	Returns license key information.
`getLiveStreams()`	Returns a list of all live streams currently publishing to a particular application.
`getLiveStreamStats()`	Returns detailed information about a live stream.
`getMsgCacheStats()`	Returns server TCMessage cache statistics.
`getNetStreams()`	Returns a list of all network streams that are currently connected to the application.
`getNetStreamStats()`	Returns detailed information about a specific network stream.
`getRecordedStreams()`	Returns an array containing the name of all the recorded streams currently playing from a particular instance of an application.
`getRecordedStreamStats()`	Returns detailed information about a recorded stream.
`getScriptStats()`	Gets the performance data for a script running on the specified instance of an application.
`getServerStats()`	Retrieves the server status and statistics about the operation of the server, including the length of time the server has been running and I/O and message cache statistics.[a]
`getServices()`	Returns an array containing the names of all the services currently connected to Flash Media Server.
`getSharedObjects()`	Returns a list of all persistent and nonpersistent shared objects that are currently in use by the specified instance of an application.
`getSharedObjectStats()`	Returns detailed information about a shared object.
`getUsers()`	Returns a list of all users who are currently connected to the specified instance of an application.
`getUserStats()`	Returns detailed information about a specified user.
`getVHosts()`	Returns an array of virtual hosts defined for the specified adaptor.
`getVHostStats()`	Returns statistics for a virtual host.
`ping()`	Returns a status string indicating the condition of the server.

a. Only server administrators can use this method. Virtual host administrators cannot use these methods. In some cases, virtual administrators can use a method with restrictions; these restrictions are described in the dictionary entry for the method.

For more information, see *Adobe Flash Media Server Administration API Reference*.

Administrative methods

Administrative methods let you add administrative users, and start and stop the server, virtual hosts, and applications. The following table lists the methods you can use to administer the server.

Method	Brief description
`addAdmin()`	Adds an administrator to the system.[a]
`addApp()`	Adds a new application.
`addVHostAlias()`	Adds an alias to a virtual host.
`changePswd()`	Changes the password for an administrator in the system.
`gc()`	Forces garbage collection of server resources.[a]
`reloadApp()`	Unloads an instance of an application if it is loaded, and then reloads the instance.
`removeAdmin()`	Removes an administrator from the system.[a]
`removeApp()`	Removes an application or an instance of an application.
`removeVHostAlias()`	Removes an alias from a virtual host.
`restartVHost()`	Restarts a virtual host.
`startServer()`	Starts or restarts Flash Media Server.[a]
`startVHost()`	Starts the specified virtual host if it stops. Enables a new virtual host if the virtual host directories have been created in the file system.[a]
`stopServer()`	Shuts down the Flash Media Server.[a]
`stopVHost()`	Stops an added virtual host (not _defaultVHost_).
`unloadApp()`	Unloads all instances of an application or one instance of an application. Disconnects all users.

a. Only server administrators can use this method. Virtual host administrators cannot use these methods. In some cases, virtual host administrators can use a method with restrictions; these restrictions are described in the dictionary entry for the method.

For more information, see *Adobe Flash Media Server Administration API Reference*.

Methods for configuring the server

Configuration methods let you view and set values for server configuration keys.

Method	Description
`getConfig2()`	Returns configuration information for the specified configuration key.
`setConfig2()`	Sets a value for a specified configuration key.

For more information about configuration keys, see the entries for `getConfig2()` and `setConfig2()` in *Adobe Flash Media Server Administration API Reference*.

Chapter 8: XML configuration files reference

Edit the configuration files in the *RootInstall*/conf directory to configure the server. For more information, see Editing configuration files.

Important: *Some XML elements should not be configured without contacting Adobe Support. These elements are marked with an* Important *note.*

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 Adaptor.xml file installed with Adobe Flash Media Server in the *RootInstall*/conf/_defaultRoot_ directory.

Summary of elements

Adaptor.xml element	Description
`Adaptor`	Root element; contains all the other adaptor configuration elements.
`Allow`	Identifies the specific hosts from which clients can connect to the server.
`Deny`	Identifies those hosts whose clients' attempts to connect to the server(s) will be rejected.
`Edge`	Identifies an edge to configure for HTTP tunneling.
`Enable`	Enables or disables tunneling connections into this application.
`EnableAltVhost`	Determines if an alternate virtual host may be specified as a part of the RTMP URL as query parameter.
`HostPort`	Specifies the IP address and port(s) to bind to.
`HostPortList`	Container element; comprised of a list of `HostPort` elements.
`HTTPIdent`	Configures the server to respond to or reject an HTTP identification request from a client.
`HTTPIdent2`	Configures the server to respond to or reject a special HTTP request from a client before attempting an RTMPT connection to Flash Media Server.
`HTTPNull`	Configures the server to respond to or reject an HTTP GET request for the "/" resource from a client.

Adaptor.xml element	Description
HTTPTunnel	Container element; the elements in this section configure the incoming HTTP tunneling connections.
HTTPUserInfo	Container element; the elements in this section configure the absolute path to XML files and the cache settings.
IdleAckInterval	Specifies the maximum time the server may wait before it returns an `ack` (acknowledgement code) for a client idle post.
IdlePostInterval	Specifies the interval at which the client should send idle posts to the server to indicate that the player has no data to send.
IdleTimeout	Specifies the maximum inactivity time, in seconds, for a tunnel session before it is closed.
MaxFailures	Specifies the maximum number of failures an edge server may incur before restarting.
MaxSize	Specifies maximum number of HTTP requests the server keeps in the cache.
MaxWriteDelay	Specifies how long the server waits for a write.
MimeType	Specifies the default MIME type header sent on tunnel responses.
NeedClose	Specifies whether HTTP 1.0 non-keepalive connections are to be closed once the response is written.
NodeID	Specifies a unique node identification to support the implementation of load balancers.
Order	Specifies the order in which to evaluate the `Allow` and `Deny` elements.
Path	Specifies the absolute path of the folder where the server looks for XML files supported by HTTPUserInfo.
RecoveryTime	Specifies the wait time for an edge server to pause after failing, before it restarts.
Redirect	Specifies whether or not the adaptor redirects unknown requests to an external server.
ResourceLimits	Container element; contains elements that configure the resources for an edge server.
RTMP	Container element; contains details and configurations for different versions of RTMP.
RTMPE	Specifies if enhanced (encrypted) RTMP can be used.
SetCookie	Specifies whether the adaptor sets a cookie.
SSL	Container element; contains elements to configure Flash Media Server as an SSL (Secure Sockets Layer) client for incoming SSL connections.
SSLCertificateFile	Specifies the name of the directory containing one or more CA certificates.
SSLCertificateKeyFile	Specifies the name of a file that contains one or more CA certificates in the PEM encryption format.
SSLCipherSuite	Specifies the encryption ciphers that Flash Media Server uses to secure incoming connections.
SSLPassPhrase	Specifies the passphrase to use for decrypting the private key file. If the private key file is not encrypted, leave this tag empty.
SSLServerCtx	Container element; contains elements to configure Flash Media Server as an SSL (Secure Sockets Layer) client for incoming SSL connections.
SSLSessionTimeout	This element specifies in minutes how long a SSL session remains valid.
UpdateInterval	Specifies how often the server refreshes the cache content for HTTPUserInfo, in milliseconds.
WriteBufferSize	Specifies the size in kilobytes of the write buffer.

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 tunneling. 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, MaxWriteDelay

Enable

Specifies whether or not to allow HTTP tunneling connections.

These are the possible values for the `Enable` element:

Value	Description
`true`	Allow all HTTP tunneling connections. This is the default value.
`false`	Disallow all HTTP tunneling connections.
`http1.1only`	Allow only HTTP 1.1 tunneling connections.
`keepalive`	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 Flash Media Server to use port 80 for HTTP tunneling, the web server cannot use port 80.*

Example

```
<Enable>true</Enable>
```

See also

IdlePostInterval, IdleAckInterval, MimeType, WriteBufferSize, SetCookie, Redirect, MimeType, MaxWriteDelay

EnableAltVhost

Determines if 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

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/fms.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. Flash 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 Flash Media Server to use port 80 for HTTP tunneling, 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 FMSEdge (fmsedge in Linux) process listens on. When an FMSCore process is started (fmscore in Linux), it connects to an FMSEdge process on this internal port to establish a control channel. Each `HostPort` element corresponds to an FMSEdge 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 will fail because the server will try to perform an SSL handshake that the client will fail 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 "/fcs/ident" resource.

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>Flash 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>Flash 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 client attempts to make a RTMPT connection to Flash 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 (Tunneled 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 Flash 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 Flash 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 Flash 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 Flash 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.

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

```
<HTTPNull enable="true">
```

See also

HTTPIdent, HTTPTunnel, HTTPUserInfo

HTTPTunnel

Container element.

The elements in this section configure the incoming HTTP tunneling connections to the adaptor. Each edge can have different HTTPTunnel configurations.

Note: *Only one application can use a port at a time. For example, if you configure Flash Media Server to use port 80 for HTTP tunneling, the web server cannot use port 80.*

Contained elements

Enable

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 tunneling 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 not liable to 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 tunneling 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.

Default number of failures is 2.

Example

```
<MaxFailures>2</MaxFailures>
```

See also

IdleAckInterval

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

MaxWriteDelay

The HTTP tunneling protocol ensures that a server will be able to write every four seconds. Occasionally, when connections close under abnormal conditions, the notification may not reach the server, which may continue to place writes in a queue.

Anomalous connections are closed after the specified wait time. The default wait time is 40 seconds.

Example

```
<Edge name="Edge1">
    <Enable>true</Enable>
    <IdlePostInterval>512</IdlePostInterval>
    <IdleAckInterval>512</IdleAckInterval>
    <MimeType>application/x-fms</MimeType>
    <WriteBufferSize>16</WriteBufferSize>
    <SetCookie>false</SetCookie>
    <Redirect enable="false" maxbuf="16384">
        <Host port="80">:8080</Host>
    </Redirect>
    <NeedClose>true</NeedClose>
    <MaxWriteDelay>40</MaxWriteDelay>
</Edge>
```

You may want to use this sample code as a template for configuring each edge server.

See also

HTTPTunnel

MimeType

Specifies the default MIME (Multipurpose 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>
```

See also

MaxWriteDelay

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: alphanumerics 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 FMS install 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 tunneling 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

MaxWriteDelay, 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 if encrypted RTMP (RTMPE) and encrypted tunneling RTMP (RTMPTE) can be used.

Contained elements

RTMPE

RTMPE

Specifies if 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 tunneling 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 tunneling 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. For additional information, see Configure SSL.

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:

- Go 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 container.

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 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 plaintext 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

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

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 Flash Media Server applications. These settings include the size of the Server-Side Media ActionScript runtime engine, the location at which streams and shared objects are stored, and bandwidth limitations.

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 is not overridable, 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 that you wish to make non-overridable.

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 in Application.xml, see the Application.xml file installed with Flash Media Server in the *RootInstall*/conf/_defaultRoot_/_defaultVhost_ directory.

Summary of elements

Application.xml element	Description
`Access`	Container element; contains element that controls the permission levels in the Module (the libconnect.dll file).
AccumulatedIFrames	Container element; contains elements that configure intermediate frames in a live stream.
`AggregateMessages (Client)`	Specifies whether or not to send aggregate messages to clients.
AggregateMessages (Queue)	Container element; contains elements that control the size of the aggregate messages. This element also specifies, when queuing is enabled, if messages in the queue can be combined to form aggregate messages.
`Allow`	Allows or disallows the `"following and Location:"` header added with HTTP redirection.
`AllowDebugDefault`	Specifies the default value for allowing debug connections per application.
`AllowHTTPTunnel`	Configures Flash Media Server to allow tunneling connections into this application.
`Application`	Root element; this element contains all elements in Application.xml.
`Audio`	Container element; contains elements to configure the audio stream settings.
`AudioSampleAccess`	Enables access to the raw uncompressed audio data in a stream.
`AutoCommit`	Enables or disables the Shared Object Manager to automatically commit shared objects.
`Bandwidth`	Container element; contains elements to configure the bandwidth settings for server-client communications.
`BandwidthCap`	Container element; contains elements that specify the maximum bandwidth values that a user can set.
`BandwidthDetection`	Container element; contains elements that specify how data is sent to the client.

Application.xml element	Description
Bits	Contains the settings for different versions of Flash Player on the Windows and Macintosh platforms.
BufferRatio	Specifies the ratio of the buffer length used by server-side stream to live buffer.
Cache	Container element; contains elements that configure the cache setting for SWF verification.
CachePrefix	Specifies the cache prefix that is passed from the origin server to the edge server.
CacheUpdateInterval	Specifies the interval for updating cache streaming in the edge server.
Client	Container element; contains elements to configure the client.
ClientToServer (Bandwidth) Bandwidth	Container element; specifies the bandwidth settings for client-to-server communications.
ClientToServer (BandwidthCap) BandwidthCap	Container element; specifies the bandwidth settings for client-to-server communications that can be set by the user.
CombineSamples	Container element; contains elements to configure how to use sound sampling.
Connections	Container element; contains elements to configure settings for HTTP connections.
DataSize	Specifies the amount of data the server sends to the client.
Debug	Container element; contains elements that configure debug connections.
Distribute	Specifies how to distribute application instances to processes.
DuplicateDir	Specifies a backup location for shared objects and recorded stream files.
Duration	Specifies the wait time before the server notifies clients when audio stops in a stream.
EnhancedSeek	Enables the fine-tuning of the seeking performance within streams by creating a key frame.
Exception	Specifies user agents to except from authentication.
FileObject	Container element; contains element with file object setting.
FlushOnData	Specifies whether the server flushes the message queue when a data message arrives.
FolderAccess	Configures folder-level permissions for the readAccess and writeAccess functions in the Access Module.
HiCPU	Specifies the upper limit to begin sound sampling.
Host	Specifies the HTTP proxy to use.
HTTP	Container element; contains elements to configure the HTTP connections for this application.
HTTP1_0	Allows or disallows use of the HTTP 1.0 protocol.
HTTPTunnel	Container element; contains elements to configure HTTP tunneling.
IdleAckInterval	Specifies the wait time before the server responds to an idle post sent to it.
IdlePostInterval	Specifies the wait time before Flash Player sends an idle post message to the server.
Interface	Specifies the name to use as the outgoing network interface.
Interval	Specifies the interval for sending silence messages when no audio is being published to a live stream.
JSEngine	Container element; contains the elements in this section that configure the JavaScript engine.
KeyFrameInterval	Specifies the time interval for saving keyframes in a FLV file.

Application.xml element	Description
LifeTime	Specifies the lifetime of stateless core processes.
Live (StreamManager)	Container element; contains elements that specify the configuration of intermediate frames in a live stream.
Live (MsgQueue)	Container element; contains elements that specify the configuration of live audio.
LoadOnStartup	Specifies whether or not to load this application when the server starts.
LockTimeout	Specifies the time-out value before unlocking a shared object file.
LoCPU	Specifies the lower limit to halt sound sampling.
Max	Specifies the maximum number of HTTP redirections allowed.
MaxAggMsgSize	Specifies the maximum size in bytes of the aggregate messages created from the message queue, when aggregate messages are enabled.
MaxAppIdleTime	Specifies the maximum time an application instance can be idle.
MaxAudioLatency	Specifies that live audio should be dropped after a set amount of time.
MaxBufferRetries	Specifies default buffer length for live audio and video.
MaxCores	Specifies the maximum number of core processes for an application.
MaxFailures	Specifies the maximum number of failures for a core process.
MaxGCSkipCount	Specifies the maximum number of times that the server will skip garbage collection (GC) when the JS engine is busy.
MaxMessageSizeLosslessVideo	Specifies the maximum size of messages for screen-sharing packets.
MaxPendingDebugConnections	Specifies the maximum number of pending debug connections. allowed.
MaxProperties	The maximum number of properties that can be set per shared object.
MaxPropertySize	The maximum size in bytes for each property of a shared object.
MaxQueueDelay	Specifies how often the server will flush the message queue, in milliseconds.
MaxQueueSize	Specifies how often the server will flush the message queue, in bytes.
MaxRate	Specifies the maximum rate at which the server sends data to the client.
MaxSamples	Specifies the maximum number of samples that can be combined into a message.
MaxSize	Specifies the maximum size of intermediate frames a live stream can hold in the buffer.
MaxStreamsBeforeGC	Specifies that a GC should be forced if the stream list grows over set value.
MaxTime	Specifies the maximum duration of intermediate frames a live stream can hold in the buffer.
MaxTimeOut (Connections) Connections	Container element; specifies the maximum time for a transfer to be completed.
MaxTimeOut (JSEngine) JSEngine	Container element; specifies the maximum time a script can take to execute a Java server function.
MaxUnprocessedChars	Specifies the amount of data that can be received from the XML server before XMLSocket closes the connection.
MaxWait	Specifies how long, in seconds, the server waits before sending data to the client.
MimeType	Specifies the default MIME-type header sent on tunnel responses.

Application.xml element	Description
MinBufferTime (Live)	Specifies the default buffer length for the live audio and video queue.
MinBufferTime (Recorded)	Specifies the default buffer length for audio and video.
MinGoodVersion	Specifies the minimum accepted version of SWF verification allowed by the server.
MsgQueue	Container element; contains elements that configure live and recorded audio.
NetConnection	Container element; specifies object encoding to use for SSAS NetConnection.
Password	Specifies whether the server is notified when an audio transmission ending on a stream is encountered.
ObjectEncoding	Specifies default object encoding to use for SSAS NetConnection.
OutChunkSize	Specifies the RTMP chunk size to use in all streams for this application.
OverridePublisher	Deprecated; see PublishTimeout.
Password	Specifies the password for connections to the edge.
Port	Specifies the proxy port to connect to if not specified.
Prioritization	Specifies whether outgoing messages are prioritized by message type when sending across server-to-server connection.
Process	Container element; contains elements to configure the process and recovery settings for applications.
Proxy	Container element; contains elements to configure the HTTP proxy.
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.
Queue	Container element; configures the settings of the message queue.
Recorded	Container element; specifies aspects of buffer length.
RecoveryTime	Specifies the recovery time for a core.
Redirect	Container element; contains elements to configure HTTP redirection.
ResyncDepth	Specifies the resyncing interval for shared object files.
Reuse	Specifies whether or not to close the HTTP connection after each transfer.
RollOver	Specifies the time length a core process is in use.
RuntimeSize	Specifies the maximum size for the script engine.
Scope	Specifies the process scope in which the application runs.
ScriptLibPath	Contains a list of paths the Java Server engine can search to resolve a script file.
SendDuplicateOnMetaData	Specifies if an onMetaData message is sent at the beginning of the video file for all commands.
SendDuplicateStart	Specifies if the status message NetStream.Play.Start is sent for all commands.
SendSilence	Container element; contains elements to configure the sending of silence messages.
Server	Container element; contains element that specifies the ratio of the buffer length used by server-side stream to live buffer.
ServerToClient (Bandwidth) Bandwidth	Specifies the bandwidth settings for server-to-client communications.

Application.xml element	Description
ServerToClient (BandwidthCap) BandwidthCap	Specifies the maximum bandwidth a user can set for data sent from the server to the client.
SharedObjManager	Container element; contains elements to configure the Shared Object Manager of an application.
StorageDir	Specifies the locations where recorded streams and shared objects are stored.
StreamManager	Container element; contains the Stream Manager settings for the application.
Subscribers	Specifies a base number of subscribers required before implementing sound sampling.
SWFFolder	Specifies a single folder or a list of folders containing copies of client SWF files that can be verified for connecting to the application.
SWFVerification	Container element; contains elements that specify how a SWF file connecting to an application is verified.
ThrottleBoundaryRequest	Controls the maximum number of concurrent boundary requests per recorded stream.
ThrottleDisplayInterval	Controls the interval at which the server displays the throttle queue length.
ThrottleDisplayInterval	Controls the interval at which the server displays the throttle queue length.
Tunnel	Specifies whether or not to tunnel all operations through a given HTTP proxy.
TTL	Specifies in minutes how long each SWF file remains in the cache. The default value is 1440 minutes (24 hours).
Type	Specifies the type of proxy being connected to.
UnrestrictedAuth	Allows or disallows sending the user name/password with each HTTP redirection.
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.
UserAgent	Specifies the version dependency settings for clients that use different versions of Flash Player on different platforms.
UserAgentExceptions	Container element; contains element that specifies exceptions to SWF verification.
Username	Specifies the user name for connections to the proxy.
Verbose	Enables or disables the use of verbose information during HTTP operations.
VideoSampleAccess	Enables access to the raw uncompressed video data in a stream.
VirtualDirectory	Specifies virtual directory mappings for Server-Side ActionScript File objects and video files.
WriteBufferSize	Specifies the size of the write buffer.
XMLSocket	Container element; contains element that specifies the amount of data XMLSocket accepts from XML server before closing the connection.

Access

Container element.

The Access plug-in consists of the libconnect.dll file. It intercepts and examines each connection request to Flash Media Server to determine whether the connection should be accepted or rejected.

Contained element

FolderAccess

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 element

MaxTime, MaxSize

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 element that control the size of the aggregate messages.

This element also specifies, when queuing is enabled, if messages in the queue can be combined to form aggregate messages. When the `enabled` attribute is set to `true` (the default value), the server will create 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 tunneling 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, the Flash Player has to negotiate a connection to Flash Media Server through an edge server, or use the HTTP protocol to transmit RTMP packets (called *HTTP tunneling*) 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 tunneling connections.
`false`	Disallows tunneling 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`, `AllowHTTP-Tunnel`, `Client`, `Debug`, `HTTP`, `SWFVerification`

Audio

Container element.

The elements in this section specify the settings for audio streams on the server.

Contained element

`CombineSamples`, `SendSilence`, `Password`

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 `<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 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 if shared objects are automatically committed when they have been changed. Setting this element to `false` disables the 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 element

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 element

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	Impact
BandwidthDetection	Set the enabled 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 MaxWait 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 also forces the client to wait longer.

Example

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

Contained element

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:\fms\flvs\foo.flv. data/?IP?</CacheDir>
```

resolves to:

```
data/xxx.xxx.xxx.xxx/c/fms/flvs/foo.flv
```

```
<CachePrefix type="path">?APPINST?/data</CacheDir>
```

resolves to:

```
app1/inst1/data/c/fms/flvs/foo.flv
```

```
<CachePrefix type="path">origin1/data/</CacheDir>
```

resolves to:

```
origin1/data/c/fms/flvs/foo.flv
```

See also

StorageDir, DuplicateDir, CacheUpdateInterval

CacheUpdateInterval

This element defines the wait interval for updating the 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

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 element

Bandwidth, BandwidthCap, BandwidthDetection, MsgQueue, HTTPTunnel, MaxMessageSizeLosslessVideo, OutChunkSize,

ClientToServer (Bandwidth)

Specifies the maximum bandwidth the client can use for sending data upstream to the server. The default bandwidth is 1,250,000 bytes per second.

Example

```
<ClientToServer>1250000</ClientToServer>
```

See also

ServerToClient (Bandwidth)

ClientToServer (BandwidthCap)

Specifies the maximum bandwidth a user can set for data to be sent upstream from the client to the server. The default bandwidth is 100,000,000 bytes per second.

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 element

`LoCPU`, `HiCPU`, `MaxSamples`, `Subscribers`

Connections

Container element.

The elements in this section configure the HTTP connections for this application.

Contained element

`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 element

`AllowDebugDefault`, `MaxPendingDebugConnections`

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. For more information, see Configure how applications are assigned to server 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
`vhosts`	All instances of applications in a virtual host run together in a process.
`apps`	All instances of an application run together in a process.
`insts`	Each application instance runs in its own process. This is the default value. If you choose this value, you must also set the `Distribute numprocs` attribute to a value greater than 1.
`clients`	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 client, 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

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

```
<DuplicateDir appName="true">c:\backupSharedObjects</DuplicateDir>
```

```
<DuplicateDir appName="true">c:\backupStreams</DuplicateDir>
```

To include the application name in the paths for the backup files, change the `appName` attribute to `"true"`.

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

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 FMS server will generate 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 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

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.

The `VirtualDirectory` element nested within this container configures the `JSEngine` file object settings.

Contained element

VirtualDirectory

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 for readAccess and writeAccess that can be set by the access module in order to access the streams and sharedObjects. 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

HiCPU

This element instructs the server to start combining samples when the CPU utilization is higher than the specified percentage of the CPU resources. 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 element

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 tunneling (sending RTMP packets through HTTP).

The tunneling 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 the 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 representative settings for these parameters. These settings recommend that you set the intervals to correspond to low, medium, or high latency.

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>
    <IdlePortInterval>512</IdlePostInterval>
    <IdleAckInterval>512</IdleAckInterval>
    <MimeType>application/x-fcs</MimeType>
    <WriteBufferSize>16</WritebufferSize>
</HTTPTunnel>
```

Contained element

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 the silence message transmission.

Example

```
<Interval>3</Interval>
```

See also

SendSilence

JSEngine

Container element.

The elements nested within this container configure the JavaScript engine.

In the server Application.xml configuration files, 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:

```
<JSEngine>
    <config>
    <user_name>jdoe</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 element

`RuntimeSize`, `MaxGCSkipCount`, `MaxTimeOut (JSEngine)`, `ScriptLibPath`, `FileObject`, `XMLSocket`, `NetConnection`

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 element

`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 element

AccumulatedIFrames, `Queue`, `PublishTimeout`

Live (MsgQueue)

Container element.

The elements nested within this container configure live audio.

Contained element

`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 startup 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 startup.

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 time-out 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 the CPU utilization is lower than the specified percentage of the CPU resources. Default percentage of utilization is 60.

Example

```
<LoCPU>60</LoCPU>
```

See also

Subscribers, HiCPU, MaxSamples

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 time specified. Time is expressed in milliseconds.

Example

```
<MaxAudioLatency>2000</MaxAudioLatency>
```

See also

MinBufferTime (Live)

MaxBufferRetries

Specifies default buffer length for live audio and video.

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 functionality 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.

Example

```
<MaxCores>0</MaxCores>
```

See also

LifeTime, RollOver

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</MaxSGCSkipCount>
```

See also

RuntimeSize, MaxTimeOut (JSEngine), ScriptLibPath,FileObject, XMLSocket, NetConnection

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

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

Specifies 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`

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`, `MaxGCSkipCount`, `ScriptLibPath`, `FileObject`, `XMLSocket`, `NetConnection`

MaxTimeOut (Connections)

This element defines the maximum time for a transfer to be completed. The default time is 60 seconds.

Operations such as DNS lookups 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)

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`, `MaxGCSkipCount`, `ScriptLibPath`,`FileObject`, `XMLSocket`, `NetConnection`

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

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 (Multipurpose 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. 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

MsgQueue

Container element.

The elements nested within this container configure live and recorded audio.

Contained element

Live (MsgQueue), Recorded, Server

NetConnection

Container element.

The element nested within this container specifies object encoding to use for SSAS NetConnection.

Contained element

ObjectEncoding

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 element

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

OutChunkSize

Specifies the RTMP chunk size to use in all streams for this application. Stream content breaks into chunks of this size in bytes. Larger values reduce CPU usage, but also commit to 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. Default 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`

Port

Specifies the proxy port to connect to, if it 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 multipoint 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 Distribute tag must be a scope that is lower in order than the value in the Scope tag (for example, if the value of Scope is adaptor, the value of Distribute can be vhosts, apps, insts, or clients). Distribution may be turned off by setting numproc to 0 or 1.
LifeTime	Specifies the lifetime of core processes. Process rollover happens only when the Scope element is set to inst.
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 element

Scope, Distribute, LifeTime, MaxFailures, RecoveryTime,

Proxy

Container element.

The elements nested within this container configure the HTTP Proxy settings.

Contained element

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, Flash 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.

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 element

`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 element

`MinBufferTime (Recorded)`

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 the Flash 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 element

`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.

Example

```
<Rollover>0</RollOver>
```

See also

MaxCores

RuntimeSize

Specifies the maximum size in kilobytes that a particular 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 and upper limits on the size of the JavaScript engine are 10 kilobytes and 51200 kilobytes (50 megabytes). 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), ScriptLibPath, FileObject, XMLSocket, NetConnection

Scope

This element determines the level at which application instances are assigned to core processes.

Starting Flash Media Server starts a process called FMSMaster.exe (Windows) or fmsmaster (Linux). Application instances run in processes called FMSCore.exe (Windows) fmscore (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 `Distribute numprocs` attribute to a value greater than 1.

Example

```
<Scope>vhost</Scope>
```

See also

`Distribute`, `LifeTime`, `MaxFailures`, `RecoveryTime`

ScriptLibPath

This element is 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`, `MaxGCSkipCount`, `MaxTimeOut (JSEngine)`, `FileObject`, `XMLSocket`, `NetConnection`

SendDuplicateOnMetaData

Specifies if an `onMetaData` message is sent at the beginning of the video file for all commands, including play, seek, and unpause. The default value is `true`.

The following values are available:

- `true` sends `onMetaData` for play, seek, and unpause commands.
- `false` sends `onMetaData` for play only.

- `once` falls back to FMS 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 if 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 element

`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 maximum 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.

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.

Example

```
<ServerToClient>10000000</ServerToClient>
```

See also

ClientToServer (BandwidthCap)

SharedObjManager

Container element.

The elements nested within this container configure the Shared Object Manager setting of an application.

Contained element

StorageDir, DuplicateDir, ResyncDepth, LockTimeout, AutoCommit, MaxProperties, MaxPropertySize

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

StreamManager

Container element.

The elements in this section configure the Stream Manager settings for this application.

Contained elements

StorageDir, DuplicateDir, CachePrefix, CacheUpdateInterval, MaxBufferRetries, ThrottleBoundaryRequest, ThrottleLoads, ThrottleDisplayInterval, EnhancedSeek, KeyFrameInterval, MaxStreamsBeforeGC, Audio, Live (StreamManager), SendDuplicateStart, SendDuplicateOnMetaData

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 that can be authenticated for connecting this application to this server.

These SWF files are used to authenticate connecting SWF files. The default value 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`.

To allow SWFs from two different directories named “SWFs” and “D”, use
`C:\apps\app1\SWFs;D:\apps\app1\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.

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

The following table lists the values available for the `SWFVerification` element.

Element	Description
`SWFVerification`	Set the `enabled` attribute to `true` or `false` to turn this feature on or off. The default value is `false`.
`SWFFolder`	A single folder or a semicolon-delimited list of folders that contain copies of client SWF files for an application. These SWF files are used to verify connecting SWF files. The default value is the application's folder appended with /SWFs. For example, for an application called myApplication, if there isn't a value set for this element, verifying SWF files should be placed in the applications/myApplication/SWFs folder.
`MinGoodVersion`	Specifies the minimum version of SWF verification to accept. The default value is 0, which allows current and all future versions.
`UserAgentExceptions`	Container. Contains the `Exception` element..
`Exception`	A user agent to except from verification. 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 more information, see the comments in the Application.xml file.
`Cache`	Container; contains the `TTL` and `UpdateInterval` elements. Configures how the cache behaves. SWFVerification data is stored in the cache.

Example

```
<SWFVerification enabled="false">
    <SWFFolder></SWFFolder>
```

```
    <MinGoodVersion></MinGoodVersion>
    <UserAgentExceptions>
        <Exception to="" from=""/>
    </UserAgentExceptions>
    <Cache>
        <TTL></TTL>
        <UpdateInterval></UpdateInterval>
    </Cache>
</SWFVerification>
```

Contained elements

SWFFolder, MinGoodVersion, UserAgentExceptions, Cache

ThrottleBoundaryRequest

Controls the maximum number of concurrent boundary requests per recorded stream. When streaming through a proxy server, the boundary information of 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

```
<ThrottleDisplayInterva>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 the Flash Player platform is Windows or Macintosh. Setting the value 0x01 will configure the player and platform for silent messages.

Contained element

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 element

`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 (under the `JSEngine` node) and for video files for a vod application (under the `StreamManager` node).

Virtual directories lets you specify file directories for different applications. If the beginning portion of a file path matches the specified virtual directory, then the storage location of the file becomes the file path of the virtual directory.

In an application-specific Application.xml file, you can use the `VirtualDirectory` element to specify a directory to use for vod applications. Put video files in this directory to make them instantly streamable, without writing any code.

For more information, see comments in the Application.xml file and Mapping virtual directories to physical directories.

Example

```
<VirtualDirectory>virtual_dir_name;physical_dir_path</VirtualDirectory>
```

See also

`FileObject`

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 element

`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 Flash Media Server 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 stream's 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 enables or disables the log files. Elements to configure the log files are in the Logger.xml file.

To see the element structure and default values in Logger.xml, see the Logger.xml file installed with Flash Media Server in the *RootInstall*/conf/ directory.

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.*

Summary of elements

Logger.xml element	Description
`Access`	Container element; contains elements used to configure the Access log file settings.
`Application`	Container element; contains elements to configure the Application log file settings.
`AuthEvent`	Container element; contains elements to configure the Authorized Events log file settings.
`AuthMessage`	Container element; contains elements to configure the Authorized Messages log file settings.
`Delimiter`	Specifies which delimiter to use when separating the fields in the log file.
`Diagnostic`	Container element; contains elements to configure the diagnostic log file settings.
`Directory`	Specifies how many lines to write to log file before repeating the field headers.
`DisplayFieldsHeader`	Specifies how many lines to write to the log file before repeating the field headers.
`EscapeFields`	Formatting element; specifies whether or not unsafe characters in the log file are escaped.
`Events`	Specifies the events written to the Access log file.
`Fields`	Specifies which fields for an event are logged in the Access log file.
`FileIO`	Container element; contains elements to configure the File IO log file settings.
`FileName`	Specifies the name of the log files.
`History`	Specifies the maximum number of log files to keep.
`HostPort`	Specifies the IP and port number of the log server.
`Logger`	Root element; this element is a container for all the other elements.
`LogServer`	Container element; contains elements to configure the server to send messages to a remote log server.
`MaxSize`	Specifies the maximum size of the log files.

Logger.xml element	Description
QuoteFields	Formatting element; specifies whether or not to use quotation marks to surround those fields in the log file that include a space.
Rename	Specifies new name for log files when rotation occurs.
Rotation	Container element; contains elements to configure the rotation of the log files.
Schedule	Specifies how frequently the log files are rotated.
ServerID	Identifies by IP address the server whose logged events are being recorded.
Time	Specifies the time zone for a log file.

Access

Container element.

The elements nested within this container configure the Access log settings.

Contained elements

LogServer, Directory, FileName, Time

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 Authorized Events 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 Authorized Messages log file settings.

Contained elements

Directory, Time. Rotation

Delimiter

Specifies whether or not to use single quotation marks (') 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 ("")
- commas (,)
- colons (:)
- hyphens (-)

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

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 ({ }); bars (|); 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

Events are written to the log file.

The following table lists the events recorded in the Access log file. Events are logged in a semicolon-separated list. The keyword * instructs Flash Media Server to log all events.

Event	Category	Description
app-start	application	Application instance starts.
app-stop	application	Application instance stops.
connect	application	Client connects to the server.
connect-pending	application	Client connects to the server, waiting for the script to authenticate.
disconnect	application	Client disconnects.
pause	application	Client pauses a recorded stream.
play	application	Client plays a recorded or live stream.
publish	application	Client publishes a live stream.
record	application	Client begins recording a stream.
recordstop	application	Client stops recording a stream.
seek	application	Client jumps to a new location within a recorded stream.
server-start	application	Server has started.
server-stop	application	Server has stopped.
stop	application	Client stops playing a recorded or live stream or stops publishing a live stream.
unpause	application	Client resumes a stream.
unpublish	application	Client unpublishes a live stream.
vhost-start	application	A virtual host has started.
vhost-stop	application	A virtual host has stopped.

The following events display a status code.

Field	Status Code	Description
connect-pending	100	Waiting for the application to authenticate.
connect	200	Successful connection.
	302	Application currently unavailable.
	400	Bad request; client connected to server using an unknown protocol.

Field	Status Code	Description
	401	Connection rejected by the application script.
	403	Connection rejected by access module.
	404	Application not found.
	409	Resource limit exceeded.
	413	License limit exceeded.
	500	Server internal error.
	502	Bad gateway.
	503	Service unavailable; for example, too many connections pending for authorization by access module.
play	200	Successful.
	400	Bad request (invalid arguments).
	401	Access denied by application.
	403	Play forbidden by stream module.
	404	Stream not found.
	415	Unsupported media type.
	500	Server internal error.
publish	200	Successful.
	400	Bad request (invalid arguments).
	401	Access denied by application.
	409	Stream is already being published.
	415	Unsupported media type.
	500	Server internal error.
stop	200	Successful.
	408	Stream stopped because client disconnected.

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.

The following table is a complete list of fields associated with events in the Access log file. Not every field is associated with each event in the log file.

Field	Event(s)	Description
x-event	application	Type of event.
x-category	application	Event category.
date	application	Date on which the event occurred.
time	application	Time at which the event occurred.
tz	application	Time-zone information.
x-ctx	application	Event-dependent context information.
x-pid	application	Server process ID.
x-cpu-load	application	CPU load.
x-mem-load	application	Memory usage (as reported by the getServerStats() method).
x-adaptor	application	Adaptor name.
x-vhost	application	Vhost name.
x-app	application	Application names.
x-appinst	application	Application instance names.
c-ip	application	Client IP address.
c-proto	application	Connection protocol: RTMP or RTMPT.
s-uri	application	URI of the application.
c-referrer	application	URI of the referrer.
c-user-agent	application	User agent.
c-client-id	application	Client ID.
cs-bytes	application	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 the value of cs-bytes in the connect event from the value of cs-bytes in the disconnect event.
sc-bytes	application	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 the value of sc-bytes in the connect event from the value of sc-bytes in the disconnect event.
x-sname	application	Stream name.
x-file-size	application	Stream size in bytes.
x-file-length	application	Stream length, in seconds.
x-spos	application	Stream position.
cs-stream-bytes	application	This field shows the number of bytes transferred from the client to the server per stream. To calculate the bandwidth usage per stream, subtract the value of cs-stream-bytes in the publish event from the cs-stream-bytes in the unpublish event.

Field	Event(s)	Description
sc-stream-bytes	application	This field shows the number of bytes transferred from the server to the client per stream. To calculate the bandwidth usage per stream, subtract the value of sc-stream-bytes in the play event by the value of sc-stream-bytes in the stop event.
cs-uri-stem	application	Stem portion of the s-uri (omitting query) field.
cs-uri-query	application	Query portion of s-uri.
x-sname-query	application	Query portion of stream URI specified in play or publish.
x-file-name	application	Full path of the file representing x-sname stream.
x-file-ext	application	Stream type, such as FLV or MP4.
s-ip	application	IP address or addresses of the server.
x-duration	application	Duration of a stream or session event.
x-suri-query	application	Same as x-sname-query.
x-suri-stem	application	This is a composite field: cs-uri-stem + x-sname + x-file-ext.
x-suri	application	This is a composite field: cs-uri-stem + x-sname + x-file-ext + x-sname-query.
x-status	application	For a complete description of the x-status codes and descriptions, see Diagnostic Log Messages.
x-sc-qos-bytes	application	Bytes transferred from server to client for quality of service

See also

Events

FileIO

Container element.

The elements in this section configure the File IO 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 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, and 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

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 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

```
<Schedule type="daily"></Schedule>
```

If the `type` attribute is `daily`, the server rotates the log files every 24 hours.

```
<Schedule type="hh:mm"></Schedule>
```

If the `type` attribute is `hh:mm`, the timestamp `00:00` causes the file to rotate every midnight.

```
<Schedule type="duration"></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.

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 `Time field` 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

The `Logging` container in the Server.xml file.

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 Flash Media Server in the *RootInstall*/conf/ directory.

Summary of elements

Server.xml element	Description
`Access`	Container element; contains the elements used to configure the Access log settings.
`ACCP`	Container element; contains elements used to configure the Admin Core Communication Protocol (ACCP).
`ActiveProfile`	Specifies the limits enforced by the server on each license key.
`Admin`	Container element; contains the elements used to configure the RTMP protocols for the FMSAdmin.exe process.
`AdminElem`	Specifies the format used to display an element name in an HTTP command.
`AdminServer`	Container element; contains elements used to configure the Flash Media Administration Server.
`Allow`	Specifies the administrator connections that should be accepted.
`Application`	Container element; the `Enable` element in this container enables or disables the log file.
`ApplicationGC`	Specifies in minutes how often to check for and remove unused applications.

Server.xml element	Description
AuthEvent	Container element. The Enable element nested within the AuthEvent container specifies whether logging of events from the authorization adaptor is enabled.
AuthMessage	Container element. The Enable element nested within the AuthMessage container specifies whether logging of messages from the authorization adaptor is enabled.
AutoCloseIdleClients	Specifies if idle clients should be closed automatically.
Cache	Container element; contains elements that configure the cache setting for SWF verification.
CheckInterval	Specifies the interval at which the server checks for active client connections.
Connector	Container element; contains elements used to configure the connector subsystem. Provides connectors that allow application scripts to connect to other Flash Media Servers or HTTP servers.
Core	Container element; contains elements used to configure the protocols for the FMSCore.exe process.
CoreExitDelay	Specifies the wait time for an idle core to exit on its own before it is removed from the server, in seconds.
CoreGC	Specifies how often, in seconds, to check for and remove idle cores.
CoreTimeout	Specifies the timeout value, in seconds, for detecting unresponsive cores.
CPUMonitor	Specifies, in seconds, how often to monitor CPU usage.
Deny	Specifies administrator connections that should be ignored.
Diagnostic	Container element; contains element to enable the diagnostic log file.
ECCP	Container element; contains elements to configure the edge core communication protocol.
Edge	Container element; contains elements to configure the RTMP protocol for the FMSEdge.exe process.
EdgeCore	Container element; these elements control the IPC message queues used by edge and core processes to communicate with each other.
Enable	A Boolean value that enables or disables the Access logs, Application logs, or diagnostic logs.
FileCheckInterval	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.
FileIO	Container element; contains an element that specifies if file IO logging is enabled.
FLVCache	Container element; contains elements that control the size and features of the FLV cache.
FLVCacheSize	Specifies the percentage of total physical memory on the system that the FLV cache may occupy.
FreeMemRatio	Sets the maximum percentage of total memory that the total pool size may occupy.
FreeRatio	Specifies the percentage of the message cache to be consumed by the free list on a per-thread basis.
GCInterval	Specifies how often to remove idle handles.
GID	Contains the group ID of the server process.
GlobalQueue	Container element; these elements control the IPC message queue used by processes to communicate with each other.
GlobalRatio	Specifies the percentage of the message cache that can be consumed by the free list on a global basis.
HandleCache	Container element; contains elements that configure the size and features of the handle cache.
HeapSize	Specifies the maximum size of the shared memory heap used for a IPC message queue.
HostPort	Specifies the IP address and port that the Flash Media Administration Server binds to.

Server.xml element	Description
HTTP	Container element; contains elements to configure the HTTP connector, which is used by remote sites for accessing Flash Media Server.
IdleTime	Specifies the amount of time to wait before releasing cached handles.
IPCQueues	Container element; contains elements to configure the IPC (interprocess communication) queues.
LargeMemPool	Container element; contains elements to configure the large memory pool.
LicenseInfo	Specifies key information about server licensing.
LicenseInfoEx	Contains license keys added using the Administration Console. For more information about license keys, see the LicenseInfo element.
LocalHost	Specifies the Flash Media Server IP loopback address.
Logging	Container element; contains elements to perform the overall logging configuration.
Mask	Contains a three-digit octal value used by the Linux `umask` (user permissions mask) command to set a file creation mask.
Master	Container element; contains elements to configure the resource limits for the master server.
MaxAge	Specifies the maximum reuse count before freeing the cache unit.
MaxCacheSize	Specifies the maximum size of the cache.
MaxCacheUnits	Specifies the maximum free units in the cache.
MaxConnectionQueueSize	Specifies the maximum number of connection requests that can be pending.
MaxConnectionRate	Specifies the maximum number of incoming connections per second that the server's socket listener accepts.
MaxConnectionThreads	Specifies the maximum number of threads used to process connection requests.
MaxConnectionThreads	Specifies the maximum number of threads used to process connection requests.
MaxIdleTime	Specifies the maximum idle time allowed before client is disconnected.
MaxIOThreads	Specifies the maximum number of threads that can be created for I/O processing.
MaxKeyframeCacheSize	Specifies the maximum number of keyframes per FLV file in the cache.
MaxNumberOfMessages	Specifies the maximum number of messages that the buffer holds before the messages are committed to file.
MaxQueueSize	Specifies the maximum number of pending IPC messages that can be in queue at a given time.
MaxSize (FLVCache)	Specifies the maximum size of the FLV cache.
MaxSize (HandleCache)	Specifies the maximum number of handles to cache.
MaxSize (RecBuffer)	Specifies the maximum size to which the buffer can grow before messages are committed to file.
MaxTimestampSkew	Specifies the maximum gap between two adjacent messages when comparing the messages' timestamps with the real time.
MaxUnitSize	Specifies the maximum size, in kilobytes, of a memory chunk allowed in a memory pool. The default size is 16 KB.
MessageCache	Container element; contains elements to control how the message cache keeps messages used by Flash Media Server.
MinConnectionThreads	Specifies the minimum number of threads in the pool for I/O operations.

Server.xml element	Description
MinGoodVersion	Specifies the minimum accepted version of SWFVerification allowed by the server.
MinIOThreads	Specifies the minimum number of threads that can be created for I/O operations.
MsgPoolGC	Specifies how often the server checks for and removes content in the global message pool.
NetworkingIPv6	Enables or disables IPv6.
NumCRThreads	Specifies the number of completion routine threads for edge server I/O processing in WIndows 32-byte systems.
Order	Specifies the order in which to evaluate the `Allow` and `Deny` elements.
Process (AdminServer)	Container element; contains elements that configure UID and GID for the Administration Server.
Process (Server)	Container element; contains elements that configure UID and GID for all server processes.
Protocol	Container element; contains elements to configure protocols and their reception.
PublicIP	Specifies that if the system has multiple network ports, a public IP address should be created.
RecBuffer	Container element; contains elements that configure the buffer for FLV recording.
ResourceLimits	Container element; contains elements to specify the maximum resource limits of the server.
Root	Root element; contains all other elements in Server.xml.
RTMP (AdminServer)	Container element; contains elements to configure different versions of RTMP.
RTMP (Connector)	Container element; contains elements to configure the RTMP connector.
RTMP (Protocol)	Container element; contains elements to configure the RTMP protocol.
RTMPE	Specifies if RTMPE (Encrypted Real-Time Messaging Protocol) can be used.
Scope	Determines whether or not to write a log file for each virtual host or write only one log file for the server.
SegmentsPool	Container element; contains elements that configure how the segments pool caches segments of video files.
Server	Container element; contains elements that configure the server.
ServerDomain	Specifies the host name (with domain) of the server machine.
Services	Container element; contains elements to control the IPC message queue used by edge and core processes to communicate with each other.
SmallMemPool	Container element; contains elements to configure the small memory pool.
SocketGC	Specifies how often to check for and remove inactive sockets.
SocketOverflowBuckets	Specifies the number of overflow buckets if all slots in socket table are in use.
SocketRcvBuff	The size of the client socket receive buffer, in bytes.
SocketSndBuf	The size of the client socket send buffer, in bytes.
SocketTableSize	Specifies the size of the direct access socket table for quick lookup.
SSL	Container element; contains elements to configure the server as an SSL-enabled client for secure communications.
SSLCACertificateFile	Specifies the name of a file that contains one or more CA certificates in PEM encryption format.
SSLCACertificatePath	Specifies the name of the directory containing one or more CA certificates.

Server.xml element	Description
SSLCipherSuite	Specifies the encryption ciphers to secure outgoing communications.
SSLCACertificateFile	Container element; contains elements to configure the server as an SSL (Secure Sockets Layer) client for outgoing SSL connections.
SSLRandomSeed	Specifies the number of bytes of entropy to use for seeding the pseudorandom number generator (PRNG).
SSLSessionCacheGC	Specifies how often to flush expired sessions from the server-side SSL session cache.
SSLVerifyCertificate	Specifies whether or not to verify the certificate returned by the server being connected to.
SSLVerifyDepth	Specifies the maximum depth in the certificate chain that the server is willing to accept.
SWFFolder	Specifies a folder containing SWF files that are authenticated for connecting to any application on this server.
SWFVerification	Container element; contains elements that configure how SWF files connecting to an application are verified.
TerminatingCharacters	Specifies the final characters of each log entry in log files.
ThreadPoolGC	Specifies how often to check for and remove unused I/O threads.
Time	Specifies the time field in a log file.
TrimSize	Specifies a percentage of cached handles to remove.
TTL	Specifies in minutes how long each SWF file remains in the cache.
UID	Contains the server process user ID.
UpdateAccessTimeInterval	Specifies how often to modify the access time of the video cache file in the proxy server when the video file is actively used by the server.
UpdateInterval	Specifies how often thread statistics are collected.

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). The Flash Media Administration Server and active cores use ACCP for communications. This protocol is also used for collecting performance metrics and issuing administrative commands to Flash Media Server cores.

When administrators connect to the server with the Administration Console, they are connecting to the Flash Media Administration Server, which in turn connects to Flash Media Server.

Contained elements

`MinIOThreads`, `MaxIOThreads`, `SocketTableSize`, `SocketOverflowBuckets`

ActiveProfile

Specifies the limits enforced by the server on each license key (set in LicenseInfo element). Select a profile to determine bandwidth, connection, and other licensed limits.

See also

`Process (Server)`, `Mask`, `LicenseInfo`

Admin

Container element.

The elements nested within the `Admin` container configure the RTMP (Real-Time Messaging Protocol) for the FMSAdmin.exe process. RTMP is the protocol used for communication between Flash Player and Flash 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 Get Active VHost feature is an extended administration command to list only the active VHosts. Get Active VHost also includes a related command, `GetActiveVHostStats`, which allows administrators to query the statistics information for all active VHosts with a single command. 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 (AdminServer)`, `Allow`, `Deny`, `Order`

AdminServer

Container element.

The elements nested within the `AdminServer` container configure the Flash Media Administration Server.

Contained elements

`RTMP (AdminServer)`, `HostPort`, `SocketGC`, `Process (AdminServer)`, `AdminElem`, `Allow`, `Deny`, `Order`

Allow

Specifies the administrator connections that are to be accepted. By default, a client can connect to Flash 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

Application

Container element.

The Enable element nested within the Application container enables the Application log file.

Contained element

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.

Example

```
<ApplicationGC>5</ApplicationGC>
```

See also

CPUMonitor, ThreadPoolGC, MsgPoolGC, FLVCacheSize, ResourceLimits, MaxAppIdleTime

AuthEvent

Container element. The Enable element nested within the AuthEvent container enables logging of events from the authorization adaptor.

Contained element

Enable

AuthMessage

Container element. The Enable element nested within the AuthMessage container enables logging of messages from the authorization adaptor.

Contained element

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 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

`MaxIdleTime`

Connector

Container element.

The elements nested within the `Connector` container configure the connector subsystem. Flash Media Server provides connectors that allow application scripts to connect to other Flash Media Servers or HTTP servers.

Contained elements

HTTP, RTMP (Connector)

Core

Container element.

The elements nested within the Core container configure the RTMP protocol for the FMSCore.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

RecBuffer, ThreadPoolGC, MsgPoolGC, ApplicationGC

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 element

Enable

ECCP

Container element.

The elements nested within the `ECCP` container configure ECCP (Edge Server-Core Server Communication Protocol). Flash Media Server edge processes and Flash Media Server core processes use ECCP to migrate socket connections and proxy nonmigrated connections.

Contained elements

MinIOThreads, MaxIOThreads, SocketTableSize, SocketOverflowBuckets, CoreTimeout

Edge

Container element.

The elements nested within the `Edge` container configure the RTMP protocol for the FMSEdge.exe (fmsedge) process.

Contained elements

MinIOThreads, MaxIOThreads, SocketTableSize, SocketOverflowBuckets

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 six elements named Enable: in the Access, Diagnostic, Application, AuthEvent, AuthMessage, and FileIO containers within the Logging container.

This element enables or disables the various, individual 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

FileCheckInterval

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; a very large number means the server will not refresh what is in the cache even when there is a file change.

Example

```
<FileCheckInterval>120</FileCheckInterval>
```

See also

MaxSize (FLVCache), UpdateAccessTimeInterval, MaxKeyframeCacheSize

FileIO

Container element.

The Enable element nested within the FileIO container enables logging from the File plug-in.

Contained element

Enable

FLVCache

Container element.

Contains elements that control the size and features of the FLV cache.

Contained elements

FileCheckInterval, MaxSize (FLVCache), UpdateAccessTimeInterval, MaxKeyframeCacheSize

FLVCacheSize

Specifies the maximum size of the FLV cache in megabytes. The FLV cache size is specified as a percentage of the total available RAM on the system. The default setting for 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>
```

See also

RecBuffer, CPUMonitor

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>
```

See also

MaxCacheUnits, MaxCacheSize, MaxUnitSize, FreeRatio, GlobalRatio, MaxAge, UpdateInterval

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>
```

See also

MaxCacheUnits, MaxCacheSize, MaxUnitSize, GlobalRatio, MaxAge, UpdateInterval, FreeMemRatio

GCInterval

Specifies in minutes how often to remove idle handles. The default is 60 minutes.

Example

```
<GCInterval>60</GCInterval>
```

See also

MaxSize (HandleCache), IdleTime, TrimSize

GID

Located in the `Process (Server)` and `Process (AdminServer)` containers.

Specifies the group ID of the process. This element is applicable to Flash Media Server running on Linux systems only.

Example

```
<GID>${SERVER.PROCESS_GID}</GID>
```

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>
```

See also

`MaxCacheUnits`, `MaxCacheSize`, `MaxUnitSize`, `FreeRatio`, `MaxAge`, `UpdateInterval`, `FreeMemRatio`

HandleCache

Container element.

Contains elements that configure the size and features of the handle cache.

Contained elements

`MaxSize (HandleCache)`, `IdleTime`, `TrimSize`, `GCInterval`

HeapSize

Located in the `GlobalQueue`, `EdgeCore`, and `Services` containers.

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

FreeMemRatio

HostPort

Specifies the IP address and port number that the Flash 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 the Flash Media Server. When administrators connect to the server with the Administration Console, they are connecting to the Flash Media Administration Server, which in turn connects to Flash Media Server.

Example

```
<HostPort>ip:port</HostPort>
```

See also

RTMP (AdminServer), SocketGC, Process (AdminServer), AdminElem

HTTP

Container element.

The elements nested within the HTTP container configure the HTTP connector, which is used by remote Flash Player sites to access Flash Media Server.

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.
>0	Associates the default value with the number (N) of processors.
-1	Allocates 1xN threads.
-2	Allocates 2xN threads.

Contained elements

MinConnectionThreads, MaxConnectionThreads, MaxConnectionQueueSize, HandleCache

IdleTime

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. Default wait time is 10 minutes.

Example

```
<IdleTime>10</IdleTime
```

See also

MaxSize (HandleCache), TrimSize, GCInterval

IPCQueues

Container element.

The elements nested within the `IPCQueues` container configure the IPC queues. Flash 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 Flash Media Server to increase performance of large allocations.

Contained elements

MaxCacheUnits, MaxCacheSize, MaxUnitSize, FreeRatio, GlobalRatio, MaxAge, UpdateInterval, FreeMem-Ratio

LicenseInfo

Specifies key information about server licensing, including how many connections are allowed.

Note: *Serial numbers that are added manually (that is, added by editing those files directly) to either fms.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>
```

See also

Mask, ActiveProfile

LicenseInfoEx

Contains license keys added using the Administration Console. For more information about license keys, see the `LicenseInfo` element.

See also

LicenseInfo

LocalHost

Specifies the Flash Media Server IP loopback address.

Flash Media 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.

Example

```
<LocalHost>localhost</LocalHost>
```

See also

Logging, PublicIP, SWFVerification

Logging

Container element.

The elements nested within the Logging container perform the overall logging configuration. You set the configuration properties of the individual log files in the Vhost.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 stream's 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, 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 Flash Media Server running on Linux systems only. This element controls who has read/write access to shared object and stream files in the server. All Flash 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 Flash Media Server object files are created with permission `0666 & ~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 & ~022 = 0644 = rw-r--r--`.

Example

```
<Mask>017</Mask>
```

See also

Process (AdminServer), LicenseInfo, ActiveProfile

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>
```

See also

`MaxCacheUnits`, `MaxCacheSize`, `MaxUnitSize`, `FreeRatio`, `GlobalRatio`, `UpdateInterval`, `FreeMemRatio`

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>
```

See also

`MaxCacheUnits`, `MaxUnitSize`, `FreeRatio`, `GlobalRatio`, `MaxAge`, `UpdateInterval`, `FreeMemRatio`

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>
```

See also

`MaxCacheSize`, `MaxUnitSize`, `FreeRatio`, `GlobalRatio`, `MaxAge`, `UpdateInterval`, `FreeMemRatio`

MaxConnectionQueueSize

Located in the `HTTP` and `RTMP (Connector)` containers.

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>
```

See also

MinConnectionThreads, MaxConnectionThreads, HandleCache

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. The default number is 5. To use the default, specify 0.

The number of threads for HTTP requests is limited to 10 by default. If the server is taking a long time to process connections, however, raise the value of MaxConnectionThreads to 20.

Example

```
<MaxConnectionThreads>20</MaxConnectionThreads>
```

See also

MinConnectionThreads, MaxConnectionQueueSize, HandleCache

MaxIdleTime

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 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.

Example

```
<MaxIdleTime>600</MaxIdleTime>
```

See also

CheckInterval

MaxIOThreads

Located in the ACCP, Admin, Core, ECCP, Edge, and RTMP (Connector) 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.

Flash 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>
```

See also

MinIOThreads, NumCRThreads, MinConnectionThreads, MaxConnectionThreads

MaxKeyframeCacheSize

Specifies the maximum number of keyframes in the cache for each FLV 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 FLV files, you may want to lower this number.

Example

```
<MaxKeyframeCacheSize>0</MaxKeyframeCacheSize>
```

See also

FileCheckInterval, MaxSize (FLVCache), UpdateAccessTimeInterval

MaxNumberOfMessages

Specifies the maximum number of messages that the buffer holds before the messages are committed to file. The default value is 200 and the minimum value is 0.

Example

```
<MaxNumberOfMessages>200</MaxNumberOFMessages>
```

See also

MaxSize (RecBuffer), MaxTimestampSkew

MaxQueueSize

Located in the GlobalQueue, EdgeCore, Services containers.

Specifies the maximum number of pending IPC messages that can be 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 message. <MaxQueueSize> can be used to limit the number of messages left in the pending queue so that shared memory is saved if the process never starts. The value is specified in kilobytes. The default size is 100 KB.

Example

```
<MaxQueueSize>10</MaxQueueSize>
```

See also

HeapSize

MaxSize (FLVCache)

Specifies the maximum size of the FLV 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.

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 FLV memory cache, increase the value of MaxSize. To decrease the amount of memory the server process uses, decrease the value of MaxSize.

Example

```
<MaxSize>500</MaxSize>
```

See also

FileCheckInterval, UpdateAccessTimeInterval, MaxKeyframeCacheSize

MaxSize (HandleCache)

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>
```

See also

IdleTime, TrimSize, GCInterval

MaxSize (RecBuffer)

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.

Example

```
<MaxSize>5120</MaxSize>
```

See also

MaxNumberOfMessages, MaxTimestampSkew

MaxTimestampSkew

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>
```

See also

MaxNumberOfMessages, MaxSize (RecBuffer)

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.

Flash Media Server has several memory pools (distinct from the FLV 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 `MessageCache` tag, the server doesn't cache any messages greater than `MaxUnitSize`.

Example

```
<MessageCache>
    <MaxUnitSize>16</MaxUnitSize>
</MessageCache>
```

See also

MaxCacheUnits, MaxCacheSize, FreeRatio, GlobalRatio, MaxAge, UpdateInterval, FreeMemRatio

MessageCache

Container element.

The elements nested within the `MessageCache` container control how the message cache holds onto messages used by the system running Flash 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 Flash 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>
```

See also

MaxConnectionThreads

MinGoodVersion

Specifies the minimum accepted version of SWFVerification allowed by the server. The default value of 0 accepts current and all future versions.

Example

```
<MinGoodVersion>0</MinGoodVersion>
```

See also

SWFFolder

MinIOThreads

This element is located in the ACCP, Admin, Core, ECCP, Edge, and RTMP (Connector) containers.

The element specifies the minimum number of threads that can be created for I/O operations.

Flash 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>
```

See also

MaxIOThreads, NumCRThreads, MinConnectionThreads, MaxConnectionThreads

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.

Example

```
<MsgPoolGC>60</MsgPoolGC>
```

See also

ThreadPoolGC, ApplicationGC

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. Flash Media Server automatically detects operating system configuration; this element can force Flash Media Server to use IPv4 even if IPv6 is available.

Example

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

NumCRThreads

Specifies the number of completion routine threads in Windows 32-bit systems for edge server I/O processing.

Example

```
<NumCRThreads>0</NumCRThreads>
```

See also

MinIOThreads, MaxIOThreads, MinConnectionThreads, MaxConnectionThreads

Order

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>
```

See also

Allow, Deny

Process (AdminServer)

Container element.

The elements nested within the `Process` container configure UID and GID for the Administration Server. In UNIX, all the Administration Server processes switch from root to the UID and GID specified in this section for security reasons. If no UID and GID are specified, the server runs as root.

Contained elements

`UID`, `GID`

Process (Server)

Container element.

The elements nested within the `Process` container contain the ID elements for all server processes. These elements are applicable to Flash Media Server running on Linux systems only. In UNIX, all the server processes switch from root to the UID and GID specified in this section for security reasons. If no UID and GID are specified, the server runs as root.

Contained elements

`UID`, `GID`

Protocol

Container element.

Flash 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 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)` containers

PublicIP

Specifies that if the system has more than two network ports, a public IP address should be created for the system.

See also

`Logging`, `LocalHost`, `SWFVerification`

RecBuffer

Container element.

Contains elements that configure the buffer for FLV recording.

Contained elements

MaxNumberOfMessages, MaxSize (RecBuffer), MaxTimestampSkew

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

FLVCache, RecBuffer, CPUMonitor, ThreadPoolGC, MsgPoolGC, ApplicationGC, FLVCacheSize, SocketGC, SSLSessionCacheGC, Connector, Protocol, IPCQueues, MessageCache, SmallMemPool, LargeMemPool, SegmentsPool, Master

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 different versions of RTMP, which are applied while connecting to the Administration Server. (RTMP is the protocol used for communication between Flash Player and Flash Media Server.)

Contained element

RTMPE

RTMP (Connector)

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 Flash 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.
<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 Flash Media Server.

Contained elements

Edge, Core, Admin

RTMPE

Specifies if Encrypted Real-Time Messaging Protocol (RTMPE) can be used to connect to the Administration Server. (RTMPE also covers RTMPTE.) The default is set to `true`. Setting this element to `false` prohibits RTMPE and RTMPTE from being used.

Example

```
<RTMPE enabled="true"></RTMPE>
```

See also

AdminServer

Scope

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.

Example

```
<Scope>server</Scope>
```

See also

Enable

SegmentsPool

Container element.

The elements in this section configure how the segments pool caches segments of video files within Flash 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 next within the `Server` element contains the elements that configure the server.

Contained elements

NetworkingIPv6, SSL, Process (Server), Mask, LicenseInfo, ActiveProfile, AdminServer, AutoCloseIdleClients, ResourceLimits, Logging, LocalHost, PublicIP, and SWFVerification

ServerDomain

Specifies the host name (with the domain) of the server computer.

You set this element in the referrer header element when a connection is established with a remote server using NetConnection. Set this element to the server's domain name so that it can pass the domain name to any application servers to which it connects. 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>
```

See also

Server

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 within Flash Media 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>
```

See also

RTMP (AdminServer), HostPort, Process (AdminServer)

SocketOverflowBuckets

Located in the Edge, Core, Admin, ECCP, and 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>
```

See also

MinIOThreads, MaxIOThreads, SocketTableSize

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.

See also

SocketSndBuf

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.

See also

SocketRcvBuff

SocketTableSize

Located in the Edge, Core, Admin, ECCP, and ACCP containers.

Specifies the size of the direct-access socket table for quick lookup. The default size is 200. Use -1 for the default value.

Example

```
<Admin>
    <SocketTableSize>-1</SocketTableSize>
</Admin>
```

See also

MinIOThreads, MaxIOThreads, SocketOverflowBuckets

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

Specifies the name of a file that contains one or more CA (Certificate Authority) digital certificates in PEM (Privacy Enhanced Mail) encryption format.

See also

SSLVerifyCertificate, SSLCACertificatePath, SSLVerifyDepth, SSLCipherSuite

SSLCACertificatePath

Specifies the name of a directory containing CA certificates. Each file in the directory must contain only a single CA certificate, and the files must be named by the subject name's hash, and "0" as an 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 `FMSMaster - console - initialize` from the command line.

See also

SSLVerifyCertificate, SSLCACertificateFile, SSLVerifyDepth, SSLCipherSuite

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 the `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

```
<SSLCipherSuite>ALL:!ADH:!EDH</SSLCipherSuite>
```

This cipher string instructs the server to accept all ciphers except those using anonymous or ephemeral Diffie-Hellman key exchange.

```
<SSLCipherSuite>RSA:!NULL!EXP</SSLCipherSuite>
<SSLCipherSuite>RSA:LOW:MEDIUM:HIGH</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>ALL:+HIGH:+MEDIUM:+LOW:+EXP:+NULL</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:!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:+SSLv2</SSLCipherSuite>
```

This string instructs the server to accept all ciphers, but order them so that SSLv2 ciphers come after SSLv3 ciphers.

Here 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

See also

`SSLVerifyCertificate`, `SSLCACertificatePath`, `SSLCACertificateFile`, `SSLVerifyDepth`,

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

Specifies how often to flush expired sessions from the server-side session cache.

Example

```
<SSLRandomSeed>ALL:!ADH:!EDH</SSLRandomSeed>
```

See also

SSLSessionCacheGC, SSLClientCtx

SSLSessionCacheGC

Specifies how often to check for and remove expired sessions from the server-side session cache.

Example

```
<SSLSessionCacheGC>5</SSLSessionCacheGC>
```

See also

SSLRandomSeed, SSLClientCtx

SSLVerifyCertificate

Specifies if 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>
```

See also

SSLCACertificatePath, SSLCACertificateFile, SSLVerifyDepth, SSLCipherSuite

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>
```

See also

SSLVerifyCertificate, SSLCACertificatePath, SSLCACertificateFile, SSLCipherSuite

SWFFolder

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>
```

See also

MinGoodVersion

SWFVerification

Container element.

Contains elements that configure how SWF files connecting to an application are verified.

Contained elements

SWFFolder, MinGoodVersion

TerminatingCharacters

Specifies the final characters of each log entry in log files. The default is `CRLF` (carriage return and line feed).

Example

```
<TerminatingCharacters>CRLF</TerminatingCharacters>
```

See also

Time

ThreadPoolGC

Specifies in minutes how often Flash 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>
```

See also

MsgPoolGC, ApplicationGC SocketGC, SSLSessionCacheGC

Time

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>
```

See also

TerminatingCharacters, Access, TrimSize

TrimSize

Specifies a percentage of cached handles to remove. Can be specified as a number between 0 and 1, representing 0% to 100%. Default is 0.2 (20%).

Example

```
<TrimSize>0.2</TrimSize>
```

See also

MaxSize (HandleCache), IdleTime, GCInterval

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 (Cache)

UID

Located in the Process (Server) and Process (AdminServer) containers.

This element contains the server process user ID.

If no UID or group ID (GID) is specified, the server or Administration Server runs as root. This element is applicable to Flash Media Server running on Linux systems only.

Example

```
<UID>${SERVER.PROCESS_UID}</UID>
```

See also

GID

UpdateAccessTimeInterval

Specifies how often, in seconds, to modify the access time of the video cache file in the edge server when the video is actively used by the server. The default value is 1200 seconds (20 minutes). Set the value to -1 to disable the server from changing the access.

Example

```
<UpdateAccessTimeInterval>1200</UpdateAccessTimeInterval>
```

See also

FileCheckInterval, MaxSize (FLVCache), MaxKeyframeCacheSize

UpdateInterval

Located in the MessageCache, SmallMemPool, LargeMemPool, and SegmentsPool containers.

Specifies how often, per reused messages, thread statistics are collected. The default count is every 1024 messages.

Example

```
<UpdateInterval>1024</UpdateInterval>
```

See also

MaxCacheUnits, MaxCacheSize, MaxUnitSize, FreeRatio, GlobalRatio, MaxAge, FreeMemRatio

UpdateInterval (Cache)

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

Users.xml file

Users.xml is located at the root level of the `conf` directory. It contains the elements and information used to identify Flash Media Server administrators and their access permissions and to configure Server Management API calls to Flash Media Administration Server. You edit the Users.xml file to add or remove Flash Media Server administrators, or change their administrative permissions.

To see the element structure and default values in Users.xml, see the Users.xml file installed with Flash Media Server in the *RootInstall*/conf/ directory.

Summary of elements

Users.xml element	Description
AdminServer	Container element; contains elements to configure access to the Flash Media Administration Server.
Allow (HTTPCommands)	Lists the Flash Media Administration Server commands that the administrator can access using HTTP.
Allow (User)	Lists the specific hosts from which an administrator can connect to the Flash Media Administration Server.
Deny (HTTPCommands)	Lists the Flash Media Administration Server commands denied access via HTTP.
Deny (User)	Lists the specific hosts from which the administrator cannot connect to the Flash Media Administration Server.
Enable	Enables or disables using HTTP requests to execute administration commands.
HTTPCommands	Container element; contains settings for those administration commands accessed through the HTTP protocol.
Order (HTTPCommands)	Specifies the order of processing for lists of denied and allowed HTTP commands for accessing the Flash Media Administration Server.
Order (User)	Specifies the order in which to evaluate the `Allow` and `Deny` elements.
Password	Specifies the password for virtual host administrators.
Root	Root element; this element is a container for all the other elements.
User	Identifies an administrator of the server.
UserList	Container element; contains information about server administrators.

AdminServer

Container element.

The `HttpCommands` container nested within the `AdminServer` container configures the access level to the Flash Media Administration Server.

The Administration Service is separate from Flash Media Server. When administrators use the Administration Console to connect to Flash Media Server, they are connecting to the Flash Media Administration Server, which in turn connects to the server.

Contained element

`HTTPCommands` container

Allow (HTTPCommands)

Lists the Flash 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)`, and `Order (HTTPCommands)`

Allow (User)

Lists the specific hosts from which an administrator can connect to the Flash 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)

Flash 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 Flash 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)

Flash 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 Flash 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

```
<Deny>foo.yourcompany.com,adobe.com,10.60.1.133,10.60</Deny>
```

This example lists the computers sending connection requests that Flash Media Administration Server will not accept.

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 Flash 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)

Flash 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)

Flash Media Server uses two `Order` elements: one in the `HTTPCommands` container, and the other in the `User` container.

Specifies the sequence in which Flash 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

Specifies the password for vhost administrators.

Passwords cannot be empty strings (`""`). Passwords are usually encrypted. In the following example, the `encrypt` attribute instructs the server to encrypt the contents of the password. When the `encrypt` attribute is set to `true`, the password you see in the file is the encrypted password, and it is interpreted as an encoded string.

Example

```
<Password encryt="true"></password>
```

See also

`Allow (User)`, `Deny (User)`, `Order (User)`

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 Flash 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 it 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 Flash Media Server in the *RootInstall*/conf/_defaultRoot_/_defaultVhost_ directory.

Summary of elements

Vhost.xml element	Description
`AggregateMessages`	Determines if aggregate messages are delivered from the edge cache when the virtual host is configured as edge.
`Alias`	Specifies the assumed name(s) of the virtual host.
`AliasList`	Container element; contains the list of `Alias` elements.

Vhost.xml element	Description
Allow	Specifies the domains that can connect to this virtual host.
AllowOverride	Specifies if overriding edge autodiscovery is allowed.
Anonymous	Determines whether or not this virtual host runs as an anonymous proxy.
AppInstanceGC	Specifies how often to check for and remove unused application instances.
AppsDir	Specifies the Applications directory for this virtual host.
AutoCloseIdleClients	Specifies whether or not to automatically close idle clients.
CacheDir	Specifies the physical location where streams are cached on server.
DNSSuffix	Specifies the primary DNS (Domain Name Server) for this virtual host.
EdgeAutoDiscovery	Container element; contains elements that configure edge autodiscovery.
Enabled	Specifies if edge autodiscovery is enabled.
LocalAddress	Specifies a local IP Address for an edge's outgoing connection.
MaxAggMsgSize	Specifies the size of aggregate messages returned from the edge cache.
MaxAppInstances	Specifies the maximum number of application instances that can be loaded onto the virtual host.
MaxConnections	Specifies the maximum number of clients that can connect to this virtual host.
MaxEdgeConnections	Specifies the maximum number of connections that can connect to this virtual host remotely.
MaxIdleTime	Specifies the maximum idle time allowed, in seconds, before a client is disconnected.
MaxSharedObjects	Specifies the maximum number of shared objects that can be created.
MaxStreams	Specifies the maximum number of streams that can be created.
Mode	Configures this virtual host to run applications locally or remotely.
Proxy	Container element; the elements in this section specify the settings for the virtual host to act as an edge server and forward connection requests from applications to another Flash Media Server, and also behave locally as a remote server.
ResourceLimits	Container element; the elements in this section specify the maximum resource limits for this virtual host.
RouteEntry	Maps the proxy's host:port pair to a different host:port pair.
RouteTable	Container element; the elements in this section specify the proxy's routing information.
SSL	Container element; the elements in this section configure this virtual host for secure communications.
Streams	Specifies the virtual directory for recorded streams.
VirtualDirectory	Container element; configures the directory mappings for resources such as recorded streams.
VirtualHost	Root element; contains all other elements for the Vhost.xml file.
VirtualKeys	Sets the virtual key mappings for connecting players.
WaitTime	Specifies length to wait for edge autodiscovery.

AggregateMessages

Determines if aggregate messages are delivered from the edge cache when the virtual host is configured as an edge server. Default 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 element

`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 if 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>falsetrue</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 directory with the application name.

- In Windows, the default `AppsDir` location is C:\Program Files\Adobe\Flash Media Server 3\applications.
- In Linux, the default location is /opt/adobe/fms/applications.

Note: *If you use this tag to map to a network drive, see Mapping directories to network drives for additional information.*

Example 1

```
<AppsDir>C:\MyApps;D:\NewApps</AppsDir>
```

You can also 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 (`fms_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\fmsapps</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 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 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

This element enables or disables writing recorded streams to disk. Set this element on an edge server to control the caching behavior.

The contents of the cache are volatile. 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 applications. Caching static content can reduce the overall load placed on the origin server.

The default location is the cache folder in the server installation directory. The default value of the `enabled` attribute is `false`.

Example

To save the contents of the cache, set the `enabled` attribute to `true` and specify a directory on the disk where the files will be written.

```
<CacheDir enabled="true">c:\mycache</CacheDir>
```

See also

`Mode`, `Anonymous`, `LocalAddress`, `RouteTable`

DNSSuffix

Specifies the primary DNS suffix for this virtual host.

If a reverse DNS lookup 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 if 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. Default is `false`.

Example

```
<Enabled>false</Enabled>
```

See also

AllowOverride, WaitTime

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

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 600 seconds (10 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>600</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

MaxStreams

Specifies the maximum number of streams that can be created. The default number of streams is 250,000.

Example

```
<MaxStreams>250000</MaxStreams>
```

See also

MaxConnections, MaxAppInstances, MaxSharedObjects, AppInstanceGC

Mode

The `Mode` element configures whether Flash Media Server runs locally as an origin server or remotely 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`, Flash 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

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 remote 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.

For more information on this section of the Server.xml file, see SSL.

Contained elements

`Mode`, `Anonymous`, `CacheDir`, `LocalAddress`, `RouteTable`, `EdgeAutoDiscovery`, `SSL`, `AggregateMessages`

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, Flash 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

```
<Proxy>
    <RouteTable protocol="">
    <RouteEntry>foo:1935;bar:80</RouteEntry>
    </RouteTable>
</Proxy>
```

This example shows how you can configure the edge to route all connections to the host `foo` to the host `bar`.

```
<RouteEntry>*:*;foo:1935</RouteEntry>
```

Flash Media Server allows the 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>*:*;*:1936</RouteEntry>
```

The example instructs Flash Media 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>*:*;*:80</RouteEntry>
```

The example instructs Flash Media 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>foo:80;null</RouteEntry>
```

The example instructs Flash Media Server to route a host:port combination to `null`. Its effect is to reject all connections destined for `foo:80`.

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 nonsecure connection, 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 nonsecure, the outgoing connection will be nonsecure.
- Specifying `rtmp` instructs the edge to use a nonsecure 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 nonsecure.

You can override the security status for a connection mapping by specifying a `protocol` attribute in a `RouteEntry` element. By default, Flash Media Server applies the protocol configured in the `RouteTable` list unless the mapping for a particular `RouteEntry` element overrides it.

Contained element

`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 Flash 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, Flash 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, Flash 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, Flash Media Server uses the default values specified in the SSL section of Server.xml to configure the SSL connection to upstream servers.

Note: *When Flash 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.

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.

Examples

```
<Streams>foo;c:\data</Streams>
```

This example maps all streams whose names begin with foo/ to the physical directory c:\data. The stream named foo/bar would map to the physical file c:\data\bar.flv.

If there is a stream named foo/bar/x, then Flash Media Server first tries to find a virtual directory mapping for foo/bar. If there is no virtual directory for foo/bar, Flash Media Server then 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.

```
<Streams>common;C:\FlashMediaServer\myApplications\shared/resources\</Streams>
```

If the virtual directory you specify does not end with a backslash, one is added by the server.

Any application that refers to a stream whose path begins with common/ will access the item in C:\FlashMediaServer\myApplications\shared\resources, regardless of the application's own directory structure. If the application VideoConference refers to an item common/video/recorded/June5 and the application Collaboration refers to common/videorecorded/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, Flash Media Server serves the resource from the physical directory. For detailed information on mapping virtual directories, see Mapping virtual directories to physical directories.

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 Flash Media Server has the right permissions to access the other computer. For more information, see Mapping directories to network drives.*

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 element

`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. For more information, see `VirtualDirectory`.

Example

When Flash Player connects to Flash Media Server, it sends the server a string containing its platform and version information. You can add `Key` elements that map the Flash Player information to keys. The keys can be any alpha-numeric 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.

See also

VirtualDirectory

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 9: Diagnostic Log Messages

Message IDs in diagnostic logs

This section contains 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), which record information about Flash Media 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 and Application logs.

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$S: %2$S.
1006	Failed to stop %1$S listeners for adaptor %2$S.
1007	Failed to create thread (%1$S).
1008	Asynchronous I/O operation failed (%1$S: %2$S).
1009	Service Control Manager failed (%1$S: %2$S).
1010	Service Control Manager failed (%1$S: %2$S).
1011	Server starting...
1012	Server stopped %1$S.
1013	Failed to create listener for adaptor %1$S, IP %2$S, port %3$S: %4$S.
1014	Command name not found in the message.
1015	Method not found (%1$S).
1016	Failed to execute method (%1$S).
1017	Failed to stop virtual host (%1$S).
1018	`The call` method failed, invalid parameters: `call(methodName[, resultObj, p1, pn]).`
1019	Dropping application (%1$S) message. Clients not allowed to broadcast message.
1020	Response object not found (%1$S).
1021	Missing unlock for shared object %1$S, lock count %2$S.
1022	Nested lock for shared object %1$S, lock count %2$S.
1023	Unlock called without matching lock for shared object %1$S.
1024	Invalid application; rejecting message (%1$S).

Message ID	Description
1025	Ignoring message from client during authentication.
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

Message ID	Description
1058	Invalid application name (%1$S).
1059	Invalid user ID (%1$S).
1060	NetConnection.Admin.CommandFailed
1061	Invalid parameters to %1$S method.
1062	Failed to unload application %1$S.
1063	Failed to load application %1$S.
1064	%1$S applications unloaded.
1065	Admin user requires valid user name and password.
1066	Invalid virtual host alias : %1$S.
1067	Error registering class: name mismatch (%1$S, %2$S).
1068	Connection rejected: maximum user limit reached for application instance %1$S.
1069	(%2$S, %3$S) : Failed to load application instance %1$S.
1070	(%2$S, %3$S) : Connection rejected to application instance %1$S. Client already connected to an application.
1071	Illegal access property (%1$S).
1072	%1$S is now published.
1073	%1$S is now unpublished.
1074	Stopped recording %1$S.
1075	Stream %1$S has been idling for %2$S second(s).
1076	Playing and resetting %1$S.
1077	Pausing %1$S.
1078	Unpausing %1$S.
1079	Started playing %1$S.
1080	Stopped playing %1$S.
1081	Recording %1$S.
1082	Failed to record %1$S.
1083	New NetStream created (stream ID: %1$S).
1084	NetStream deleted (stream ID: %1$S).
1085	Publishing %1$S.
1086	Failed to publish %1$S.
1087	Failed to restart virtual host (%1$S).
1088	Connection to Flash Media Server has been disconnected.
1089	Failed to play (stream ID: %1$S).
1090	Failed to play %1$S (stream ID: %2$S).

Message ID	Description
1091	Play stop failed, stream ID: %1$S.
1092	Audio receiving enabled (stream ID: %1$S).
1093	Audio receiving disabled (stream ID: %1$S).
1094	Failed to enable audio receiving (stream ID: %1$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$. Flash Media Server is already running or other processes are using the same ports.
1120	Configuration file not found: %1$S
1121	Invalid configuration file: %1$S
1122	Server aborted.

Message ID	Description
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).
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.

Message ID	Description
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.
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$S.
1166	Attempt to reconnect to Flash Media Server.
1167	Failed to remove application: %1$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$S) is now below the maximum threshold.
1173	*Generic message code.*
1174	Listener started (%1$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 exceeds 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

Message ID	Description
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]
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.

Message ID	Description
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$S) has exceeded 90 usage. Consider increasing the heap size to prevent future memory allocation failures.
1228	Failed to create process mutex.
1229	Process (%1$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!

Message ID	Description
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.
1262	OpenProcess(PROCESS_TERMINATE) failed with %1$S.
1263	OpenProcess(PROCESS_QUERY_INFORMATION) failed for pid (%1$S) with %2$S.
1373	SWF verification failed, disconnecting client.
1374	SWF verification timeout, disconnecting client.
1375	SWF verification unsupported by client, disconnecting client.

Index

W