

ADOBE® FLASH® MEDIA SERVER 3.5

Technical Overview

© 2009 Adobe Systems Incorporated. All rights reserved.

Adobe® Flash® Media Server 3.5 Technical Overview

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe, the Adobe logo, Adobe AIR, Adobe Premiere, Acrobat Connect, ActionScript, After Effects, ColdFusion, Flash, Flash Lite, Flex, and XMP are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Windows is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and other countries. All other trademarks are the property of their respective owners.

This work is licensed under the Creative Commons Attribution Non-Commercial 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/us/>

Portions include software under the following terms:

Sorenson Spark. Sorenson Spark™ video compression and decompression technology licensed from Sorenson Media, Inc.

Speech compression and decompression technology licensed from Nellymoser, Inc. (www.nellymoser.com)

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

Notice to U.S. Government End Users. The Software and Documentation are “Commercial Items,” as that term is defined at 48 C.F.R. §2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation,” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

Contents

Chapter 1: Introduction

Server editions	1
System requirements	2
What's new in Flash Media Server 3.5	3
Common uses for the server	5

Chapter 2: Architecture

Core server architecture	8
Language libraries	13
Scaling the server	16
Configuring the server	17
Administering the server	18

Chapter 3: Security

Streaming content securely	20
Protecting content	21
Configuring the server securely	22
Security for server-side scripts	23
Secure application scenario	23

Index	25
--------------------	----

Chapter 1: Introduction

Adobe® Flash® Media Server is a real-time media server that can deliver video on demand, live video, streaming music, video blogging, video messaging, multimedia chat environments, real-time datacasting, and multiuser gaming. Flash Media Server client applications run in Adobe® Flash® Player, Adobe® AIR™, and Adobe® Flash® Lite™ and work consistently across platforms and browsers.

There are three editions of Flash Media Server 3.5: Adobe Flash Media Streaming Server, Adobe Flash Media Interactive Server, and Adobe Flash Media Development Server.

Server editions

Flash Media Streaming Server

Flash Media Streaming Server is a powerful and secure video and audio streaming solution for Flash Player, AIR and Flash Lite. This server is designed to deliver live video and video on demand to Flash Player. It includes two built-in streaming services, live and video on demand (vod), which let you deliver live or recorded content to clients. Flash Media Streaming Server offers a client API that lets you develop Flash Player, AIR, or Flash Lite solutions that use live and vod services. This server edition is not intended for high-performance, highly scalable, or custom video solutions.

Flash Media Streaming Server provides two streaming services: live streaming media and video on demand streaming media. You can create clients that use these services to stream live video to an unlimited number of clients. A live stream could be time delimited, like a short event, or always on, like a television or radio station. Recorded video could be anything from short commercials, movie trailers, and music videos to television programs and full-length movies.

Flash Media Interactive Server

Flash Media Interactive Server is a powerful, secure, extendable, and scalable solution for delivering streaming media and interactive applications such as video recorders and multiplayer games or DVR-enabled live streams for Flash Player, AIR, and Flash Lite. The server offers an SDK that lets you develop media applications in Server-Side Adobe® ActionScript® and client-side ActionScript. You can also develop plug-ins in C++ that extend the built-in functionality of the server. Flash Media Interactive Server also can be configured to run as an edge server that manages connections and bandwidth, and caches content closer to clients so you can scale your deployment easily.

Flash Media Interactive Server provides the same streaming services as Flash Media Streaming Server. In addition, with Flash Media Interactive Server, you can use Server-Side and client-side ActionScript to develop new video services or interactive applications that stream audio and video that users generate, such as video blogging and messaging and personal video broadcast sites. This edition of the server also supports DVR functionality to enable instant replay or catch-up streaming solutions. You can also create real-time media applications that let users chat or collaborate in real time. Flash Media Interactive Server is the upgrade from Flash Media Streaming Server.

Flash Media Interactive Server is also designed for broadcasting data to large groups of users (for example, in an online game). You can also use the server to create interactive applications with live video and collaboration applications like Adobe® Acrobat® Connect™.

Flash Media Development Server

Flash Media Development Server is a limited, free edition of Flash Media Interactive Server that can be used to develop new applications for Flash Media Interactive Server or, in production, to implement basic low-volume streaming or social media solutions. The server can be used in production to leverage the new multipoint publish feature, which enables you to create a live publishing point on a network, inject data messages into a stream, and push the stream to a Content Delivery Network. Flash Media Development Server has no functionality limit, but it is limited to 10 simultaneous connections.

Server edition comparison

The following table compares the features in each server edition (features supported by all server editions are not listed):

Feature	Flash Media Interactive Server	Flash Media Streaming Server	Flash Media Development Server
Simultaneous connections between client and server	Unlimited	Unlimited	10
Bandwidth	Unlimited	Unlimited	Unlimited
Processor limit	8-way SMP	4-way SMP	8-way SMP
Streaming services (live and vod)	Yes	Yes	Yes
Multiple applications and publishing points	Unlimited	Unlimited (This server edition only runs the vod and live applications)	Unlimited
Archive (record) video on server	Yes	No	Yes
Server-side programming	Yes	No	Yes
Multipoint publishing	Yes	No	Yes
Edge server caching	Yes	No	Yes
Custom C++ plug-in support	Yes	No	Yes
Core server processes	Unlimited	1	1
Encrypted streaming (RTMPE/RTMPS)	Yes	Yes	Yes
SWF verification	Yes	Yes	Yes
HTTP Services	Yes	Yes	Yes
DVR functionality	Yes	No	Yes
Dynamic streaming	Yes	Yes	Yes
Streaming encrypted video	Yes	Yes	Yes

System requirements

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

What's new in Flash Media Server 3.5

DVR functionality for live streaming

Flash Media Server 3.5 allows users to pause live streams and continue playing from where they paused. Users can seek anywhere in the stream. Developers can create experiences that support instant replay or catch-up services. You can use edge caching to scale DVR functionality or to deploy on Content Developer Network.

Content managers can enable this feature on any number of streams on a server. You can make the entire live event available, or only a section of the event. Either way, the video cache is on the server so content is protected. DVR enabled streams support dynamic streaming.

Developers can inject cue points in the live video to trigger dynamic advertising. For example, when a user passes a cue point, the advertising appears. Developers also can create interfaces that disable seeking during advertisements.

DVR functionality is supported in Flash Player 6 and later.

Dynamic streaming

Flash Media Server can receive commands to switch between versions of a single content stream that are encoded at multiple bit rates. This feature lets your media application adapt to changing network conditions. It lets your application adapt to clients with different capabilities, such as mobile devices with lower processing power and smaller screens. You can use this feature to switch between live video encoded with multiple bit rates using Flash® Media Live Encoder 3.0. You can also use dynamic streaming to swap content in a playlist upon events that you specify.

Dynamic streaming works with both FLV files and MP4/F4V files. No special encoding is required. Dynamic streaming is supported in Flash Player 10 and later.

Integrated HTTP Server

All editions of Flash Media Server 3.5 include Apache HTTP Server. This feature lets you deliver client SWF files, container HTML files, and all media assets from the same server. For media assets, you can use Flash Media Server to stream media using RTMP. If streaming over RTMP fails, you can fall back to delivering content over HTTP on the same server.

Use the File plug-in to retrieve external SWF files for verification

You can develop a custom File plug-in to retrieve SWF files that are stored in external content repositories or on another server in a cluster. By retrieving SWF files with the File plug-in, the server can verify SWF files without requiring copies of the SWF files on the server computer. This feature helps reduce the load on your network.

Get client statistics with the Authorization plug-in

When a client connects to Flash Media Server, the server tracks statistical data about the client. This statistical data includes bytes in, bytes out, and so on. In Flash Media Server 3.5, the Authorization plug-in can retrieve client statistical data from the server repeatedly without affecting server performance. This feature lets you create a custom monitoring solution in the C++ application layer.

Dynamically assign applications to core processes with the Access plug-in

With new APIs in the Access plug-in, you can develop a plug-in that dynamically assigns an application to a core process. Use this feature to balance a load across core processes based on real-time performance metrics. You can also use this feature to provide a higher quality of service (QoS) to certain customers.

XMP metadata

You can deliver Adobe Extensible Metadata Platform (XMP) metadata embedded video streaming through Flash Media Server to Flash Player. 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. In addition, speech-to-text metadata embedded within files and encoded from Adobe encoding tools such as Flash Media Live Encoder can be delivered. AMF0 and AMF3 connections are supported. XMP metadata can be internal information about the file or information for end users.

For example, you could create a trailer in Adobe® Premiere® CS4 and transfer the metadata to the FLV file. When users view the file, they can use Flash Player 10 search to look for metadata and jump to a specific location in the file. For detailed information about XMP, see www.adobe.com/go/learn_fms_xmp_en.

See also

“[Supported metadata formats](#)” on page 12

MP4/F4V recording

Flash Media Server can record video in MP4/F4V format. Live video and audio streams containing codecs supported by Flash Media Server can be recorded on the server. You can capture audio and video with Flash Player, Flash Media Live Encoder, a partner solution, or a live stream containing pre-recorded assets. Recording can be started from a client or server API. You can also access record start and stop events in the Authorization plug-in.

See also

“[Supported file formats](#)” on page 11

Speex codec support

Flash Media Server supports the Speex codec when the client application runs in Flash Player 10 or later. For example, a publisher running an application in Flash Player 10 can publish an audio stream encoded using Speex to the server. If a subscriber using an earlier version of the player subscribes to the stream, the subscriber cannot play the audio stream. The subscriber must use Flash Player 10 or later to hear the Speex audio.

See also

“[Supported codecs](#)” on page 11

Start Screen and sample video player

The Flash Media Server Start Screen provides a comprehensive interface to Flash Media Server administration tools, resources, documentation, samples, supports, and community links. The Start Screen also lets you test dynamic streaming, progressive download, the vod service, and the live service with an interactive video player embedded in the page. You can easily embed this video player into your own applications.

To launch the Start Screen, choose Start > All Programs > Adobe > Flash Media Server 3.5 > Flash Media Server Start Screen. If you installed Apache HTTP server with Flash Media Server, enter <http://localhost> in a browser.

Support for additional operating system versions

Flash Media Server 3.5 supports Windows Server 2008 (all editions) and Red Hat Enterprise Linux 5.2.

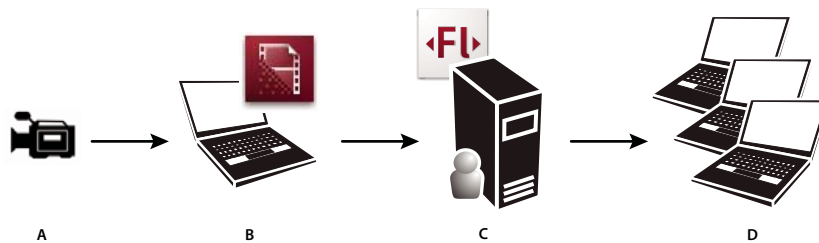
Common uses for the server

Capture and broadcast live video

Adobe Flash Media Live Encoder lets you capture audio and video while streaming it live to Flash Media Server. You can use anything from a webcam to a sophisticated digital video camera to capture the video. For more information about Flash Media Live Encoder, see www.adobe.com/go/learn_fms_fme_en.

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 Flash Media Streaming Server. However, Flash Media Streaming Server does not support multipoint publishing because it requires server-side scripting.



A. Live video B. Flash Media Live Encoder (or custom-build Flash Player or AIR solutions) C. Flash Media Server D. Flash Player, AIR, or Flash Lite clients

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 Flash Media Development server. The development server would publish the stream to a CDN that pushes the stream to millions of users.

Broadcast recorded video

To deliver recorded video, or “video on demand”, Adobe provides a video player called the FLVPlayback component, which supports playback of FLV and MP4/F4V files. Flash Media Server 3.5 also installs with a sample video player of production quality (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 Flash Media Server.

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

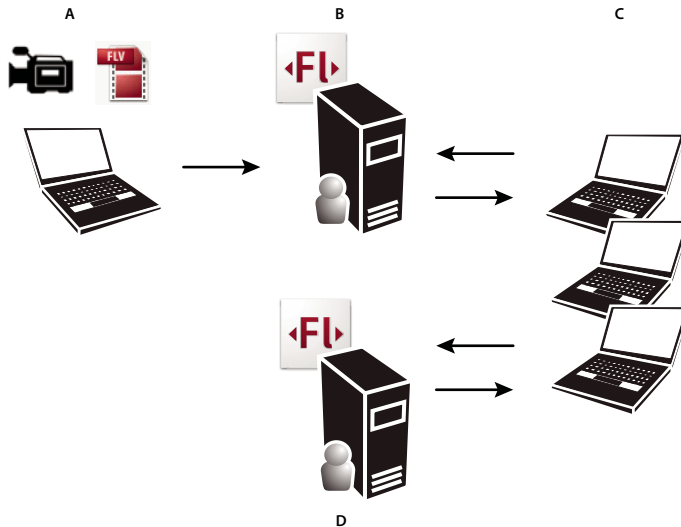
- Short video clips, such as commercials up to 30 seconds long
- Longer video clips, such as user-generated videos up to 30 min. long
- Recorded television shows or movies up to several hours long
- Client-side or server-side media playlists can play a list of streams in a sequence, whether live streams,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



A. Flash Media Server streams recorded media to clients. B. Internet (RTMP) C. Flash Player, AIR, or Flash Lite run the video players

Broadcast video with advertising

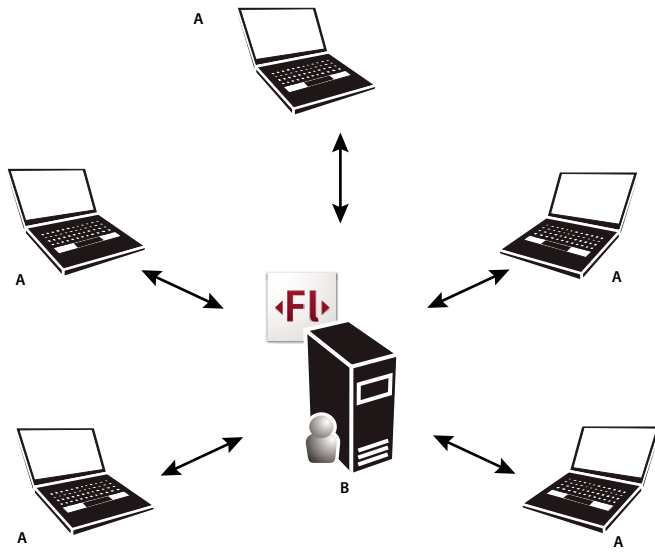
A streaming video application can insert advertising at various points, such as a short commercial that plays before a recorded television show or live video. The advertisement is often streamed from one server and the content is streamed from another server or from a Content Delivery Network. A video-with-advertising 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.



A. Live video B. Flash Media Server (serving recorded and live content) C. Flash Player, AIR, or Flash Lite clients D. Flash Media Server (serving ads)

Integrate video with interactive applications

A Flash Media Interactive 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.



A. Clients can send and receive audio and video and data messages. B. Flash Media Interactive Server broadcasts the media and data to all connected users.

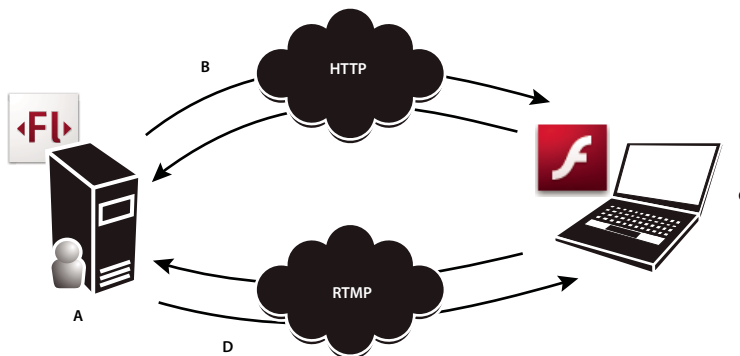
Chapter 2: Architecture

Core server architecture

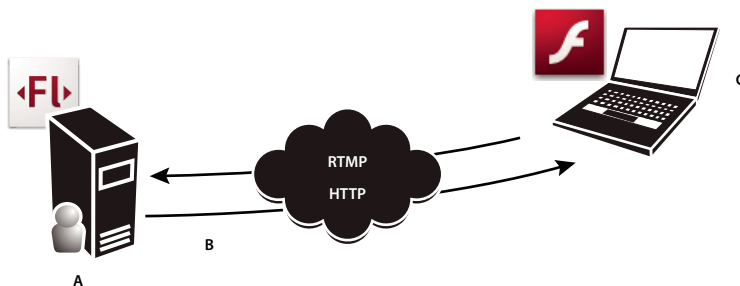
Client-server architecture

Adobe Flash Media Server applications have a client-server architecture. The client code is written in ActionScript (1, 2, or 3) and runs in Adobe Flash Player 6+, Adobe AIR 1+, or Adobe Flash Lite 3+. The server code is written in Server-Side ActionScript, which is like ActionScript 1.0.

The server and the client 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 Flash Media Server over RTMP. The connection allows data to stream between client and server in real time. Flash 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 Flash Media Server installation and serve SWF and HTML content from any external web server.



Flash Media Server applications consist of client and server components that work together. A. Flash Media Server B. Web server sends SWF file. C. Flash Player, AIR, or Flash Lite client plays SWF file. D. SWF file connects to an application on Flash Media Server. The server streams data over a persistent connection.



A. Flash Media Server B. Web server sends SWF file. C. Flash Player, AIR, or Flash Lite client plays SWF file.

Real-time media server

Flash Media Streaming Server provides two streaming services: live and vod (video on demand). 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.

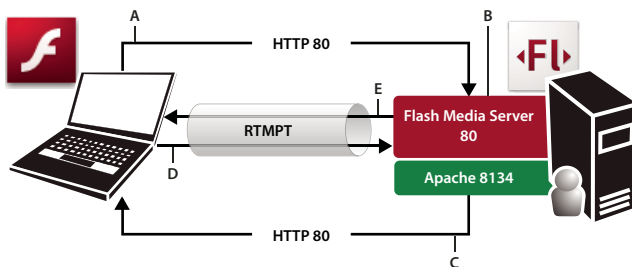
Real-time collaboration application server

Flash Media Interactive Server and Flash Media Development Server include the same streaming services as Flash Media Streaming Server. 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.

Built-in Apache HTTP Server

All versions of Flash Media Server 3.5 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.

Content delivered by HTTP is progressively downloaded and cached by the client. (Content streamed through RTMP is not cached.) By default, Flash Media Server 3.5 proxies HTTP requests from port 80 to port 8134, as shown in the following diagram:



A. Client requests content over HTTP. **B.** Flash Media Server proxies the request to port 8134. **C.** Apache web server delivers content for progressive download. **D.** Client requests content over RTMPT. **E.** Flash Media Server streams content to client.

You can configure proxying in the `fms.ini` file. For more information, see “Configuring the web server” in [Adobe Flash Media Server Configuration and Administration Guide](#).

Hosting multiple applications

Flash Media Interactive Server and Flash Media Development Server can host an unlimited number of applications. Flash Media Streaming Server can host an unlimited number of instances of the live and vod services.

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

For example, Flash Media Interactive Server 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. For more information, see “Connecting to the server” in [Adobe Flash Media Server Developer Guide](#).

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. For more information, see “[Configuring the server](#)” on page 17.

RTMP (Real-Time Messaging Protocol)

All server editions communicate with Flash Player, AIR, and Flash Lite 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.

There are five types of RTMP connections supported by Flash Media Server 3:

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. (Flash 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.

HTTP (Hypertext Transfer Protocol)

Flash Media Server 3.5 includes Apache HTTP Server. If you install and enable Apache, you can deliver the client SWF file, container HTML page, and additional assets over HTTP.

Additionally, 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.

Streaming media

Media (which can contain audio, video, and data) is sent between a client and Flash 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 Flash 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.

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, AIR, or Flash Lite 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: *Flash Media Interactive Server and Flash Media Development Server 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.

Supported codecs

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

To encode media that has already been captured (in other words, media that is not live), 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	SWF file format version (earliest supported publish version)	Flash Player version (earliest version required for playback)
Sorenson Spark	6	6, Flash Lite 3
On2 VP6	8	8, Flash Lite 3
H.264 (MPEG-4 Part 10)	9	9 Update 3, AIR
MP3	6	6, Flash Lite 3
AAC (MPEG-4 Part 3)	9	9 Update 3, AIR
Speex (audio)	10	10
Nellymoser	6	6, Flash Lite 3

Note: AIR is a cross-operating system runtime that contains Flash Player 9 Update 3 or later. Flash Player and AIR support an alpha channel in the On2 VP6 codec only.

Flash Player Update 3 and later 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
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 file formats

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

FLV format All versions of Flash Media Server (1, 2, and 3 and later) support playback and recording of the FLV file format. The FLV file format supports the following codecs: On2 VP6, Sorenson Spark, and MP3.

F4V format (MPEG-4 compatible) Flash Media Server versions 3 and later support playback of all Flash Media Server-supported audio and video codecs within MPEG-4 Part 14 container formats. File types supported by the MPEG-4 format include F4V, MP4, M4A, MOV, MP4V, 3GP, and 3G2.

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

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

Note: *Files recorded in MPEG-4 format cannot be played back by earlier versions of Flash Media Server.*

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

Codec or Data type	File format	Flash Player support	Usual codec pairing
Sorenson Spark	FLV	6, 7, 8, 9, 10	Nellymoser/MP3
On2 VP6	FLV	8, 9, 10; Flash Lite 3	Nellymoser/MP3
H.264*	MPEG-4: MP4, M4V, F4V, 3GPP	9,0,115,0; 10	AAC+/MP3
Nellymoser	FLV	6, 7, 8, 9, 10	Spark/On2
MP3	MP3	6, 7, 8, 9, 10; Flash Lite 3	Spark/On2
AAC+ / HE-AAC / AAC v1 / AAC v2	MPEG-4: MP4, M4V, F4V, 3GPP	9,0,115,0; 10	H.264
SPEEX	FLV	10	Spark/On2
AMF 0	FLV, MPEG-4: MP4, F4V	6, 7, 8, 9, 10; Flash Lite 3	Spark/On2/H.264
AMF 3	FLV, MPEG-4: MP4, F4V	8, 9, 10	Spark/On2/H.264

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).*

Supported metadata formats

You can now deliver Adobe Extensible Metadata Platform (XMP) metadata embedded video streaming through Flash 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 Flash 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. For detailed information about XMP, see www.adobe.com/go/learn_fms_xmp_en.

Data model

Flash 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 Flash 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 Flash Media Streaming Server.

For more information, see [Adobe Flash Media Server Developer Guide](#).

Invoking remote methods

Flash Media Interactive Server and Flash Media Development Server 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 [Adobe ActionScript 2.0 Language Reference for Adobe Flash Media Server](#) and [Adobe ActionScript 3.0 Language and Components Reference](#).

Connecting to external sources

In addition to the communication models that streams and shared objects provide, Flash Media Interactive Server and Flash Media Development Server can interact with external data sources, such as web services and relational databases, or with other Flash Media Server applications. For example, Server-Side ActionScript can be written to connect to a web service (use the `WebService` class) or to a ColdFusion® application to retrieve a list of names and phone numbers. The results of the query can then be placed in a shared object.

For more information, see [Adobe Flash Media Server Developer Guide](#) and [Adobe ActionScript 2.0 Language Reference for Adobe Flash Media Server](#).

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

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 2.0 or ActionScript 3.0 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 Flash 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 Flash Media Server. Flash Media Server can send the video to other servers and broadcast it to other clients running Flash Player, AIR, or Flash Lite.

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 Flash Media Server. Flash 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 Flash Media Server or between Flash Media Server and Flash Remoting.

NetStream A one-way stream between the client and Flash 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 Flash 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 [Adobe ActionScript 3.0 Language and Components Reference](#) and [Adobe ActionScript 2.0 Language Reference for Adobe Flash Media Server](#).

Server-side ActionScript API

Flash Media Interactive Server and Flash Media Development Server 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 Flash 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.

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.

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.

NetConnection Creates a two-way connection between Flash Media Interactive Server (or Flash Media Development Server) and an application server, another Flash Media Interactive Server (or Flash Media Development Server), or another application instance on the same server.

NetStream Opens a one-way stream through a NetConnection object between two installations of Flash Media Server.

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

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 Flash 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

Flash Media Interactive Server and Flash Media Development Server provide a plug-in API. Use the 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, 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 Flash Media Server 3.5, the File plug-in supports operations on streams and SWF files.

For more information, see [Adobe Flash Media Interactive Server Plug-in Developer Guide](#) and [Adobe flash Media Interactive 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 Flash Media Server Administration API Reference](#).

Scaling the server

Deploying a cluster of servers

Scaling Flash Media Server increases the number of supported connections and the hardware capacity for streaming and for multi-way applications. To scale Flash Media Server, you can deploy a cluster of origin servers. With Flash Media Interactive Server and Flash Development Server, you can also set up an edge-origin server cluster, as described in the section “[Deploying edge servers](#)” on page 16.

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 Flash Media Server.

For more information, see “Deploying servers in a cluster” in [Adobe Flash Media Server Configuration and Administration Guide](#).

Balancing a load with hardware

When clients request a connection to a Flash 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).

Balancing a load with Server-Side ActionScript

Note: Flash Media Streaming Server 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, you 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: Flash Media Streaming Server 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 Flash Media Interactive Server and Flash Media Development 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.

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

For more information, see “Deploying the server” in [Adobe Flash Media Server Configuration and Administration Guide](#).

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.

Configuring the server

Provisioning the server

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 `fms.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 “Provisioning the server” in [Adobe Flash Media Server Configuration and Administration Guide](#).

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” in [Adobe Flash Media Server Configuration and Administration Guide](#).

Managing content

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

- Map network drives. See “Mapping directories to network drives” in [Adobe Flash Media Server Configuration and Administration Guide](#).
- Configure the storage location for streams and shared objects. See “Setting the location of recorded streams and shared objects” in [Adobe Flash Media Server Configuration and Administration Guide](#).
- Use the Authorization plug-in to map logical URLs to physical locations. See “Developing an Authorization plug-in” in [Adobe Flash Media Interactive Server Plug-in Developer Guide](#).
- Create virtual directories to store streams and shared objects. See “Mapping virtual directories to physical directories” in [Adobe Flash Media Server Configuration and Administration Guide](#).
- Use the File plug-in to access content over HTTP. See “Developing a File plug-in” in [Adobe Flash Media Interactive Server Plug-in Developer Guide](#).
- Use the File plug-in to create a custom file I/O system. See “Developing a File plug-in” in [Adobe Flash Media Interactive Server Plug-in Developer Guide](#).

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. For more information, see [Adobe Flash Media Server Developer Guide](#).

Note: Application developers can also use the Administration Console to debug applications.

Command-line tools

Adobe provides two command-line tools to help you administer the server: `FMSCheck` and `FLVCheck`. `FMSCheck` provides information about whether the server is running or not, what the response time is, and which `fmscore` processes are not responding. `FLVCheck` lets you verify that a video file will run properly on Flash Media Server. The `FLVCheck` tool supports MP4 files and FLV files.

Log files

Flash 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.

Flash 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).

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

httpaccess.xx.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`.

Chapter 3: Security

All editions of Adobe Flash 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, 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.

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. Flash Media Server uses true streaming, not progressive download. This means that media streamed from Flash 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—Flash Media Server always uses true streaming technology to protect your content.

Secure network protocols

Flash 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:

Encrypted RTMP (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")
```

Note: Only Flash Player 9 Update 3 and AIR clients can make RTMPE connections.

Secure Sockets Layer (SSL) A protocol for enabling secure communications over TCP/IP. Flash 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")
```

For information about connecting to the server, see the [Adobe Flash Media Server Developer Guide](#). For information about configuring SSL, see [Adobe Flash Media Server Configuration and Administration Guide](#).

Protecting content

Using Adobe Flash Media Rights Management Server

Flash Media Server can stream content encrypted by Flash Media Rights Management Server to AIR applications, including Adobe Media Player. It can stream both encrypted FLV files and encrypted MP4/F4V files. You can stream encrypted files using all supported protocols. Flash Media Rights Management Server supports recorded streams (not live streams).

No specific configuration is required to configure Flash Media Server to work with Flash Media Rights Management Server.

For more information on rights management, see the [Flash Media Rights Management Server documentation](#).

Verifying clients

Flash 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 Flash 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” in [Adobe Flash Media Server Configuration and Administration Guide](#).

- Allow and deny connections from specified domains.

For more information, see “Restrict which domains can connect to a virtual host” in [Adobe Flash Media Server Configuration and Administration Guide](#).

Flash Media Streaming Server does not support the following features:

- Use the File plug-in to verify SWF files. See “Verify SWF files with the File plug-in” in [Adobe Flash Media Server Configuration and Administration Guide](#).
- 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, which 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 Flash Player version.

For more information about these features, see “Authenticate clients” in [Adobe Flash Media Server Developer Guide](#).

Controlling server access

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

Note: The following are not supported by Flash Media Streaming Server.

- 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. For more information, see [Adobe Flash Media Server Developer Guide](#).

- 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. For more information, see [Adobe Flash Media Server Developer Guide](#).
- Use the Access plug-in to accept, reject, or redirect connections before they reach the server-scripting layer. For more information, see [Adobe Flash Media Interactive Server Plug-in Developer Guide](#).
- Use the Authorization plug-in to authorize access to events on the server such as playing and publishing streams. For more information, see [Adobe Flash Media Interactive Server Plug-in Developer Guide](#).

Note: The following feature is supported by all server editions.

- Configure virtual storage folders for streams, shared objects, and files. For more information, see “Configuring content storage” in [Adobe Flash Media Server Configuration and Administration Guide](#).

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 [Adobe Flash Media Server Developer Guide](#).

Configuring the server securely

Securing the Administration Console

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

- Define administrator users and passwords. See “Managing administrators” in [Adobe Flash Media Server Configuration and Administration Guide](#).
- Define IP address ranges or domain names required or denied to administrator users. See “Admin” in [Adobe Flash Media Server Configuration and Administration Guide](#).
- Define a connection port for administrators, thus allowing access only to users behind a firewall. See “Managing administrators” in [Adobe Flash Media Server Configuration and Administration Guide](#).

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 the server” in [Adobe Flash Media Server Configuration and Administration Guide](#).

Security for server-side scripts

Limiting script memory usage

In the `JSEngine` 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” in [Adobe Flash Media Server Configuration and Administration Guide](#). For more information about configuring the JavaScript engine, see “JSEngine” in [Adobe Flash Media Server Configuration and Administration Guide](#).

Loading a script securely

The Flash Media Interactive 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 ActionScript 2.0 Language Reference for Adobe Flash Media Server](#).

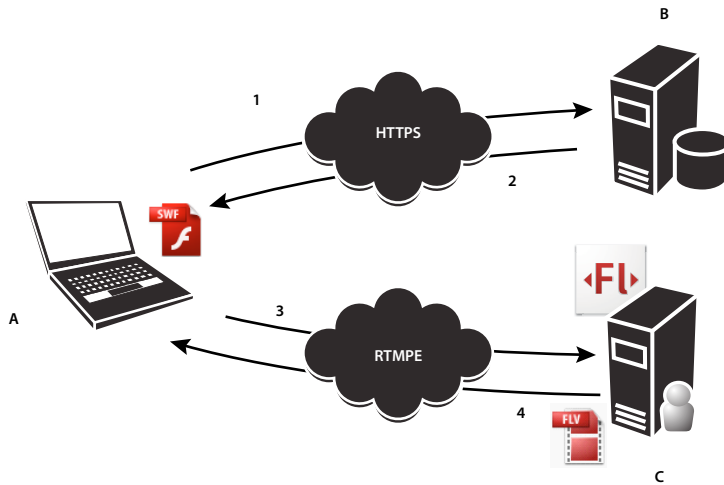
Secure application scenario

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 Flash Media Server over RTMPE and Flash Media Server streams live or recorded content. The stream is encrypted during transmission.



This secure video application uses an application server to authenticate clients before allowing them to request content from the server.
A. Flash Player or AIR client **B.** Application server with database **C.** Flash Media Server **1.** HTTPS authentication request **2.** Authentication response **3.** RTMPE request for content triggers SWF verification **4.** Content streams to client

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.

Index

Numerics

3G2 file format 12
3GP file format 12

A

AAC codec 11
Access plug-in 4, 15, 22
access.log file 19
ActionScript 8, 12, 13, 14
Adaptor.xml file 17
adaptors 17
admin.log file 19
Administration API 15, 16
Administration Console 18, 22
Adobe AfterEffects 12
Adobe Flash 13
Adobe Flex 13
Adobe Premiere 12
ADPCM codec 11
advertising, inserting in a stream 6
AIR (Adobe Integrated Runtime) 8, 11, 21
Apache HTTP Server 3, 4, 8, 9, 10
Application class 14
application server 9
application.log file 19
Application.xml file 17, 23
applications
 about 9
 instances 13
authentication 22
Authorization plug-in 3, 15, 22

B

bandwidth 22
black-listing domains 21

C

C++ plug-ins. *See* plug-ins
Camera class 13
Client class 13, 14
Client.audioSampleAccess property 22
Client.call() method 13
Client.ip property 21
Client.readAccess property 21
Client.referrers property 21

Client.videoSampleAccess property 22
Client.writeAccess property 21
clients 8, 21
client-side ActionScript. *See* ActionScript
clustering 16
codecs 11
configuration 17
connections
 about 8
 controlling access 15
 domains, allowing and denying from 21
 edge servers 16
 external sources 13
 RTMP 10
 secure 20
 SSL 20
content 18
Content Delivery Network (CDN) 6
core processes 4
core.log file 19

D

data model 12
deployment
 clusters 16
 edge servers 16
dynamic streaming 3

E

ECMAScript 14
edge servers 16
edge.log file 19
editions, of Flash Media Server 2
external sources, connecting to 13

F

F4V file format 12
F4V recording 4
File class 14
file formats 11
File plug-in 3, 15
Flash Lite 8
Flash Media Development Server 2, 12, 13, 14
Flash Media Interactive Server 1, 6, 12, 13, 14

Flash Media Live Encoder 5, 10
Flash Media Rights Management Server 21
Flash Media Streaming Server 1
Flash Player 8, 11, 21
FLV file format 11
FLVCheck tool 18
FLVPlayback component 5
fms.ini file 17
FMSCheck tool 18

G

garbage collection 22

H

H.264 codec 11
HTTP 8
HTTP server 3
httpcache.log file 19

J

JavaScript 14

L

Linux Red Hat 5.2 5
live video 5
load balancing 16
LoadVars class 14
Log class 14
Logger.xml file 17
logging 19

M

M4A file format 12
master.log file 19
metadata 4, 12
Microphone class 13
MOV file format 12
Mozilla SpiderMonkey 14
MP3 codec 11
MP4 file format 12
MP4 recording 4
MPEG-4 standard 11
multipoint publishing 5

N

Nellymoser codec 11
 NetConnection class 13, 14
 NetConnection.call() method 13
 NetStream class 13, 14

O

On2 VP6 codec 11
 operating systems (supported) 5

P

performance 18, 22
 permissions 15
 plug-ins
 Access plug-in 15, 22
 Authorization plug-in 15, 22
 File plug-in 3, 15
 ports 10
 protocols
 HTTP 8
 RTMP 8, 10
 RTMPE 10, 20
 RTMPS 10
 RTMPT 10
 RTMPTE 10
 secure 20
 SSL 20
 TCP/IP 8
 publishing
 about 10
 multipoint 5

R

recording media 10
 remote method invocation (RMI) 13
 RTMP 8, 10
 RTMPE 10, 20
 RTMPS 10, 20
 RTMPT 10
 RTMPTE 10

S

script security 23
 security
 protecting content 21
 protocols 20
 sample scenario 23
 scripts 23
 server access, controlling 21
 Server.xml file 17

Server-Side ActionScript 8, 12, 13, 14, 23
 shared objects 12
 SharedObject class 12, 13, 14
 SharedObject.getRemote() method 13
 SOAPCall class 14
 SOAPFault class 14
 social media 6, 10
 Sorenson Spark codec 11
 Speex 4
 SSL 10, 20
 statistics (about client) 3
 storage 18, 22
 Stream class 14
 streaming
 about 10
 progressive 20
 size, limiting 22
 true 20
 SWF files 3, 21
 system requirements 2

T

TCP/IP 8
 tunneling 10

U

Users.xml file 17

V

verifying SWF files 3
 vhost. *See* virtual host
 Video class 13
 video on demand 4, 5
 video players 5
 virtual host 17

W

W3C format 19
 web cam 10
 Webservice class 14
 white-listing domains 21
 Windows Server 2008 5

X

XML 11
 XML class 14
 XML configuration files 17
 XMLSocket class 14
 XMLStreams class 14

XMP 12

XMP metadata 4