

ADOBE® FLASH® MEDIA SERVER

TECHNICAL OVERVIEW

© 2007 Adobe Systems Incorporated. All rights reserved.

Adobe® Flash® Media Server 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, and Flex are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other trademarks are the property of their respective owners.

Portions include software under the following terms:

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

Licensee shall not use the MP3 compressed audio within the Software for real time broadcasting (terrestrial, satellite, cable or other media), or broadcasting via Internet or other networks, such as but not limited to intranets, etc., or in pay-audio or audio on demand applications to any non-PC device (i.e., mobile phones or set-top boxes). Licensee acknowledges that use of the Software for non-PC devices, as described herein, may require the payment of licensing royalties or other amounts to third parties who may hold intellectual property rights related to the MP3 technology and that Adobe has not paid any royalties or other amounts on account of third party intellectual property rights for such use. If Licensee requires an MP3 decoder for such non-PC use, Licensee is responsible for obtaining the necessary MP3 technology license.

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	3
Common uses for the server	6

Chapter 2: Architecture

Core server architecture	9
Language libraries	13
Scaling the server	15
Configuring and administering the server	17

Chapter 3: Security

Streaming content securely	19
Protecting content	19
Configuring the server securely	21
Security for server-side scripts	21
Secure application scenario	22

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: 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 social media applications such as video recorders and multiplayer games for Flash Player, AIR, and Flash Lite. The server offers an SDK that lets you develop media applications in Server-Side 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 social media 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 is also designed for real-time media applications, in which many 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 version 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 configuration	Yes	No	Yes
Custom C++ plug-in support	Yes	No	Yes
Core server processes	Unlimited	1	Unlimited
Encrypted streaming (RTMP)	Yes	Yes	Yes
Simple access control (SWF verification)	Yes	Yes	Yes
Adobe Media Player tracking service	Yes	No	No

System requirements

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

What's new

Built-in streaming services

Video on demand service

Place a video into the *RootInstall/applications/vod/media* folder and you can stream video immediately. Use the Adobe Flash CS3 Professional and Flash 8 FLVPlayback component with the vod service to stream video to Flash Player 8 and Flash Player 9 without writing any code. You can also write your own client applications for the vod service in ActionScript 2.0 or ActionScript 3.0. To provision customers, the application can be renamed and duplicated on a server an unlimited number of times. For more information, see “The vod service” in [Adobe Flash Media Server Developer Guide](#).

Live publishing point

Use Adobe® Flash® Media Encoder and the live service to capture and stream live video to Flash Player 8 and Flash Player 9 without writing any code. You can also write your own client applications for the live service in ActionScript 2.0 or ActionScript 3.0. To create multiple publishing points, the application can be renamed and duplicated on a server an unlimited number of times. For more information, see “The live service” in the [Developer Guide](#).

Live publishing enhancements

Data keyframes

Publishers can encode metadata into live streams. All recipients receive the metadata when they subscribe to the live stream. This prevents latecomers from missing important information about a video or event. For more information, see “Add metadata to a live stream” in the [Developer Guide](#).

Multipoint publish

Use Flash Media Interactive Server or Flash Media Development Server to push live streams to additional origin servers that distribute the streams to users. This is useful for streaming data to a Content Delivery Network capable of delivering your stream to millions of users. For more information, see “Publish from server to server” in the [Developer Guide](#).

Security enhancements

Encrypted RTMP (RTMPE)

An 128-bit encrypted edition of Real-Time Messaging Protocol (RTMP). This protocol is more lightweight than SSL but still provides a high level of security. For more information, see [Real-Time Messaging Protocol](#).

Verify SWF files

Verify the authenticity of SWF files before they can connect to any server resources. This guarantees that the only clients that connect to your application or service are clients you created or authorized. For more information, see “Verify SWF files” on page 21 in *Adobe Flash Media Server Configuration and Administration Guide*.

Access stream data

The read/write ACLs (access control list) have been extended with two new permission types to control access to the audio and video data in streams. Applications can extract video frames as bitmaps if allowed by the server. The Server-Side ActionScript API and the Authorization plug-in API have been updated to support this feature. For more information, see the *Developer Guide*.

Performance improvements

Published performance benchmarks

Load testing statistics are published at Adobe.com. The statistics are derived from careful testing and reflect real-world scenarios, including random connection and streaming patterns.

Smart pausing

Flash Player 9 Update 3 no longer clears the buffer when a stream is paused. This allows viewers to resume playback without experiencing any hesitation. Developers can also use `NetStream.pause()` in code to buffer data while viewers are watching a commercial, for example, and then unpause the stream when the main video starts. For more information, see the `NetStream.pause()` entry in *Adobe Flash Media Server ActionScript 2.0 Language Reference* or in *ActionScript 3.0 Language and Components Reference*.

Distribute applications over multiple processes

Note: This feature is available on Flash Media Interactive Server and Flash Media Development Server only.

Specify the scope at which application instances are assigned to server processes and select any number of server processes over which to distribute those instances. Possible scopes include adaptors, virtual hosts, applications, instances, and connections. The ability to fine-tune process distribution lets you get maximum performance from the server for different types of applications. For more information, see “Configure how applications are assigned to server processes” on page 18 in the *Configuration and Administration Guide*.

Limit connection requests

Configure the server to limit surges of incoming connections to maintain server performance. This prevents a sudden, large event, such as a popular news story, from disrupting the playback of video for users who are already connected. For more information, see “Limit connection requests” on page 16 in the *Configuration and Administration Guide*.

Close idle connections

Configure the server to disconnect long-standing idle connections to free core processes to run additional applications. For more information, see “Close idle connections” on page 16 in the *Configuration and Administration Guide*.

Native bandwidth detection

Server-to-client bandwidth detection now occurs in the native server code instead of in a Server-Side ActionScript script. Native bandwidth uses less memory and less CPU power, which provides better performance and better scalability. In addition, bandwidth is now measured between the edge server and the client for clients connecting through an edge server, rather than between the client and the origin. For more information, see “Configure or disable native bandwidth detection” on page 30 in the *Configuration and Administration Guide*.

Configure optimal memory use for the recorded media cache

Set a limit on the amount of memory used by the recorded media cache. The cache also uses a new replacement policy that increases the likelihood of keeping the most requested data segments in the cache. For more information, see “Configure the recorded media cache” on page 14 in the *Configuration and Administration Guide*.

Plug-in support

File plug-in

Note: This feature is available on Flash Media Interactive Server and Flash Media Development Server only.

The File plug-in C++ API lets you customize the way the server handles file access. For example, you could create a remote content management solution that makes HTTP requests to access the bytes needed to stream video. For more information, see [Plug-in API](#).

Authorization plug-in

Note: This feature is available on Flash Media Interactive Server and Flash Media Development Server only.

The Authorization plug-in C++ API gives you access to server events and lets you authorize or not authorize certain events, such as playing or seeking through a file. You can also disconnect clients and dispatch an arbitrary message to a server-side script. This provides a powerful bridge between plug-ins and the Server-Side ActionScript API that makes Authorization plug-ins extensible. For more information, see [Plug-in API](#).

Access plug-in

Note: This feature is available on Flash Media Interactive Server and Flash Media Development Server only.

The Access plug-in C++ API lets you control access to the server. The Access plug-in adds another layer of security to the server; it intercepts connection requests and lets you examine the client and the server to determine whether requests should be accepted, rejected, or redirected before the requests reach the server's script layer. For more information, see [Plug-in API](#).

Administration tools

FMSCheck Tool

Performs a deep check to determine whether streaming is functioning correctly. For example, you can check whether a stream can be played, whether a stream can be published, and whether application-level connections are being accepted. For more information, see “Checking server status” on page 65 in the *Configuration and Administration Guide*.

FLVCheck Tool

Verifies that files encoded in third-party, non-Adobe technology can be streamed on the server without error. For more information, see “Checking video files” on page 67 in the *Configuration and Administration Guide*.

Simplified configuration

The most commonly used configuration parameters have been moved from the XML files to the fms.ini file. This lets you configure the server from one location. For more information, see [Configuring the server](#).

Platform and standards compliance

H.264 video and HE-AAC audio support

Flash Player 9 Update 3 and AIR support video and audio encoded in H.264 and HE-AAC from within MPEG-4 standard file formats. These formats stream high quality video at lower bit rates. Developers can leverage industry standard tools, including Adobe Premiere Pro and Adobe After Effects, to create and deliver compelling video content. All editions of Flash Media Server 3 can stream H.264 and HE-AAC content to Flash Player 9 Update 3 and AIR. For more information, see [Streaming media](#).

302-Redirect Connection

Note: This feature is available on Flash Media Interactive Server and Flash Media Development Server only.

The `NetConnection.Connect.Reject` status code has been enhanced to convey redirect information in a standard format. The Access plug-in API, Authorization plug-in API, and Server-Side ActionScript API have been updated to support redirection. For more information, see the *Plug-in API Reference* and the *Server-Side ActionScript Language Reference*.

AMF3 support

All server editions support Action Message Format (AMF) 3 for data serialization introduced in Flash Player 9 and ActionScript 3.0. For more information, see “About the Action Message Format (AMF)” in the *Developer Guide*.

IPv6 compliance

All editions of Flash Media Server 3 adhere to the IPv6 standard, a next-generation standard that extends the addressing capability of the Internet Protocol. For more information, see “Configuring IPv6” on page 28 in the *Configuration and Administration Guide*.

Common uses for the server

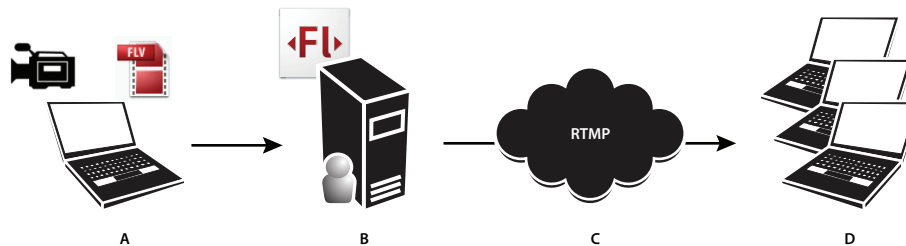
Video players

Video players stream live or recorded video from Flash Media Server to users. Recorded video is called *video on demand*. Video that is being captured and streamed live is called *live video*. To deliver recorded video on demand, Adobe provides a video player called the FLVPlayback component; you can also use ActionScript to develop your own client video player. Adobe Flash Media Encoder lets you capture audio and video while streaming it to Flash Media Server. You can also use ActionScript to build a custom Flash Player application that captures audio and video.

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

- Short video clips, such as commercials up to 30 seconds
- Longer clips (often called “long tail” content), such as user-generated videos up to 30 minutes
- Very long clips, such as recorded television shows or movies up to several hours long

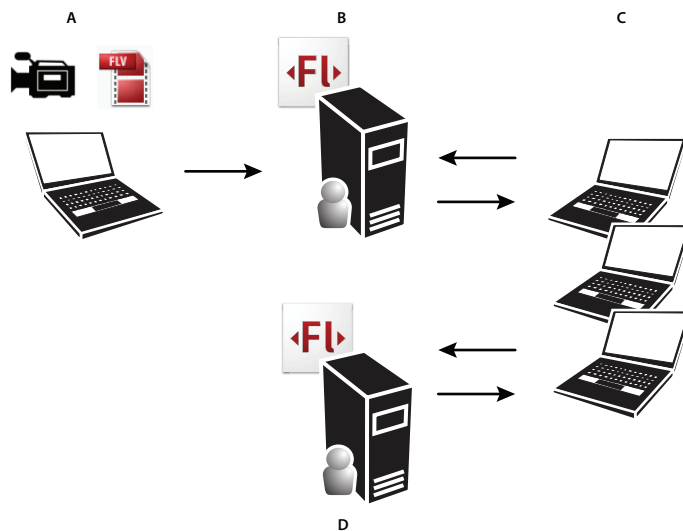
- Video playlists can play a list of streams in a sequence, whether live streams, recorded streams, or a mix. The playlist can be in a client-side script or, on Flash Media Interactive Server, in a server-side script.



A. Flash Media Encoder (or custom-built Flash Player or AIR solutions) capture live audio and video and publish it to the server.
B. Flash Media Server streams live and recorded media to clients. C. Internet (RTMP) D. Flash Player, AIR, or Flash Lite run the video players

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)

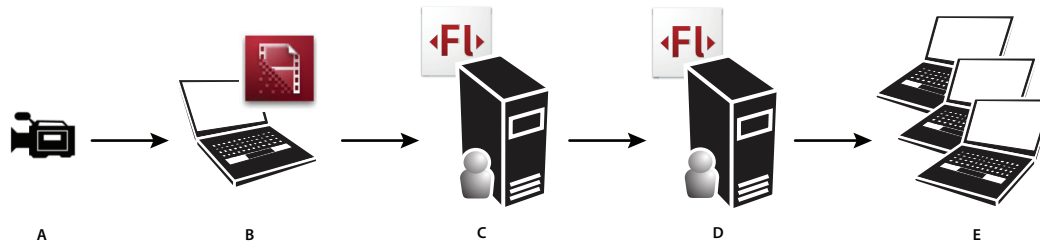
Live video with multipoint publishing

Note: This feature is supported by Flash Media Interactive Server and Flash Media Development Server.

Whether live audio and video is captured by a webcam or by a sophisticated corporate video filming group, it can be converted to video and streamed live using the Adobe Flash Media Encoder, available from http://www.adobe.com/go/learn_fms_fme_en.

Note: You can also create your own Flash Player or AIR solutions to capture and stream live video.

When you need to broadcast media to a large number of viewers, you can use multipoint publishing to stream the live video from a camera to a publishing server, and then to a broadcast server (which is often a Content Delivery Network).

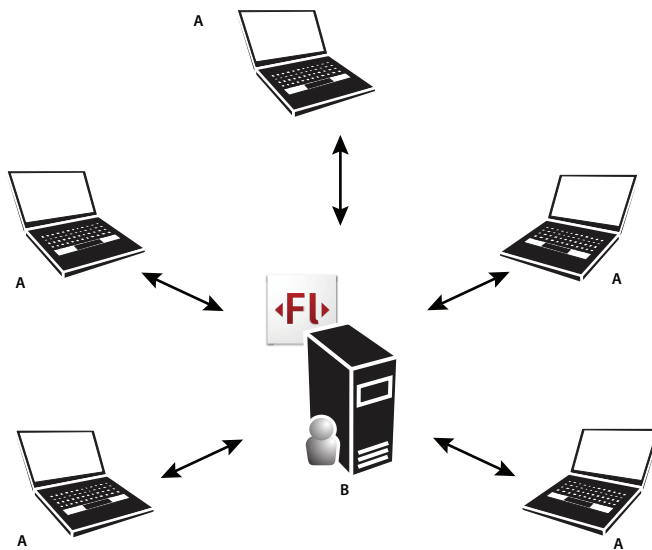


A. Live video B. Flash Media Encoder (or custom-build Flash Player or AIR solutions) C. Flash Media Server (publishing) D. Flash Media Server (broadcasting) E. Flash Player, AIR, or Flash Lite clients

Multipoint publishing can be used to build large-scale live video applications or 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, which would publish the stream to a larger Flash Media Interactive Server deployment, such as a CDN that pushes the stream to millions of users.

Social media 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 text messages. B. Flash Media Interactive Server broadcasts the media and data to all connected users.

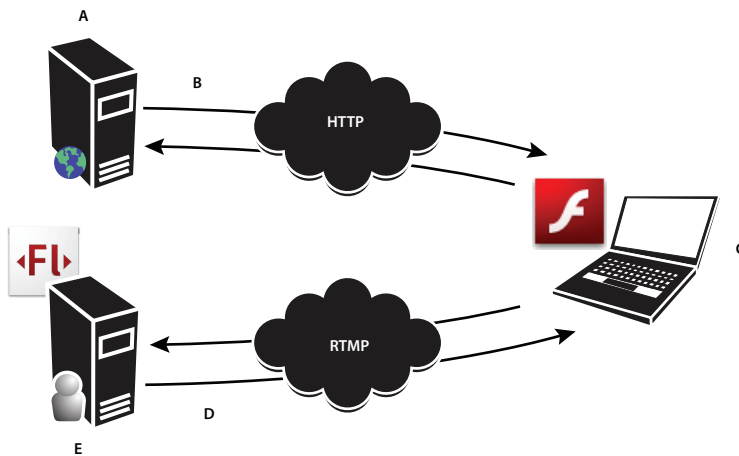
Chapter 2: Architecture

Core server architecture

Client-server architecture

Adobe Flash Media Server applications have a client-server architecture. The client code is written in ActionScript and runs in Adobe Flash Player, Adobe AIR, or Adobe Flash Lite. The server code is written in Server-Side ActionScript, which is similar to 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 applications consist of client and server components that work together.

A. Web 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. E. Flash Media Server

Real-time media server

Flash Media Streaming Server provides two streaming services: live and video on demand (vod). 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. Flash Media Interactive Server and Flash Media Development Server also offer an SDK that lets developers write plug-ins in C++ to extend the core functionality of the server.

Hosting multiple applications and application instances

Flash Media Interactive Server and Flash Media Development Server can host an unlimited number of applications and an unlimited number of instances of each application. 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 applications 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, as well as the live and vod services. You can create multiple instances of each of those applications. For example, use instances to give groups of people access to the same application without having the groups interact with each other, such as a video chat application with rooms for different topics.

Applications must be registered with Flash Media Server so the server knows who they are when they try to connect. To register an application with the server, a developer creates a folder for the application in the main applications folder, which is located in the root installation folder by default (Flash Media Server 3/applications). To create instances of an application, a developer creates subfolders within that application's folder (for example, Flash Media Server 3/applications/exampleApplication/instance1). For more information, see the [Developer Guide](#).

The server administrator can change the location of the 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](#).

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.

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 [Flash Media 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.

To publish a stream, a client calls `NetStream.publish()`. To subscribe to a stream, a client calls `NetStream.play()`. (These calls can be made in ActionScript 2.0 or ActionScript 3.0.) To publish a stream to a client, a server calls `Stream.play()`. To publish a stream to another server, a server calls `NetStream.publish()`. (These calls are made in Server-Side ActionScript.)

Note: *When Flash Media Encoder 2 is the publisher, you don't have to write any code. Flash Media Encoder 2 is configured to publish a live stream to Flash Media Server 3 by default.*

Supported codecs

Flash Media Server doesn't encode or decode audio and video information; it streams media that has already been encoded. To encode and stream 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 Encoder 2 or use ActionScript to build your own Flash Player or AIR client. Flash Media Encoder 2 encodes live video with the On2 VP6 codec. Flash Player and AIR encode live video with the Sorenson Spark codec and encode live audio with a proprietary Nellymoser codec. If you want to encode live video with On2 VP6, you must use Flash Media Encoder 2.

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	6	8, Flash Lite 3
H.264 (MPEG-4 Part 10)	9	9 Update 3, AIR
ADPCM	6	6, Flash Lite 3
MP3	6	6, Flash Lite 3
AAC (MPEG-4 Part 3)	9	9 Update 3, AIR

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

Flash Player 9 Update 3 and AIR support playback of the following subsets of the MPEG-4 standards:

MPEG-4 standard	Flash Player 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 3 support playback of the following file formats:

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

To create FLV files from existing digital video or audio files, use third-party video encoding utilities, like On2 or Sorenson Squeeze, or export them from Flash. Recorded streams can contain data messages, as well as video and audio and are stored in the FLV file format.

F4V, MP4, M4A, MOV, MP4V, 3GP, and 3G2 Flash Media Server 3 supports playback of H.264-encoded video and AAC-encoded audio within MPEG-4 Part 14 container formats.

You can use any number of industry-standard tools to create files in these formats with these types of encoding, including Adobe Premiere and Adobe AfterEffects.

Important: If a file contains portions of audio and video not encoded using H.264 and AAC, those parts of the stream will not play. For more information on H.264/AAC support, see [Exploring Flash Player support for high-definition H.264 video and AAC audio](#).

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 the [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 [Server-Side ActionScript Language Reference for Adobe Flash Media Server](#), [ActionScript 2.0 Language Reference](#), and [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 the [Developer Guide](#) and the [Server-Side ActionScript Language Reference](#).

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

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 plug-ins written in C++ that let you 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 intercepts and examines each connection request (before the request reaches the server's script layer) to determine whether the server should accept or reject the request based on server statistics, such as the number of current connections. You can also code the Access plug-in to set permissions to server resources such as streams and shared objects.

Authorization plug-in Authorize client access to events that occur on the server. The Authorization plug-in authorizes connections to the server, playing of streams, and publishing of streams. It also maps logical URLs to physical locations. The Authorization plug-in can be configured to allocate the delivery of content for Flash Media Interactive Server clients according to geographic location, level of subscription, origin of the Flash Media Interactive Server stream, or time and duration of a user's access to specific streams.

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 server's 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.

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](#).

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” on page 4 in the *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. For more information, see the [Developer Guide](#).

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 *edge server*. 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 edge servers” on page 5 in the *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 and administering the server

Configuring 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`. There are also configuration files for information about administrators and logging: `Users.xml` and `Logger.xml`. The most important configuration parameters have been pulled out to the `fms.ini` file, which 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 “Configuring adaptors, virtual hosts, and applications” on page 8 in the *Configuration and Administration Guide*.

Administering the server

Each edition of the server installs with an additional server called Flash Media Administration Server. This server has an Administration API that queries and configures Flash Media Server. Developers can use the Administration API to create their own server administration tools.

For more information, see “Working with the Administration API” on page 74 in the *Configuration and Administration Guide*.

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 “Using the Administration Console” on page 38 in the *Configuration and Administration 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:

access.xx.log Tracks information about server access; for example, the date and time a client connected to the server, the total bandwidth consumed during a session, the streams accessed by the connection, whether the client published a stream, and 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, the event's server process ID, the event status level (warning, error, information, debug, and so on).

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 [Developer Guide](#). For information about configuring SSL, see “Configure SSL” on page 24 in the *Configuration and Administration Guide*.

Protecting content

Verifying clients

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

- Verify SWF files.

This technique ensures that only SWF files you created can access this server. This prevents third parties from creating their own SWF files that attempt to stream your resources. You can configure Flash Media Server to verify client SWF files before allowing them to connect to an application. For more information, see “Verify SWF files” on page 21 in the *Configuration and Administration Guide*.

- Allow and deny connections from specified domains.

For more information, see “Restrict which domains can connect to a virtual host” on page 21 in the *Configuration and Administration Guide*.

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

To prevent unauthorized clients from sending streams to the server or from playing streams they aren’t authorized to access, Flash Media Server supports the following security options:

- 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 the *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 the *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 the *Developer Guide*.
- Use the Access plug-in to accept, reject, or redirect connections before they reach the server-scripting layer. For more information, see the *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 the *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” on page 31 in the *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 the *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” on page 48 in the *Configuration and Administration Guide*.
- Define IP address ranges or domain names required or denied to administrator users. See Admin in the *Configuration and Administration Guide*.
- Define a connection port for administrators, thus allowing access only to users behind a firewall. See “Managing administrators” on page 48 in the *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” on page 8 in the *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” on page 11 in the *Configuration and Administration Guide*. For more information about configuring the JavaScript engine, see `JSEngine` in the *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 [Server-Side ActionScript Language Reference for Adobe Flash Media Server](#).

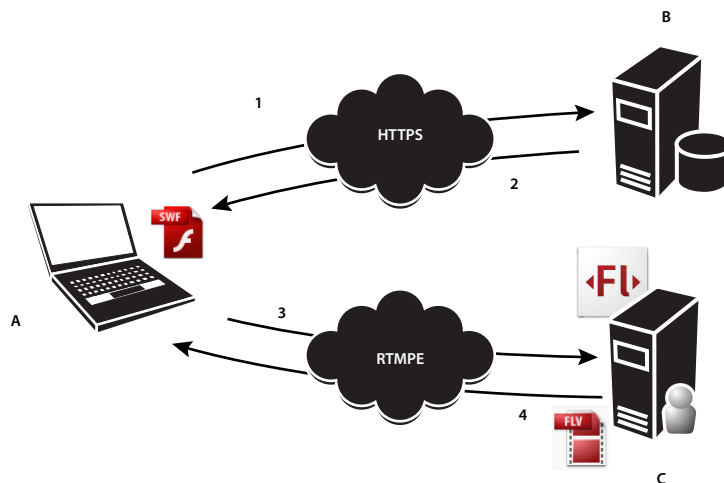
Secure application scenario

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

302-redirect 6
3G2 file format 12
3GP file format 12

A

AAC codec 11
Access plug-in 15, 20
access.log file 18
Action Message Format (AMF) 6
ActionScript 9, 12, 13
ActionScript 1.0 9, 14
Adaptor.xml file 17
adaptors 17
admin.log file 18
administration 17
 Administration API 15
 Administration Console 17, 21
 Flash Media Administration Server 17
 FLVCheck tool 5
 FMSCheck tool 5
Administration API 15
Administration Console 17, 21
Adobe AfterEffects 12
Adobe Flash 13
Adobe Flex 13
Adobe Premiere 12
ADPCM codec 11
advertising, inserting in a stream 7
AIR (Adobe Integrated Runtime) 9, 11
Application class 14
application server 9
application.log file 18
Application.xml file 17, 21
applications
 about 10
 instances 13
authentication 20
Authorization plug-in 15, 20

B

bandwidth 21
black-listing domains 20

C

C++ plug-ins. *See* plug-ins
Camera class 13
Client class 13, 14
Client.audioSampleAccess property 20
Client.call() method 13
Client.ip property 20
Client.readAccess property 20
Client.referrer property 20
Client.videoSampleAccess property 20
Client.writeAccess property 20
clients 9
 verifying 20
client-server architecture 9
client-side ActionScript. *See* ActionScript
clustering 15
codecs
 AAC 11
 ADPCM 11
 H.264 11
 MP3 11
 Nellymoser 11
 On2 VP6 11
 Sorenson Spark 11
configuration 17
connections
 about 9
 controlling access 15
 domains, allowing and denying from 20
 edge servers 16
 external sources 13
 RTMP 10
 secure 19
 SSL 19
Content Delivery Network (CDN) 7
core.log file 18

D

data model 12
deployment
 clusters 15

edge servers 16

E

ECMAScript 14
edge servers 16
edge.log file 18
editions
 comparison 2
 Flash Media Development Server 2
 Flash Media Interactive Server 1
 Flash Media Streaming Server 1
external sources, connecting to 13

F

F4V file format 12
File class 14
file formats
 3G2 12
 3GP 12
 F4V 12
 FLV 12
 MOV 12
 MP4 12
 MP4V 12
File plug-in 15
Flash Lite 9
Flash Media Administration Server 17
Flash Media Development Server 2, 12, 13, 14
Flash Media Encoder 7, 10
Flash Media Interactive Server 1, 8, 12, 13, 14
Flash Media Streaming Server 1
Flash Player 9, 11
 version, verifying 20
FLV file format 12
FLVCheck tool 18
FLVPlayback component 6
fms.ini file 17
FMSCheck tool 18

G

garbage collection 21

H

H.264 codec 11
httpcache.log file 18

I

IPv6 compliance 6

J

JavaScript 14

L

live service 3
live video 3, 6, 7
load balancing 16
LoadVars class 14
Log class 14
Logger.xml file 17
logging 18

M

M4A file format 12
master.log file 18
metadata 3
Microphone class 13
MOV file format 12
Mozilla SpiderMonkey 14
MP3 codec 11
MP4 file format 12
MPEG-4 standard 11
multipoint publishing 7

N

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

O

On2 VP6 codec 11

P

performance 4, 21
permissions 15
plug-ins
 Access plug-in 15, 20
 Authorization plug-in 15, 20
 File plug-in 15
ports 10
protocols

HTTP 9

RTMP 9, 10

RTMPE 10, 19

RTMPS 10

RTMPT 10

RTMPTE 10

secure 19

SSL 19

TCP/IP 9

publishing

 about 10

 live video 3

 multipoint 3, 7

 video on demand 3

R

recording media 11
redirect 6
remote method invocation (RMI) 13
RTMP (Real-Time Messaging Protocol) 9
RTMPE 19
RTMPS 19

S

security
 new features 3
 protecting content 20
 protocols 19
 sample scenario 22
 server access, controlling 20
Server.xml file 17
Server-Side ActionScript 9, 12, 13, 14, 21
shared objects 12
SharedObject class 12, 13, 14
SharedObject.getRemote() method 13
SOAPCall class 14
SOAPFault class 14
social media 8, 11
Sorenson Spark codec 11
SSL 10, 19
standards compliance 6
storage 20
Stream class 14
streaming
 about 10
 live media 3
 progressive 19

 size, limiting 21

 true 19

SWF files, verifying 20

system requirements 2

T

TCP/IP 9

tunneling 10

U

Users.xml file 17

V

vhost. *See* virtual host

Video class 13

video on demand 3, 6

video players 6

virtual host 17

vod service 3

W

W3C format 18

web cam 11

WebService class 14

white-listing domains 20

X

XML 11

XML class 14

XML configuration files 17

XMLSocket class 14

XMLStreams class 14