ADOBE® MEDIA SERVER
Technical Overview

Legal notices

Contents

Chapter 1: Introduction

Adobe® Media Server is a real-time media server that delivers adaptive bit rate video on demand (vod) and live video to Adobe® Flash® Player, Adobe® AIR™, Adobe® Flash® Lite™, Apple® iOS, and Mac OS®.

Adobe Media Server also delivers streaming music, video blogging, video messaging, multimedia chat environments, real-time datacasting, and multiuser gaming. Adobe Media Server delivers content Applications and media run consistently across platforms and browsers.

There are four editions of Adobe Media Server 5.0.1:

- Adobe Media Server Standard
- Adobe Media Server Professional
- Adobe Media Server Extended
- Adobe Media Server Starter

Server editions

For a feature comparison of server editions, see www.adobe.com/products/flashmediaserver/compare/.

System requirements

For the most up-to-date system requirements, see www.adobe.com/go/ams_techspecs.

What's new in Adobe Media Server 5.0.3

The following new features are available in Adobe Media Server 5.0.3:

- Protected RTMP onDemand Enhancements
    - **Logging improvement for billing**: The Adobe Media Server access log supports an additional column "`x-sprotection-ver`." This is logged with the application and stream events with value "1" if the stream is protected (PRTMP enabled).
    - **Integration of common key and white-list access with file-plugin interface**: The common key and white-list access in the Protected RTMP on-demand workflow is routed through file adaptor if configured for handling "PRTMP" requests. Following configuration is added to `server.xml`:

```
Root/Server/Plugins/FilePlugin:
    <!-- This section configures PRTMP common key and file         -->
    <!-- whitelist file operations  in present plugin. By         -->
    <!-- default such operations won't be managed by this         -->
    <!-- plugin.                                                 -->
    <Content type="PRTMP">false</Content>
```

- **DRM refresh**: The DRM refresh is needed if there is a change in swf whitelist on runtime or update of common-key. New DRM will be in effect as soon as new clients subscribe in the following scenarios:
  - new whitelist hashes are provided through the auth adaptor or
  - whitelist folder modification time is changed

  This is independent of whether the update interval is configured or not. If only the exiting whitelist file is modified and the swf folder modified time is not updated, the new DRM is in effect from the next update interval onwards.

  The existing streams will play with the old DRM.

  You can update `F_APP_PRTMP_SWF_WHITELIST` or `F_APP_PRTMP_AIR_WHITELIST` at the time of connect. This updates the swf whitelist for whole application.

- **Common-key update**: Common key can be updated in the same location. The Origin Server flushes the cache and reloads the data.
- **Key rotation:** You can rotate the key for protected RTMP live playback. You can periodically change the encryption key by specifying the frequency of content encryption key change. Rotated keys are generated dynamically using the common key and content id.

  Added configuration:

  ```
  <EnableKeyRotation>true</EnableKeyRotation>
  <KeyRotationInterval>500</KeyRotationInterval> // in seconds
  ```

  Added configuration in `Application.xml.`

  ```
  <Application>
  <ProtectedRTMP enabled="true"></ProtectedRTMP>
  <EnableKeyRotation>true</EnableKeyRotation>
  <KeyRotationInterval>500</KeyRotationInterval>
  ...
  </Application>
  ```

What’s new in Adobe Media Server 5.0.1

Adobe Media Server is available in the following configurations:

- Adobe Media Server Starter (formerly Adobe Flash Media Development Server)
- Adobe Media Server Standard (formerly Adobe Flash Media Streaming Server)
- Adobe Media Server Professional (formerly Adobe Flash Media Interactive Server)
- Adobe Media Server Extended (formerly Adobe Flash Media Enterprise Server)
- Adobe Media Gateway is a bundled software available with your license

Support for closed captions

Closed captioning allows the content provider to display text overlayed on the video content thus providing additional information about the video content to the consumers. Closed captioning is primarily used as an accessibilty tool to show a transcription of the audio portion of the video content as it occurs. The term “closed” in closed captioning indicates that the captions are not made part of the video by default. Hence, the captions have to be decoded/extracted by an external tool. For more information on configuring the server to support closed captions, see *Configure closed captioning*.

Support for alternate audio

Adobe Media Server 5.0.1 has support for including multiple language tracks for HTTP video streams, without requiring duplication and repackaging of the video for each audio track. This feature, called as "late" binding of audio tracks allow content providers to easily provide multiple language tracks for a given video asset, at any time before or after the asset's initial packaging. The initial packaging of the video asset can include an audio track, and the publisher can still choose to provide one or more additional audio tracks for the video. Video players can provide support for allowing the viewers to switch between audio tracks either before or during playback. For more information on configuring the server to support alternate audio, see *Configure alternate audio*.

Important changes in Adobe Media Server 5.0.1

Adobe Media Server 5.0.1 has introduced a few important branding and configuration-level changes. See the following changes:

Change	Description
Changes in the `httpd.conf` configuration file	The following list shows the summary of changes: • All the configuration elements starting with the prefix""`FlashAccess`" have been renamed to start with "`HdsDrm`" for HDS and "`JitDrm`" for JIT. • All the configuration elements starting with the prefix "`HLSFAXS`" have been renamed to start with "`HLSDrm`". • New property: `PHLSCommonKeyFile` • New property: `PHLSContentID` • Modified property: `PHLSOutputProtection` • Modified property: `PHLSPlaybackExpiration` • Modified property: `PHLSPlayerBindingEnabled` • Modified property: `PHLSWhitelistFolder` • Renamed property: From "`HLSDrmEmbedMetadata`" to "`HLSEmbedMetadata`" • Protection scheme property values starting with the prefix "`FlashAccessV`" have been renamed to "`AdobeAccessV`". Note that the previous values will also work so the existing content/event specific configurations will work without any refactoring.
Changes in executables	The following list shows the summary of changes on Linux: • fmsmaster has been renamed to amsmaster • fmscore has been renamed to amscore • fmsedge has been renamed to amsedge • fmsadmin has been renamed to amsadmin The following list shows the summary of changes on Windows: • FMSMaster.exe has been renamed to AMSMaster.exe • FMSAdmin.exe has been renamed to AMSAdmin.exe • FMSCore.exe has been renamed to AMSCore.exe • FMSEdge.exe has been renamed to AMSEdge.exe

Change	**Description**
Changes in installer scripts	The following list shows the summary of changes: • fmsmgr has been renamed to amsmgr • installFMS has been renamed to installAMS • uninstallFMS has been renamed to uninstallAMS
Changes in names of the tools	The following list shows the summary of changes: • fmscheck has been renamed to amscheck • fms_adminConsole.swf has been renamed to ams_adminConsole.swf
Changes in names of the services	The following list shows the summary of changes on Linux: • fmsadmin has been renamed to amsadmin • fms has been renamed to ams The following list shows the summary of changes on Windows: • FMSAdmin has been renamed to AMSAdmin • FMS has been renamed to AMS • FMSHttpd has been renamed to AMSHttpd
Changes in the name of configuration files	fms.ini has been renamed to ams.ini
Other changes	The following list shows the summary of changes: • Default installed directory on Linux is now /opt/adobe/ams • Default installed directory on Windows is now "C:\Program Files\Adobe\Adobe Media Server 5" • Default User name is changed from "fms" to "ams" • Registered Services reference location on Linux is changed from "/etc/adobe/fms/services" to "/etc/adobe/ams/services"

Features introduced in the earlier versions

The following sections have information on the changes introduced in the earlier versions of Adobe Media Server.

What's new in Adobe Media Server 5

New content protection options for HLS streaming

There are new content protection and Digital Rights Management (DRM) workflows for streaming to both Flash using HTTP Dynamic Streaming (HDS) and native Apple IOS applications using HTTP Live Streaming (HLS). With this feature Adobe Media Server can dynamically segment, encrypt, and deliver standard MP4 assets using the HLS format with Adobe Access DRM policies on native Apple iOS applications (using the Adobe Access Objective-C library for iOS).

Adobe Media Server 5 supports content protection of HLS video streams using Adobe Access protection without requiring a separate license server.

Just in Time (JIT) content protection

Adobe Media Server 5 enables real time packaging and encryption of your media for HTTP delivery to both Adobe Flash and Apple HLS file formats. This means that your single live source publishing need not be duplicated to serve Adobe Flash and Apple HLS formats, which significantly reduces your storage and stream management requirements. Using dynamic and real time encryption, Adobe Media Server 5 can prepare and encrypt a single live source of F4F recording and deliver the encrypted segments over HTTP to both HDS (Flash) and HLS (Apple iOS).

HLS Offline Segmenter

A new HLS offline segmenter utility converts MP4 video files into HLS and M3U8 files for HTTP streaming to devices that support HLS such as the Apple iPad. When used with Adobe Access, the HLS packager can encrypt the HLS file for playback in native IOS applications using the Adobe Access iOS client SDK.

What’s new in Flash Media Server 4.5.2

HTTP Streaming failover

HTTP Streaming failover lets you control how AMS handles certain streaming problems. In particular, HTTP Streaming failover helps you address liveness and dropout:

- **Liveness** - A packager advertises a stale bootstrap (that is, a stale view of a live stream).
- **Dropout** - A packager has gaps in its bootstrap (that is, gaps in its fragment list).

For more information, see “HTTP Streaming failover” on page 5.

What’s new in Flash Media Server 4.5.1

24/7 Live streaming

In Flash Media Server 4.5.1, timestamps are converted to 64-bit values. Input timestamps are 32-bit values, but the server creates 64-bit timestamps to use internally. This feature allows the server to stream live video continuously while maintaining a 12 hour DVR window.

No server configuration or media player development is required to support 24/7 live streaming.

Protected RTMP

Use protected RTMP (pRTMP) to encrypt and deliver on-demand content to Flash Player and AIR. Protected RTMP replaces RTMPE with a higher level of content protection and is easy to deploy.

See Stream on-demand encrypted media (pRTMP).

***Note:** Protected RTMP isn’t a protocol. It delivers encrypted content over the RTMP protocol.*

Encryption key rotation

Flash Media Server 4.5.1 supports Key Rotation for protected HTTP Dynamic Streaming when used with Adobe Access. You can encrypt content packaged with FMS 4.5.1 using a set of keys. You can periodically change the encryption key and specify how often the content encryption key is to be changed. You can also specify the list of keys for encryption.

Output protection

Flash Media Server 4.5.1 includes four additional policy files, which support Output Protection. Those policies allow different level of playback restriction based on client hardware capabilities.

What's new in Flash Media Server 4.5

Deliver content using Adobe HTTP Dynamic Streaming and Apple HTTP Live Streaming

Deliver adaptive bit rate on-demand and live content over HTTP to Flash Player, AIR, iOS, and Mac OS. The server packages live and on-demand content in real time when a client requests it. HTTP Dynamic Streaming and HTTP Live Streaming both support DVR.

***Note:** Flash Media Server 4.0 supported HTTP Dynamic Streaming, but Flash Media Server 4.5 adds support for just-in-time packaging of on-demand content.*

For more information, see the following:

- Stream live media (HTTP) and Stream on-demand media (HTTP)in *Flash Media Server Developer's Guide*.
- Configure HTTP Dynamic Streaming and HTTP Live Streaming in *Flash Media Server Developer's Guide*.
- Configure ports for HTTP streaming in *Flash Media Server Configuration and Administration Guide*.

Protected HTTP Dynamic Streaming and Protected HTTP Live Streaming

Deliver protected live and on-demand multi-bitrate video to Flash Player, AIR, iOS, and Mac OS without using a DRM License Server. Protected HTTP Dynamic Streaming also supports SWF verification.

For more information, see the following in *Flash Media Server Developer's Guide*:

- Configuring content protection for HDS
- Configuring content protection for HLS

Publish an audio-only stream for HTTP streaming

Apple HTTP Live Streaming requires that one stream in a multi-bitrate set be audio-only to deliver the content over a cellular network.

See Publish an audio-only stream (HLS)

Set-level F4M/M3U8 files

Create manifest files that describe a set of content for multi-bitrate streaming. These files are called *set-level manifest files*. The HTTP packagers generate *stream-level* manifest files in real time when the content is requested.

***Note:** For Adobe HTTP Dynamic Streaming, manifest files are F4M files. For Apple HTTP Live Streaming, the equivalent file is called a "variant playlist" and the filename extension is M3U8. The documentation uses the generic term "manifest file" to refer to both file types.*

Set-level F4M/M3U8 files contain bit rate information about a set of content. Stream-level F4M/M3U8 files contain bootstrap information and DRM metadata. Flash Media Server 4.5 includes a Set-level File Generator tool that creates set-level F4M v2.0 files for HTTP Dynamic Streaming and set-level M3U8 variant playlists for HTTP Live Streaming.

See Set-level manifest files.

Use the File plug-in to manage content for live HTTP streaming

Use the File plug-in to manage content for live HTTP streaming, including asynchronous file IO operations. The live packager application (livepkgr) ingests a live stream and packages it into fragments (F4F files) and additional helper files. The server operations that record these files are routed through the File plug-in.

See Use the File plug-in to manage content for live HTTP streaming

Improved HTTP file system performance

The IO buffer improves the read and write performance of HTTP streaming (for HTTP Dynamic Streaming and HTTP Live Streaming). The IO buffer loads the disk file into an in-memory buffer. It reads and writes to the in-memory buffer instead of making system calls.

See Configure the size of the IO buffer

Disk management enables 24/7 live streaming

Configure the amount of live content the server stores on the disk. By default, the server stores three hours worth of content. This feature allows you to serve live streams 24 hours a day, 7 days a week without filling up a disk. It also allows you to use DVR for long live events.

See Disk management.

Distribute RTMFP peer introductions across servers

Flash Media Server introduces RTMFP clients to each other so the clients can connect to each other directly. This direct connection is called a peer-to-peer connection. Flash Media Server 4.0 can introduce clients to each other only if the clients are connected to a single server. Use Server-Side ActionScript APIs added in Flash Media Server 4.5 to introduces clients to each other even if the clients are connected to separate servers. Distributing introductions across servers allows you to scale peer-assisted networking applications.

See Distribute peer introductions across servers

Ingest a multicast stream

Use Server-Side ActionScript to ingest a multicast RTMFP stream. After the server ingests the multicast stream, write script to do the following:

- Convert the multicast stream to a Stream object.
- Connect to the livepkgr application and package the Stream for delivery using HTTP Dynamic Streaming and HTTP Live Streaming.
- Record the Stream object.
- Deliver the Stream object to clients over RTMP/T/S/E.

See Streaming media files

Administrator password stored securely

The administrator password for the Flash Media Administration Console is stored securely, not as plain text. See Delete administrator accounts and reset passwords.

Configuration file updates

Some configuration files have changed to incorporate new functionality and security enhancements. If you are upgrading from Flash Media Server 4.0 to 4.5, use the documentation to help you migrate from the older version.

Common uses for the server

Capture and broadcast live video

Adobe Flash Media Live Encoder lets you capture audio and video while streaming it live to Adobe Media Server. You can use anything from a webcam to a sophisticated digital video camera to capture the video. Deliver video over HTTP to Flash Player, AIR, iOS, and Mac OS. Deliver video over RTMP to Flash Player and AIR.

To broadcast media to a large number of viewers, use multipoint publishing. This feature streams the live video from a camera to a publishing server, and then to a broadcast server. The broadcast server is often a Content Delivery Network.

***Note:** You can stream live video to and from Adobe Media Server Standard. However, Adobe Media Server Standard does not support multipoint publishing because it requires server-side scripting..*

Multipoint publishing can be used to inject metadata into a live stream. For example, you could create an Internet TV station and publish the stream to a Adobe Media Server Starter server. The Adobe Media Server Starter server would publish the stream to a CDN that pushes the stream to millions of users.

Broadcast recorded video

To deliver recorded, or "on-demand", video Adobe provides a video player called Strobe Media Playback, which supports playback of FLV and MP4/F4V files. The server installs this video player to *rootinstall*/samples/videoPlayer. You can also develop your own video player. Video players run in Flash Player or AIR and stream recorded or live video from Adobe Media Server. You can broadcast media over HTTP to Flash Player, AIR, iOS devices, and Mac OS.

The following are examples of the type of content you can stream:

- Short video clips, such as commercials up to 30 seconds long
- Longer video clips, such as user-generated videos up to 30 min. long
- Recorded television shows or movies up to several hours long
- Client-side or server-side media playlists can play a list of streams in a sequence, whether live streams, recorded streams, or a mix. The playlist can be in a client-side script or, on all server editions except Adobe Media Server Standard, in a server-side script.

Multicast live video within the enterprise

Multicast live video to Flash Player and AIR clients using peer-assisted networking, IP-level multicast, or both. When both technologies are used together, the technique is called "multicast fusion".

When video is broadcast, the server sends a stream to every client. When video is multicast, a multicast-enabled router sends the video to clients, or clients send the video to each other.

***Note:** You can also multicast on-demand video, but the use-case isn't as common.*

Broadcast video with advertising

A streaming video application can insert advertising at various points, such as a short commercial that plays before a recorded television show or live video. The advertisement is often streamed from one server and the content is streamed from another server or from a Content Delivery Network. A video-with-advertising application typically connects to the ad server, streams the ad, and then closes the connection to the ad server. It then connects to the content server, streams the content, and closes that connection, repeating this sequence each time video is streamed.

Integrate video with interactive applications

An Adobe Media Server application can engage the user through video sharing, online chat, web conferencing, and other community-building features. Users can send audio and video as well as text messages to the server, and the server streams the data to all connected users. The server can also record media for playback at a later time, such as in a video messaging application.

Chapter 2: Architecture

Adobe Media Server architecture

Client-server architecture

Adobe Media Server applications have a client-server architecture. The client code is written in ActionScript (1, 2, or 3) or Objective-C and runs on all supported platforms. Adobe Media Server can also stream media to Apple HTTP Live Streaming clients such as iOS devices.

The server code is written in Server-Side ActionScript, which is like ActionScript 1.0.

Real-time media server

Adobe Media Server Standard provides four streaming services: live, vod (video on demand), livepkgr (HTTP streaming), and multicast (RTMFP). Streaming services are prebuilt server-side applications. Each streaming service offers prebuilt sample clients as well as a client SDK that developers can use to write their own clients.

For tutorials that get you started streaming quickly, see Getting started streaming media.

Real-time collaboration application server

To create real-time collaboration application, use RTMP and the classic client-server model, or use RTMFP to create peer-assisted networking applications. Flash Player 10.1 and AIR 2 support Real-Time Media Flow Protocol (RTMFP)and RTMFP groups. RTMFP is built on User Datagram Protocol (UDP). RTMP is built on Transmission Control Protocol (TCP). UDP provides lower latency than TCP. It also enables end-to-end peering—that is, direct data transmission between two clients. Clients make an initial connection to Adobe Media Server. The server acts as an introducer to connect to the client to a group. Once connected to a group, members of the group exchange data among themselves without passing it back to the server.

The server and the client can communicate over a persistent connection using Real-Time Messaging Protocol (RTMP). RTMP is a reliable TCP/IP protocol for streaming and data services. In a typical scenario, a web server delivers the client over HTTP. The client creates a socket connection to Adobe Media Server over RTMP. The connection allows data to stream between client and server in real time. Adobe Media Server installs with Apache web server by default. You can serve HTTP content from this web server. Alternatively, you can choose to exclude Apache from the Adobe Media Server installation and serve SWF and HTML content from any external web server.

Adobe Media Server Extended, Adobe Media Server Professional, and Adobe Media Server Starter include the same streaming services as Adobe Media Server Standard. In addition, they provide an SDK that lets developers write both the client-side and the server-side components of media applications to create interactive, two-way applications. These server editions also offer a plug-in SDK. This SDK lets developers write C++ plug-ins to extend the core functionality of the server.

Note: *Adobe Media Server Standard supports the Access plug-in only.*

Built-in Apache HTTP Server

All versions of Flash Media Server 3.5 and later include of Apache 2.2 HTTP Server. If you install and enable Apache, you can deliver client SWF files, container HTML files, and all media assets from the same server.

Use Adobe Media Server with Apache 2.2 HTTP Server to deliver content using Adobe HTTP Dynamic Streaming and Apple HTTP Live Streaming.

By default, Adobe Media Server proxies HTTP requests from port 80 to port 8134, as shown in the following diagram:

For more information, see Configure ports for HTTP streaming.

You can configure proxying in the ams.ini file. For more information, see Configuring Apache HTTP Server.

Hosting multiple applications

Adobe Media Server Extended, Adobe Media Server Professional, and Adobe Media Server Starter can host an unlimited number of applications. Adobe Media Server Standard can host an unlimited number of instances of the live and vod services.

***Note:** Adobe Media Server Standard is restricted to running services provided by Adobe.*

For example, Adobe Media Server Professional could host a web conferencing application, a video blogging application, a video chat application, and a multiplayer game. The server can also host the live and vod services. You can create multiple instances of each of these applications. Instances allow groups of people access to the same application without having the groups interact with each other. For example, you could create a video chat application with rooms for different topics.

To create an application, create a folder on the server with the name of the application. To connect to the server, write code in the client that calls the `NetConnection.connect()` method and passes it the name of the application.

The server has a root applications folder located at *RootInstall*/applications, by default. To create an application, create a subfolder in the root applications folder. For example, *RootInstall*/applications/mySampleApp creates an application called “mySampleApp”. To connect to mySampleApp, call `myNetConnection.connect("rtmp://`*`serverName`*`/mySampleApp")` from the client.

To create instances of an application, create subfolders within the folder of the application. For example, *RootInstall*/applications/mySampleApp/instance1 creates an instance of the mySampleApp application. To connect to this instance, call `myNetConnection.connect("rtmp://`*`serverName`*`.mySampleApp/instance1")` from the client.

The server administrator can change the location of the root applications folder. The server administrator can also divide the server into multiple adaptors and virtual hosts, and each virtual host can have its own applications folder.

Data model

Adobe Media Server applications use a simple, yet powerful, distributed data model based on *shared objects*. Both client-side ActionScript and Server-Side ActionScript have a SharedObject class that lets developers share data between clients connected to a server.

There are two types of shared objects: *local* and *remote*. Local shared objects are stored on the client computer and remote shared objects are stored on the server. Both local and remote shared objects can be either temporary or persistent.

Local shared objects are like cookies: they save data to a user’s computer for offline access, or for saving preferences. Local shared objects are a feature of Flash Player and do not require Adobe Media Server.

Remote shared objects are managed and stored by the server. Developers can use remote shared objects for messaging, data synchronization, and storing data. Clients connect to a remote shared object and receive updates whenever a change is made to that shared object. Messages can be sent to all clients connected to a remote shared object.

***Note:** Remote shared objects are not supported by Adobe Media Server Standard.*

For more information, see the Server-Side ActionScript SharedObject class, the client-side ActionScript SharedObject class, and Developing social applications.

Invoking remote methods

Adobe Media Server Extended, Adobe Media Server Professional, and Adobe Media Server Starter support two-way, asynchronous, remote method invocation. Clients can invoke methods defined on the server, and the server can invoke methods on clients connected to the server. In client-side script, call the `NetConnection.call()` method to invoke a method defined on a server-side Client object. In a server-side script, call the `Client.call()` method to invoke a method defined on the client-side NetConnection object. You can also call `NetConnection.call()` in a server-side script to invoke a method on a remote server.

For more information, see the client-side `NetConnection.call()` method, and the server-side `Client.call()`, `Client.remoteMethod()`, and `NetConnection.call()` methods.

Connecting to external sources

Note: *Adobe Media Server Standard does not support this feature.*

Adobe Media Server can interact with external data sources, such as web services and relational databases, or with other Adobe Media Server applications. For example, write Server-Side ActionScript to connect to a web service or to a ColdFusion® application to retrieve a list of names and phone numbers. Place the results of the query in a shared object.

For more information, see Server-Side ActionScript Language Reference.

AMF (Action Message Format) is binary file format representing a serialized ActionScript object. Use to send ActionScript objects over the Internet. There are two versions of AMF: AMF 0 (ActionScript 1.0 and 2.0) and AMF 3 (ActionScript 3.0). The following table shows which server versions and player versions added support for these two AMF versions:

AMF version	Adobe Media Server version	Flash Player version
AMF 0	1	6 ; Flash Lite 3
AMF 3	3	8

Note: *Flash Player, AIR, and Flash Lite clients can use any supported client-side ActionScript API to interact with external sources as well; Adobe Media Server doesn't limit this functionality. For more information, see ActionScript 3.0 Reference for the Adobe Flash Platform.*

Streaming media

Stream media

Media (which can contain audio, video, and data) is sent between a client and Adobe Media Server in real time and displayed as it arrives. This type of data transmission is called *streaming*. The media streamed between a client and Adobe Media Server is called a *stream*. Streams use a publish-and-subscribe model; *subscribe* means play. Either a client or a server can publish a stream; only a client can play a stream. Media can stream over RTMP or HTTP.

For example, a producer could use Adobe Flash Media Live Encoder to capture and encode live audio and video from a keynote speech and publish it to the server. Users could view the speech in a Flash Player or AIR client that subscribes to the stream.

In another scenario, a user on a social media website could publish a live stream in a video chat application; in this case, Flash Player would capture the video from the user's webcam. Other users on the social media site could view the stream live, or, if the developer added this functionality, play the recorded stream at a later time.

Note: *All editions of the server except Adobe Media Server Standard can record live streams for playback at a later time.*

A server can publish a stream to clients or to other servers. For example, you might want to pull XML data into a server-side script to create a playlist and publish it as a stream to clients. A server would publish a stream to another server to scale live broadcasting applications to support more clients.

Multicast media

Multicasting is distributing audio and video data among members of an RTMFP group. The server doesn't send data to each client—the data is distributed among peers. Multicasting allows few publishers to send a large amount of data. To multicast media, use the GroupSpecifier class to construct an opaque groupspec string that describes an RTMFP group. Pass the groupspec to the NetStream constructor. Call `NetStream.publish()` or `NetStream.play()` to multicast the data. You can use the client-side NetStream class and the server-side NetStream class to multicast data. See Multicasting.

Supported protocols

RTMP (Real-Time Messaging Protocol)

All server editions communicate with Flash Player and AIR over Real-Time Messaging Protocol. RTMP is optimized to deliver high-impact streams in real time. An RTMP connection can multiplex any number of streams. Each stream contains synchronized audio, video, and data channels. Remote method invocation and shared object messages are carried in a data-only stream.

Adobe Media Server supports the following types of RTMP connections:

RTMP This is the standard, unencrypted Real-Time Messaging Protocol. The default port is 1935; if a port is not specified, the client attempts to connect to ports in the following order: 1935, 443, 80 (RTMP), 80 (RTMPT).

RTMPT This protocol is RTMP tunneled over HTTP; the RTMP data is encapsulated as valid HTTP. The default port is 80.

RTMPS This protocol is RTMP over SSL. SSL is a protocol for enabling secure communications over TCP/IP. (Adobe Media Server provides native support for both incoming and outgoing SSL connections.) The default port is 443.

RTMPE This protocol is an encrypted version of RTMP. RTMPE is faster than SSL, does not require certificate management, and is enabled in the server's Adaptor.xml file. If you specify RTMPE without explicitly specifying a port, the Flash Player scans ports just like it does with RTMP, in the following order: 1935 (RTMPE), 443 (RTMPE), 80 (RTMPE), and 80 (RTMPTE).

RTMPTE This protocol is RTMPE with an encrypted tunneling connection. The default port is 80.

RTMFP (Real-Time Media Flow Protocol)

Flash Player 10, AIR 1.5

This protocol is a UDP-based, secure network transport between Flash Player or AIR and Adobe Media Server. UDP provides lossy transmission, lower latency, and better prioritization than RTMP. RTMFP can be used for client-to-server and peer-to-peer communication. RTMFP allows clients to publish and play audio and video to each other without relaying media through the server. Clients must connect to Adobe Media Server before they can pass data to each other. You can use RTMFP to build peer-assisted networking applications. You can also use RTMFP for traditional broadcast applications.

HTTP (Hypertext Transfer Protocol)

Progressive download

Flash Media Server 3.5 or higher

If you install Apache HTTP Server with Adobe Media Server, you can deliver the client SWF file, container HTML page, and additional assets over HTTP.

You can write client-side ActionScript that causes Apache to serve media assets over HTTP progressive download if RTMP streaming fails. For example, if a client attempts to stream a video over RTMP and fails, the server attempts to tunnel RTMP over HTTP. If that attempt fails, the server delivers the video over HTTP.

See Configuring Apache HTTP Server.

Adobe HTTP Dynamic Streaming and Apple HTTP Live Streaming

Flash Media Server 4.5 or higher

Delivering content over HTTP is called “progressive download”. The content must transfer from the server to the client in a progression from the beginning to the end of a file. A client cannot seek to a forward location until that location and all the data before it has downloaded.

Delivering content over RTMP is called “streaming”. The media server (such as Adobe Media Server) creates a socket connection to the client over which the content is sent in a continuous stream. The client can seek to any point in the content instantly, regardless of how much data has been transferred.

HTTP Dynamic Streaming combines these approaches. HTTP Dynamic Streaming packages media files into fragments that Flash Player clients can access instantly without downloading the entire file. Adobe HTTP Dynamic Streaming contains several components that work together to package media and stream it over HTTP to Flash Player. These components install with Adobe Media Server.

Flash Media Server 4.0 supports live real-time packaging for HTTP Dynamic Streaming.

Flash Media Server 4.5 adds support for live and on-demand real-time packaging for Apple HTTP Live Streaming. Serve on-demand and live media to Flash Player, AIR, and iOS devices over HTTP.

Flash Media Server 4.5 also adds support for on-demand real-time packaging for Adobe HTTP Dynamic Streaming.

See Configure Apache for Adobe HTTP Dynamic Streaming and Apple HTTP Live Streaming, Stream live media (HTTP), and Stream on-demand media (HTTP).

Supported file formats and codecs

It's important to understand the difference between a file format and a codec. A file format is a container that holds one or more pieces of media. F4V is a file format. A codec describes the format of the media. H.264 is a codec. A container can hold multiple pieces of media. For example, an F4V file can contain audio that uses the AAC codec and video that uses the H.264 codec.

File formats

All editions of Adobe Media Server stream the following file formats:

Note: *Adobe recommends that you use the MP4 or F4V format for video streaming applications. In the future, Adobe may discontinue support for the FLV format.*

FLV All versions of Adobe Media Server support playback and recording of the FLV file format. On-demand, real-time HTTP Dynamic Streaming does not support the FLV format.

F4V (MPEG-4 compatible) Adobe Media Server versions 3 and later support playback of all Adobe Media Server-supported audio and video codecs within MPEG-4 Part 12 container formats. File types supported by the MPEG-4 format include F4V, MP4, M4A, MOV, MP4V, 3GP, and 3G2.

Flash Media Server 3.5 introduces support for recording in MPEG-4 format. Live video and audio streams containing any codecs supported by Adobe Media Server can be recorded on the server into the F4V format.

Important: *If a file contains portions of audio and video whose codecs are unsupported by Adobe Media Server, those parts of the stream do not play.*

MP3 The MPEG Layer-3 (mp3) file format is an audio file format that uses lossy data compression.

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

F4M The Adobe Media Manifest file format contains information about a Adobe Media asset. Manifest information includes the location of media, codecs, resolutions, and the availability of files encoded at multiple bit rates, DRM authentication information, and so on. Use the F4M file format for HTTP Dynamic Streaming.

F4F The F4F file format describes how to divide media content into segments and fragments. Each fragment has its own bootstrap information that provides cache management and fast seeking. Use the F4F file format for HTTP Dynamic Streaming.

Use the following table to see which file formats, server versions, and Flash Player versions the codecs support:

File format	Codec	Adobe Media Server version	Flash Player version
FLV	Sorenson Spark	1	6; AIR 1; Flash Lite 3
FLV	Nellymoser	1	6; AIR 1; Flash Lite 3
FLV	On2 VP6	1	8; AIR 1; Flash Lite 3
FLV	Speex	3	10; AIR 1.5
MPEG-4: F4V, MP4, M4V, 3GPP	H.264*	Play back—3; Recording—3.5	9,0,115,0; AIR 1
MPEG-4: F4V, MP4, M4V, 3GPP	AAC+ / HE-AAC / AAC v1 / AAC v2	Play back—3; Recording—3.5	9,0,115,0; AIR 1

File format	**Codec**	**Adobe Media Server version**	**Flash Player version**
MPEG-4: F4V, MP4, M4V, 3GPP	MP3	Play back—3; Recording—3.5	9,0,115,0; AIR 1
MPEG-4: F4V, MP4, M4V, 3GPP	On2 VP6	Play back—3; Recording—3.5	9,0,115,0; AIR 1
mp3	mp3	1	6; AIR 1; Flash Lite 3
RAW	All codecs supported by Flash Player.	3.5.3	6; AIR 1; Flash Lite 3
F4M	N/A	4.0	10.1, AIR 2
F4F	All codecs supported by Flash Player except mp3 and Speex.	4.0	10.1, AIR 2

Note: *H.264 playback in Flash Player supports most profiles including Base, Main, and HiP. The F4V format is a subset of MPEG-4 ISO 14496-10 and AAC+ (ISO 14496-3).*

Codecs

Adobe Media Server doesn’t encode or decode audio and video information. Adobe Media Server streams media that has already been encoded.

To encode on-demand media, use any codec that supports the version of Flash Player or AIR that you want to target.

To capture, encode, and stream live video, you can either use Flash Media Live Encoder or use ActionScript to build your own Flash Player or AIR client. Flash Media Live Encoder 2.5 and later encodes video in the On2 VP6 codec or the H.264 codec. Flash Player and AIR encode live video with the Sorenson Spark codec and encode live audio with a proprietary Nellymoser codec.

The following table lists the supported codecs and their earliest required SWF file format and Flash Player versions:

Codec	**Audio or Video**	**SWF file format version (earliest supported publish version)**	**Flash Player version (earliest version required for playback)**
Sorenson Spark	Video	6	6, Flash Lite 3, AIR 1
Nellymoser	Audio	6	6, Flash Lite 3, AIR 1
mp3	Audio	6	6, Flash Lite 3, AIR 1
On2 VP6	Video	8	8, Flash Lite 3, AIR 1
H.264 (MPEG-4 Part 10)	Video	9	9 Update 3, AIR 1
AAC (MPEG-4 Part 3)	Audio	9	9 Update 3, AIR 1
Speex	Audio	10	10, AIR 1.5

Note: *Flash Player and AIR support an alpha channel in the On2 VP6 codec only.*

Flash Player 9 Update 3 (9, 0, 115, 0) supports playback of the following subsets of the MPEG-4 file format standards:

MPEG-4 standard	Flash Player 9 Update 3/AIR support
ISO/IEC 14496-3 (Audio AAC)	AAC Main; AAC LC; SBR. These codecs are also known as HE-AAC.
ISO/IEC 14496-10 (Video AVC)	Base (BP); Main (MP); High (HiP). All levels are supported.
ISO/IEC 14496-12 (Container)	1 Audio track; 1 Video track
3GPP TS 26.245 (Timed text format)	Full support

Supported metadata formats

Flash Media Server 3.5 or higher

You now deliver Adobe Extensible Metatdata Platform (XMP) metadata embedded video streaming through Adobe Media Server to Flash Player. XMP is a labeling technology that allows you to embed data about a file into the file itself. XMP metadata is a system that communicates critical media information from the point where the media is created to the point where media is viewed. With XMP, applications and publishing systems can capture, share, and leverage metadata. In Adobe Media Server, all the metadata embedded within a media file is accurately delivered. In addition, speech-to-text metadata embedded within files and encoded from Adobe Media Encoder CS4 can be delivered.

Language libraries

Client-side ActionScript API

Client-side scripts run in Flash Player, AIR, or Flash Lite. You can use Adobe Flash or Adobe Flex to write client-side scripts in ActionScript that access server resources (such as capturing live audio and video and streaming it to the server, which streams it to all connected clients).

Developers can use any Flash Player classes in a client-side script. The following classes are used to access Adobe Media Server resources:

Camera Captures video from a camera attached to a computer running Flash Player or AIR. Use the NetConnection and NetStream classes to transmit the video to Adobe Media Server. Adobe Media Server can send the video to other servers and broadcast it to other clients running Flash Player, AIR, or Flash Lite.

GroupSpecifier The GroupSpecifier class is used to construct the opaque groupspec strings that can be passed to NetStream and NetGroup constructors. A groupspec specifies an RTMFP Peer-to-Peer Group, including the capabilities, restrictions, and authorizations of the member using the groupspec.

Microphone Captures audio from a microphone attached to a computer running Flash Player or AIR. Use the NetConnection and NetStream classes to transmit the audio to Adobe Media Server. Adobe Media Server can send the audio to other servers and broadcast it to other clients running Flash Player, AIR, or Flash Lite.

NetConnection A two-way connection between the client and Adobe Media Server or between Adobe Media Server and Flash Remoting.

NetGroup Instances of the NetGroup class represent membership in an RTMFP group.

NetStream A one-way stream between the client and Adobe Media Server through a connection made available by a NetConnection object.

SharedObject Share data between multiple SWF files. You can also share data between multiple Adobe Media Server application instances.

Video Displays live or recorded video in an application without embedding the video in a SWF file. A Video object is a display object on the application's display list and represents the visual space in which the video runs in a user interface.

For more information, see ActionScript 3.0 Reference for the Adobe Flash Platform.

Server-side ActionScript API

Adobe Media Server Extended, Adobe Media Server Professional, and Adobe Media Server Starter provide access to Server-Side ActionScript. You can write server-side code to control log-in procedures, republish content to other servers, allow and disallow users access to server resources, and to allow users to update and share information.

Server-Side ActionScript is Adobe's name for JavaScript 1.5, which is similar, but not identical to, ActionScript 1.0. Both languages are based on ECMAScript Language Specification Edition 3, but ActionScript 1.0 did not implement the specification exactly. Server-Side ActionScript runs in the Mozilla SpiderMonkey engine embedded in Adobe Media Server.

The following are the Server-Side ActionScript classes:

Application A singleton class that represents an instance of an application in a server-side script. Use the Application class event handlers to control the flow of code in an application and to accept, reject, or redirect connections.

ByteArray Provides methods and properties to optimize reading, writing, and working with binary data.

Client Represents a client connected to an application. The server creates a client object each time a client connects. Use this class to get information about a client, set read and write access to server resources, and for remote method invocation.

File Lets applications write to the server's file system. Use this class to store information without using a database, to create log files for debugging, or to track usage.

GroupSpecifier Defines the capabilities, restrictions, and authorizations of an RTMFP peer-to-peer group. Pass GroupSpecifier objects to the NetGroup and NetStream constructors to create a peer-to-peer group and multicast audio and video.

GroupControl The GroupControl class represents an association between a remote peer and a group that it has joined.

LoadVars Use this class to send variables in an object to a specified URL and load variables at a specified URL into an object over HTTP.

Log Use this class to create log files when using web services.

MulticastStreamInfo specifies various Quality of Service (QoS) statistics related to a NetStream object's underlying RTMFP peer-to-peer and IP Multicast stream transport.

MulticastStreamIngest Use the MulticastStreamIngest class to ingest RTMFP multicast streams and convert the multicast data units to messages that a Stream object can use.

NetConnection Creates a two-way connection between Adobe Media Server and an application server, another Adobe Media Server, or another application instance on the same server.

NetGroup An RTMFP group. Use this object to post messages to the group, directly route messages to a group member, and replicate objects within the group. There are several NetGroup helper classes, as well.

NetStream Opens a one-way stream through a NetConnection object between two installations of Adobe Media Server. You can also use this class to publish a source Stream into an RTMFP Group as a multicast stream.

ProxyStream Proxies a stream from one Adobe Media Server to another Adobe Media Server over a server-side NetConnection object. Use the ProxyStream class to build large-scale applications that use DVR functionality.

SharedObject Stores data on the server and shares data between multiple client applications in real time.

SHA256 An implementation of SHA-256 (secure hash algorithm) as described in Federal Information Processing Standards Publication 180-2. Use this class to generate a digest, or signature, for binary data. A receiver of the data can compute its own digest and compare that to the original digest value to ensure that the binary data has not been tampered with.

SOAPCall The object type returned from all web service calls.

SOAPFault The object type of the error object returned to `WebService.onFault()` and `SOAPCall.onFault()`.

Stream Use this class to manage or republish streams. For example, use the Stream class to create server-side playlists. You can also use this class to send data to clients subscribed to a stream and to add metadata to a live stream.

WebService Use this class to create and access a WSDL/SOAP web service.

XML Use this class to load, parse, send, build, and manipulate XML document trees.

XMLSocket Use this class to implement client sockets that let Adobe Media Server communicate with another server identified by an IP address or domain name.

XMLStreams A variation of the XMLSocket class that transmits and receives data in fragments.

For more information, see the Server-Side ActionScript Language Reference.

Plug-in API

Use the Adobe Media Server Plug-in API to write C++ plug-ins that extend the functionality of the server.

Access plug-in Adds a layer of security before the server accepts connection requests from clients. The Access plug-in examines each connection request before it reaches the script layer of the server. The plug-in determines whether the server accepts or rejects the request based on server statistics, such as the number of current connections. You can code the Access plug-in to set permissions to server resources such as streams and shared objects. You can also code the Access plug-in to assign client connections to applications to certain core processes.

Authorization plug-in Authorize client access to events that occur on the server. Use the Authorization plug-in to authorize connections to the server, playing of streams, and publishing of streams. You can also map logical URLs to physical locations. The Authorization plug-in can deliver content according to the geographic location of the client, the subscription level, or the duration a client has viewed a stream. In Flash Media Server 3.5 and later, you can access client statistics, such as bytes in and out, RTMP messages in and out, and RTMP messages dropped.

File plug-in Provides an interface between the file I/O mechanism and the server. Developers can implement an alternative file I/O system in place of the default operating system-based file system. The File plug-in interface supports a fully asynchronous model for file operations, making it easier to implement network-based file I/O. In Adobe Media Server 3.5 and later, the File plug-in supports operations on streams and SWF files.

For more information, see Developing Plug-ins and Adobe Media Server Plug-in API Reference.

Administration API

The Administration API can be used to monitor, manage, and configure the server from a Flash Player or AIR client over RTMP or from a web client over HTTP. The Administration API lets you do the following:

- Monitor the server and its applications and services.
- Perform administrative tasks, such as adding administrative users and starting and stopping the server, virtual hosts, and applications.
- View and set values for server configuration keys.

Some methods are available only to server administrators; virtual host administrators cannot use these methods. In some cases, virtual host administrators can use a method with restrictions; any restrictions are described in the entry for that method. For more information, see Adobe Media Server Administration API Reference.

Scaling the server

Deploying a cluster of servers

Scaling Adobe Media Server increases the number of supported connections and the hardware capacity for streaming and for multi-way applications. To scale Adobe Media Server, you can deploy a cluster of origin servers. With all editions except Adobe Media Server Standard, you can also set up an edge-origin server cluster.

Clustering improves the performance of a single server by maintaining assets locally for easy deployment. In a cluster, clients request a connection from a common URL, and either a hardware load balancer or a server-side script directs the connection to Adobe Media Server.

Using hardware to load balance

When clients request a connection to a Adobe Media Server application, hardware load balancers can direct the connection to the server. Hardware load balancers provide various approaches to distributing client load over multiple servers. This approach is best when clients do not need to communicate with each other (for example, as they would in a text or video chat application).

Using Server-Side ActionScript to load balance

Note: *Adobe Media Server Standard does not allow access to Server-Side ActionScript.*

You can write a server-side script to redirect incoming connections to a specific server in a cluster, for example, in a multi-way application that requires connections to be routed to a specific server, or to perform load balancing. You can choose several methods of assigning clients to servers. For example, can assign clients randomly, or you can call the Administration API `getServerStats()` method to determine which server has the lightest load.

Deploying edge servers

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

Although no edition of the server has a connection or bandwidth limit, all server editions are restricted by hardware capacity. Every connection into the origin server consumes resources independent of the data flowing through the connection. As the number of connections increase, this load can become very large and adversely affect server performance.

To overcome this restriction and increase the number of simultaneous users that can connect to the server, you can configure the server to run as a reverse proxy or *edgeserver*. With an edge server, clients connect to the edge server rather than connecting to the origin server. The edge server aggregates requests from a large number of clients and sends them to the origin.

Edge servers are a good solution for one-way video on demand (vod), one-way live events, and one-way live 24x7 streams. Edge servers enhance security, distribute the load of connection requests, and conserve bandwidth and system resources for high-volume one-way streaming.

Edge servers manage connections, cache content, and push data to clients. Having assets cached at the edge reduces the need for the server to access storage—a process that can be a bottleneck with large-scale media delivery—and delivers video to the client faster. The content can be cached in memory and also on local storage. The server administrator can control the size of the cache by setting a limit on the amount of memory used by the cache. They can also control how the cache is cleared by configuring the least-recently-used (LRU) settings such as the number of cache files and the number of files that are cleared.

Note: *Server-Side ActionScript is executed on origin servers, not on edge servers.*

For more information, see Deploying edge servers.

Deploying edge servers in geographic zones

You can deploy edge servers individually or in clusters. Edge servers can also be chained, where one edge server collects and aggregates the connection requests from other edge servers and their clients, then transmits the requests to an origin server.

To distribute the demands on network and system resources, administrators can assign clients in a geographic region to a specific edge server. For example, one edge server might aggregate and forward requests from clients in Tokyo and another might aggregate and forward requests from Paris. The edge servers in Paris and Tokyo gather the requests from their clients and forward them to the origin server located in another location, such as Chicago.

Clients in these zones always access the origin server through their assigned edge servers. These edge servers receive the responses from the origin server, then distribute them back to the clients in their respective zones: Paris or Tokyo. An edge server also stores the data received from the origin server in a cache, and makes it available to other clients that connect to the edge server. Reusing the data is one more way that edge servers use resources efficiently; caching content reduces the overall load on the origin server.

Deploying edge servers in two tiers

To further distribute the load, you can stack edge servers in two tiers, a host tier and a distribution tier. For example, you could have an origin server in London that broadcasts content all over the world. The origin would push data to a tier of host edge servers in, for example, New York, Munich, Caracas, and Tokyo. Each host edge server would be connected in series to a cluster of edges that would comprise the distribution tier.

Scaling peer-assisted networking applications

Flash Media Server 4.5 or higher

Flash Media Server 4.5 introduced RTMFP clients to each other so the clients can connect to each other directly. This direct connection is called a peer-to-peer connection. Flash Media Server 4.0 can introduce clients to each other only if the clients are connected to a single server. Use Server-Side ActionScript APIs added in Flash Media Server 4.5 to introduces clients to each other even if the clients are connected to separate servers. Distributing introductions across servers allows you to scale peer-assisted networking applications.

See Distribute peer introductions across servers.

Configuring the server

Configure the server for virtual hosting

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

Each of these levels—server, adaptor, virtual host, and application—has distinct configuration parameters stored in XML files: Server.xml, Adaptor.xml, Vhost.xml, and Application.xml. The most important configuration parameters have been pulled out to the ams.ini file, which lets you use one file to configure the server. You can tune each level of the server to provide the highest quality of service for each application the server hosts.

For more information, see Configure the server for virtual hosting.

Tuning server performance

Edit the XML configuration files stored in RootInstall/conf to tune server performance. You can tune the server to provide maximum performance for the type of application you are delivering. You can also tune each server level: server, adaptor, and virtual host. Configure the size of the recorded media cache, the size of stream chunks, how to combine audio samples, whether to limit connections, and so on.

For more information, see Configuring performance features.

Managing content

To manage content served by Adobe Media Server, you can do any of the following:

- Setting the location of application files
- Mapping directories to network drives
- Setting the location of recorded streams and shared objects
- Create virtual directories to store streams and shared objects. See Mapping virtual directories to physical directories.
- Use the Authorization plug-in to map virtual directories to physical locations. See Developing an Authorization plug-in.
- Use the File plug-in to access content over HTTP. See Developing a File plug-in.
- Use the File plug-in to create a custom file I/O system. See See Developing a File plug-in.

Administering the server

Administration Console

The Administration Console (built with the Administration API) has an easy-to-use interface to manage administrators, monitor servers, and view details about applications running on the server. There are two levels of Administration Console users: server administrators and virtual host administrators. This enables hosting organizations to provide lesser administrative rights to companies using their hosting services. Application developers use the Administration Console to debug applications.

For more information, see Using the Administration Console.

Command-line tools

Adobe provides two command-line tools to help you administer the server: AMSCheck and FLVCheck. AMSCheck provides information about whether the server is running or not, what the response time is, and which amscore processes are not responding. AMSCheck supports only FLV files.

FLVCheck lets you verify that a video file will run properly on Adobe Media Server. The FLVCheck tool supports MP4 files and FLV files.

See Checking server statusand Checking video files.

Log files

Adobe Media Server logs provide real-time server monitoring to help you manage the server and troubleshoot issues. The log files track activity, such as general traffic and server load, who is accessing the server, client behavior and interaction, and general diagnostics. Log files are written in W3C format. You can use standard parsing tools to parse the log files. You can also view log data in the Administration Console.

Adobe Media Server maintains the following logs in the RootInstall/logs folder:

access.xx.log Tracks information about server access. For example, the date and time a client connected to the server and the total bandwidth consumed during a session. It also tracks the streams accessed by the connection and whether the client published a stream. In addition, it tracks whether the client jumped to a new location within a recorded stream.

application.xx.log Tracks information about activities in application instances. For example, the date and time of the event and the server process ID of the event. It also tracks the event status level (warning, error, information, debug, and so on).

error_log.log Tracks information about Apache HTTP Server errors. This file is in the *RootInstall*/Apache2.2/logs directory.

access_log.log Tracks information about Apache HTTP Server access. This file is in the *RootInstall*/Apache2.2/logs directory.

Diagnostic logs Track information about server operations; for example, information about stream events, application instances, virtual hosts, and edge/origin issues. By default, the server is configured to create a diagnostic log for each type of server process. The default diagnostic logs are master.xx.log, edge.xx.log, core.xx.log, admin.xx.log, and httpcache.xx.log.

For more information, see Monitoring and Managing Log Files.

Chapter 3: Security

All editions of Adobe Media Server support features that protect your content from being stolen and misused. Some features, such as true streaming, are intrinsic to the server and don't need to be configured. Other features, such as enhanced RTMP (RTMPE), can be configured or disabled using XML configuration files. Still other features, such as controlling read and write access to specific server folders, can be custom built using client-side ActionScript and Server-Side ActionScript.

***Important:** For the definitive guide to Adobe Media Server security, see Hardening guide for Adobe Media Server in the Adobe Developer Center. This guide was written by Adobe Media Server Escalations and Security Engineering Manager, Asa Whillock.*

Streaming content securely

True streaming

If an application uses progressive download (often called *progressive streaming*), content is downloaded to the client's hard drive where malicious agents can capture the video and redistribute it. Adobe Media Server uses true streaming, not progressive download. This means that media streamed from Adobe Media Server to Adobe Flash Player is not stored locally in the client's cache or anywhere on the client's hard drive. There is no configuration necessary—Adobe Media Server always uses true streaming technology to protect your content.

Secure network protocols

Adobe Media Server supports two network protocols that offer different levels of security. Select the network protocol that best meets your organization's and your application's needs:

RTMPE Uses a 128-bit encrypted channel for data between the client and the server. It does not use certificate management. It is best for applications that don't require endpoint authentication and that require more performance and speed than is possible with SSL. RTMPE requires 15% more processing power than RTMP. To specify an encrypted channel, use the RTMPE protocol in the connection URI, for example:

```
nc.connect("rtmpe://www.example.com/mediaapplication")
```

Data passed over RTMPE is encrypted with the well-known stream cipher encryption algorithm. This algorithm is keyed using a well-known public-key-exchange algorithm which prevents passive observation and provides perfect forward secrecy.

***Note:** Flash Player 9 Update 3 and later and AIR 1.0 and later support RTMPE .*

RTMPS A protocol for enabling secure communications over TCP/IP. Adobe Media Server provides native support for both incoming and outgoing SSL connections with the RTMPS protocol. SSL protects against domain impersonation. It allows you to choose the level of encryption you want. SSL potentially provides the highest level of security but requires extra processing power, almost 50% more than RTMP. To specify an RTMPS connection, use the RTMPS protocol in the connect URI, for example:

```
nc.connect("rtmps://www.example.com/mediaapplication")
```

Data passed over RTMPS is protected by SSL guarantees. The server certificate check authenticates that the server is a valid server but not that the client is a valid Flash Player or AIR client. See Configure SSL.

RTMFP Uses 128-bit encryption. Data passed over RTMFP is encrypted with the well-known block cipher encryption algorithm. The algorithm is keyed using a well-known public-key-exchange algorithm which prevents passive observation and provides perfect forward secrecy. Session nonces are provided and are tied to the key exchange so that you can build client and server authentication on top of the exchanged values.

To establish a peer-to-peer connection, a subscribing peer must know the peer ID of a publishing peer. These peer IDs are guaranteed to be unique for each instance of Flash Player. Peer IDs are 256 bit values created from a cryptographically random source. When an RTMFP peer connects to another RTMFP peer using a peer ID, the public-key-exchange algorithm is tied to the peer ID so that it is not possible to conduct a man-in-the-middle attack. All other guarantees provided by RTMFP are also in place.

To specify an RTMFP connection, use the RTMFP protocol in the connect URI, for example:

```
nc.connect("rtmfp://www.example.com/mediaapplication")
```

Session nonces Flash Player 10 and Adobe Media Server 3.0.3 and 3.5.1 add support for session nonces for RTMFP and RTMPE. Nonces are tied to the key exchange so that you can build client and server authentication on top of the exchanged values.

On Flash Player, nonces are available as properties of the NetConnection object (`nearNonce` and `farNonce`). On Adobe Media Server, nonces are available on the Server-Side ActionScript Client object (`nearNonce` and `farNonce`), the Server-Side ActionScript NetConnection object (`nearNonce` and `farNonce`) the Access plug-in (`nearNonce` and `farNonce` fields on the CONNECT event), and on the Authorization plug-in (`F_CLIENT_NEAR_NONCE` and `F_CLIENT_FAR_NONCE` fields on any authorization event that has a Client object).

Protecting content

Protected HTTP Dynamic Streaming and Protected HTTP Live Streaming

Flash Media Server 4.5 or higher

Use Protected HTTP Dynamic Streaming to serve live and on-demand protected content to Flash Player and AIR over HTTP without using a DRM License Server. Protected HTTP Dynamic Streaming also supports SWF verification. When Adobe Media Server packages the content, it generates the license and embeds it in the metadata of the content stream.

Use Protected HTTP Live Streaming to serve live and on-demand protected content to iOS and MacOS without using a DRM License Server.

For more information, see Protected HTTP Dynamic Streaming and Protected HTTP Live Streaming.

Using Adobe Access

Adobe Media Server can stream content encrypted by Adobe Access to Flash Player and AIR applications. Adobe Media Server can stream encrypted FLV files, encrypted MP4/F4V files, and encrypted HTTP Dynamic Streaming files. You can stream encrypted files using all supported protocols.

No specific configuration is required for Adobe Media Server to work with Adobe Access.

For more information, see the Adobe Access.

Verifying clients

Adobe Media Server supports features that prevent unauthorized clients from sending streams to the server or from playing streams they aren't authorized to access.

All server editions support the following features:

- Verify SWF files.

  You can configure Adobe Media Server to verify client SWF files before allowing them to connect to an application. This technique ensures that only SWF files you created can access this server. Third parties cannot create their own SWF files that attempt to stream your resources. For more information, see Verify SWF files.

- Allow and deny connections from specified domains.

  For more information, see Restrict which domains can connect to a virtual host.

All editions except Adobe Media Server Standard support the following features:

- Use the File plug-in to verify SWF files. See Retrieving external SWF files for verification.
- Generate a unique key that is verified against the server, or request and accept an encrypted token from an application server.
- Use the Server-Side ActionScript Client object. This object provides information about clients that you can use to accept or reject connection requests. Check the URL of the SWF file or the server from which the client connection originated using the `Client.referrer` property. Check the IP address of the client using the `Client.ip` property.
- Verify the Flash Player version.

For more information about these features, see Securing applicationsin the *Developer's Guide*.

Controlling server access

The following techniques allow developers and administrators to control the data a client can access:

Note: *The following features are not supported by Adobe Media Server Standard.*

- Use the Server-Side ActionScript `Client.readAccess` and `Client.writeAccess` properties to specify a client's read/write permissions to server resources, such as shared objects and streams.
- Use the Server-Side ActionScript `Client.audioSampleAccess` and `Client.videoSampleAccess` properties to allow a client to access raw, uncompressed data from streams in specified folders.
- Use the Authorization plug-in to authorize access to events on the server such as playing and publishing streams. See Developing an Authorization plug-in.

Note: *The following features are supported by all server editions.*

- Configure virtual storage folders for streams, shared objects, and files. For more information, see Configuring content storage.
- Use the Access plug-in to accept, reject, or redirect connections before they reach the server-scripting layer. For more information, see Developing an Access plug-in.

Authenticating users

You can pass credentials, such as a user name and password, in the client-side `NetConnection.connect()` call and verify them against an external resource, such as a database, LDAP server, or other access-granting service. For more information, see Authenticate users.

Configuring the server securely

Securing the Administration Console

The following techniques let administrators secure access to the Administration Console:

- Define administrator users and passwords.
- Define IP address ranges or domain names required or denied to administrator users. See AdminServerin the XML configuration files chapter.
- Set a limit to the number of allowed login attempts. See `LogInLimits`.
- Define a connection port for administrators, thus allowing access only to users behind a firewall. See Manage administrators.

Securing server performance

The following techniques let administrators configure the server to maintain performance levels:

- Log server access and application events.
- Set performance parameters, such as thread limits, garbage collection intervals, and stream allocation size.
- Set limits on application instances, streams, and shared objects, to prevent an application from consuming all the disk space on a computer.
- Limit bandwidth capacity for an application, to prevent one application from consuming all the network bandwidth on a computer.
- Set the default bandwidth for a client connecting to an application.

For more information, see Configuring performance features.

Security for server-side scripts

Limiting script memory usage

In the `ScriptEngine` section of the Application.xml configuration file, you can limit the amount of memory that can be used by Server-Side ActionScript on the virtual host.

You can also configure other aspects of the JavaScript (Server-Side ActionScript) engine, such as how often garbage collecting occurs, the maximum amount of time a function can take to execute, and the script library path.

For more information about editing configuration files, see Working with configuration files. For more information about configuring the script engine, see ScriptEngine.

Loading a script securely

The Adobe Media Server script security model enables administrators to limit the exposure to potentially malicious or buggy third-party code that may be included on the server side. The script security model is not designed to detect or prevent error conditions such as an infinite loop in third-party code, but it is useful for preventing or limiting certain potentially dangerous functionality, such as the ability to make arbitrary connections, or read or write file objects.

Script security can be very valuable when building dynamically extensible applications that load and evaluate code from external sources.

When an application is started, the server looks in the application's folder for a secure.asc file. If the file exists, the server loads it. During this period of time, it makes the `protectObject()` and `getGlobal()` methods available. Use these methods to manipulate global functions, classes, and objects in a way that is not possible during normal application execution. The `getGlobal()` method provides access to the global object from the secure.asc file while the file is loading. The `protectObject()` method prevents application code from accessing or inspecting the methods of an object directly. You can only use the `protectObject()` method in the secure.asc file. Once the server is done loading the secure.asc file, these methods are unavailable. Then the server loads the main.asc file and other scripts in the normal manner.

For more information, see Adobe Server-Side ActionScript Language Reference.

Secure application scenario

All server editions offer many ways to protect video streams. This example scenario uses external authentication, encrypted Real-Time Messaging Protocol (RTMPE), and the ability to verify connecting SWF files.

External authentication Authenticate clients against an external application server (such as a J2EE-based server) and database.

Verify SWF files Verify SWF files by comparing connecting SWF files with a copy of the SWF file stored on the server. Only verified SWF files can access content on the server.

RTMPE Encrypted Real-Time Messaging Protocol sends 128-bit encrypted data between the client and the server. A verified SWF running on the client sends a request to Adobe Media Server over RTMPE and Adobe Media Server streams live or recorded content. The stream is encrypted during transmission.

Note: *A video publisher might have a very large audience, perhaps millions of users. In that case, authenticating users by application server and database is unrealistic. The publisher could eliminate the external authentication and just verify SWF files before allowing them to connect over RTMPE.*