Planning for Large-Scale Streaming Deployments with Adobe Flash Media Interactive Server 3.5

This document covers the basic principles for deploying Flash Media Server 3.5. Deployment options for dynamic streaming and DVR functionality will be covered in a future version of this document.

Introduction to Large-Scale Deployment for Media Streaming

Flash Media Interactive Server (FMIS) can be deployed in large scale to meet high-demand streaming to the Adobe Flash platform. A large-scale deployment can consist of multiple servers linked together to increase capacity and quality of service, meaning less interruption in the media flow and an improved user experience.

Efficient large-scale deployments should take into account issues such as datacenter choices (local or remote), bandwidth issues, storage and cache management, authentication and security, live streaming considerations, and troubleshooting. This document will address these issues, and will provide information needed to help make the right choices for your network.

Large-scale deployments serve multiple users with multiple tenants, possibly with numerous deployments across multiple data centers.

Flash Media Server offers flexible solutions for large-scale deployment, with two configurations available. Flash Media Interactive Server can be set up in a native Edge and Origin configuration or an Origin-Only Cluster. This technical paper will discuss the high-level benefits of each.

Understanding Live vs. VOD Deployment

To ensure efficiency of large-scale deployments, the type of content being delivered must be considered. Live (“stateful”) content and video on demand (“stateless”) content need to be handled differently, as they each have a different effect on cache and disk storage, and require a different level of load balancing intelligence.

Video on demand (VOD) streams can be load balanced using standard DNS or GEO load balancing. DNS load balancing directs requests to a server closest to the location of the consumer’s DNS server. GEO load balancing directs requests to a server closest to the consumer’s actual IP address.

Because live streams originate from a single source, they will require more load balancing intelligence, often relying on server-side ActionScript to set up proxies. In this scenario, a group of primary and backup servers will need to be defined as an ingest point for live streams from Flash Media Live Encoder (FMLE) or third-party live encoders. This stream would then be proxied to the Edge servers using ActionScript.

It is also good practice to leverage your stand-by capacity, allowing for your network to scale up for large live events while using the idle capacity for VOD when not in use for live streams.

Also, when deploying live streams, it is important that access controls be set to allow FMLE and other encoders to publish to the server, while also setting permissions to protect against denial-of-service (DoS) attacks and ingest overload.

Live streams will require less storage space than VOD. A possible exception to this rule could be the use of the DVR functionality in FMS 3.5. DVR requires that the entire live stream be archived to allow navigation of the stream, so ample disk space will be required. Generally, however, live streams require little or no hard drive storage.
Cache handling is also different for live and VOD content, and is critical to network efficiency, server load, and overall quality of service. With VOD, server and network load is compounded with the use of multi-bitrate streaming. For example, if someone is watching a stream, all of the bit rate versions of that stream need to be cached to ensure smooth stream switching if his bandwidth changes during playback. Because this would be too much to store in RAM, intelligent, content-aware caching needs to be implemented. Also, configuring the segment cache will enable you to manage what parts of the video are available on the Edge. For more information about setting up and managing cache, refer to the Storage and Cache section of this white paper.

Protocol support

Flash Media Server communicates with connected clients using Adobe’s Real-Time Messaging Protocol (RTMP) over Transmission Control Protocol (TCP). The RTMP connection manages two-way communication and transmission of video, audio and data. There are five configurations of RTMP with Flash Media Server 3.5, as shown in the following table.

<table>
<thead>
<tr>
<th>NAME</th>
<th>PORT</th>
<th>ENCRYPTED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTMP</td>
<td>1935</td>
<td>No</td>
<td>Standard RTMP</td>
</tr>
<tr>
<td>HTTP</td>
<td>80</td>
<td>No</td>
<td>(Optional) if Apache web server is installed, FMS will proxy port 80 requests to either HTTP or RTMPT based on the protocol.</td>
</tr>
<tr>
<td>RTMPT</td>
<td>80</td>
<td>No</td>
<td>HTTP, with encapsulated RTMP packets</td>
</tr>
<tr>
<td>RTMPS</td>
<td>443</td>
<td>Yes</td>
<td>Secure RTMP over SSL, requires certificate management</td>
</tr>
<tr>
<td>RTMPTS</td>
<td>80</td>
<td>Yes</td>
<td>Secure RTMP over SSL, tunneled within HTTP</td>
</tr>
<tr>
<td>RTMPE</td>
<td>1935</td>
<td>Yes</td>
<td>High-performance Encrypted RTMP</td>
</tr>
<tr>
<td>RTMPE</td>
<td>80</td>
<td>Yes</td>
<td>Encrypted RTMP tunneled within HTTP</td>
</tr>
</tbody>
</table>

The standard RTMP protocol will yield the best performance.

RTMPT or RTMPE will enable you to reach more people in networks that don’t allow RTMP or block port 1935. However, RTMPT tunneling does reduce the capacity of Flash Media Server, yielding 30% of RTMP (non-tunneled) traffic, and requires more CPU usage.

RTMPE does require additional CPU usage as well, but still saturates a 1 Gbps network interface at less than 70% utilization of the CPU, and is less CPU-intensive than SSL encryption.

Hardware Considerations

Flash Media Server deployments, like any large scale bandwidth-intensive hosting, benefit from fast disk access, plentiful RAM, and fast processor speeds. As seen in the following table, the listed configurations should be sufficient to saturate a 1 Gbps or higher network connection without performance issues.

<table>
<thead>
<tr>
<th>DEPLOYMENT</th>
<th>PROCESSORS</th>
<th>RAM</th>
<th>RAID TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small: one-to-many</td>
<td>1 Dual Core</td>
<td>2 - 4 GB</td>
<td>RAID 1, 5, or 6</td>
</tr>
<tr>
<td>Medium: one-to-many or many-to-many</td>
<td>1 Quad Core</td>
<td>4 - 8 GB</td>
<td>RAID 1, 5, or 6</td>
</tr>
<tr>
<td>Large: one-to-many or many-to-many, average shared object usage, interactive elements</td>
<td>1 - 2 Quad Core</td>
<td>4 - 16 GB</td>
<td>RAID 1, 10, 5, 50, or 6</td>
</tr>
<tr>
<td>Extreme: one-to-many, many-to-many, extreme shared object usage, rich interactive elements</td>
<td>1 - 2 Quad Core</td>
<td>16 - 32 GB</td>
<td>RAID 1, 10, 5, 50, or 6</td>
</tr>
</tbody>
</table>

As a general rule, you’ll want to choose the fastest disks possible (10,000 to 15,000 RPM).

RAM

An abundance of RAM is important because FMS uses it to cache content such as pre-recorded streams, interactive content and data. A lack of RAM will force FMS to use the hard drive to cache content, causing sluggish performance. Flash Media Server stores its cache in RAM or on the hard drive in 256-kilobyte chunks, so the information is quickly accessible. Origins store their cache in RAM, while Edges can use hard drive caching to lighten the load on the Origin. Using the `<CacheDir>` tag in Vhost.xml configuration file, Edges can be configured to cache recorded data streams on their local hard drives, so the streams do not have to be accessed from the Origin every time they are requested.
CPU

CPU speed is also crucial to efficiently handling connections, executing server-side code and managing server operations. An overloaded CPU will cause sluggish performance, including slower connection time and more frequent rebuffering events, and may introduce latency. In general, you can expect an 80% performance increase with each additional CPU/core. Two dual-core machines will perform better than one quad-core machine. A minimum 3.2GHz Intel® Pentium® 4 CPU is recommended.

If RAM or CPU usage regularly exceed 75% capacity, a hardware upgrade should be seriously considered.

When purchasing your own hardware for large FMS deployments, it is recommended that you buy multiple copies of identical hardware configurations. This makes hardware replacement simple in the event of hardware crashes or failures.

While Adobe does not certify specific hardware, if you require a server with an integrated hardware solution, you should consider the CISCO Internet Streamer (Content Delivery Server). This solution has preconfigured hardware optimized for streaming with Flash Media Server.

Network Interfaces

Since FMS 3.5 can saturate a 1 Gbps network interface card with as little as 20% CPU usage,* you may want to consider adding multiple network cards to increase the network capacity of your server. Using multiple network cards could increase capacity beyond 1.4 Gbps per server (measured at 70% utilization/NIC). This is referred to as “bonding” in Linux or “teaming” in Windows. FMS then leverages the hardware resources through the operating system, requiring no special FMS configuration.

Issues to consider when planning network interface upgrades include:

• Your datacenter's network connection must provide enough bandwidth to support your server’s network capacity.

• Hard drive speed limitations could become a bottleneck. Most SATA hard drives can burst to 3 Gbps, but can usually only sustain 1-2 Gbps of throughput. SAS hard drives with fast rotations (10,000 RPM or higher) may help with access times and throughput. Since FMS efficiently caches data in RAM, hard disk speed may not become an issue unless there is a huge amount of I/O activity on the disk, such as with intensive recording and viewing applications, where hundreds of people are recording videos at the same time, and hundreds or thousands are watching the videos.

• When using multiple NIC cards, you amalgamate the total throughput to break the 1 Gbps limit. The sub-100% utilization of each network interface should still be observed in your capacity planning.

• Flash Media Server can be also configured to manage multiple network cards as separate resources. You can provision streaming at the adaptor level, binding delivery of specific media or communications on different network cards. When using this solution, it is important to remember that connections will not be persistent across the server, since data is bound to different network cards. This lack of persistence will affect stateful applications such as real-time data sharing and live streaming.

• High network usage (70% of available network bandwidth) can cause extra overhead on the CPU and memory, and bog down performance, regardless of actual network capacity.

Datacenter/Colocation Facility

You can choose to outsource your IT infrastructure or host it internally.

An in-house deployment requires hands-on management, but if qualified IT administrators are already available, more control can be leveraged over the servers and infrastructure.

Hardware costs, maintenance, management, power, and bandwidth costs should all be considered. In-house bandwidth and power is likely to be less expensive than external datacenters, since the costs are not marked up by a middleman. However, an external datacenter takes care of hardware costs and maintenance, and likely has a larger fixed infrastructure and network capacity.

*Average 300kbps streams running on Linux Red Hat. Refer to the official Adobe Flash Media Server 3.5 white paper at [http://www.adobe.com/go/fms_whitepaper](http://www.adobe.com/go/fms_whitepaper) for more information on network usage.
There are several points to consider when considering an external datacenter or colocation facility:

- **Network reliability.** Who are their bandwidth carriers and what are their reputations? Are they Tier 1, Tier 2? Lower? Note that a carrier does not need to be Tier 1 to offer good performance. Lower tier networks generally pay transit fees to other providers for data transfer, which can result in higher fees for you, but can ultimately offer the same quality as a Tier 1.

- **Failover and security.** Do their datacenters provide redundant backbone connections and power grid connections with failover generators? Are their facilities equipped with security systems, fire suppression and effective HVAC?

- **Support.** Do they keep a 24/7/365 NOC staff at all datacenters, with 24/7/365 technical support? Do all support representatives speak your native language fluently? Where is the call center located? Support staff should be located in a time zone close to yours to facilitate easy communication during normal business hours.

- ** Trouble ticket resolution.** What are their average response times for support tickets? Average ticket response time should never exceed 24 hours, and would ideally be real-time.

- **Server and network maintenance.** What are the average resolution times for server, network, power, or other failures? Reputable managed datacenter providers offer a four-hour server repair or replacement policy in their Service Level Agreements. All facilities should have redundant network and power systems, but catastrophic failures should be rare and last no more than two to four hours.

**Operating System**

Flash Media Server can be installed on Windows or Linux operating systems.

Microsoft® Windows Server 2008 or Windows Server® 2003 (with Service Pack 2) are supported. Windows deployments can be easier to set up and manage than Linux, and experienced administrators can be more plentiful.

Red Hat Linux 4 or 5.2 deployments tend to provide more efficiency, offering 10-15% better CPU performance for live streams, and 25-35% better CPU performance for pre-recorded streams than Windows installations.

For the most up-to-date system and operating system requirements, refer to [http://www.adobe.com/go/fms](http://www.adobe.com/go/fms).

**Capacity Planning and Options**

Effectively planning a successful large-scale FMS deployment requires optimization of your network, server hardware and FMS software installations.

**Planning for Optimal Performance**

Single server deployments quickly become inadequate if you encounter significant growth or drastic spikes in your content volume or server traffic. Peak throughput for a single server is limited to about 700 Mbps, leaving a cushion for normal operating overhead. If you are consistently reaching your server’s limits, the FMS logs will show frequent dropped or rejected connections.

**Geographical Distribution**

Before deployment, consider your potential geographic distribution. If your user base is widely disbursed, multiple datacenter deployments may be required to maintain a high quality of service. If you already have a live web server, analyze your user traffic statistics. If you have done any studies or market research, incorporate those into traffic projections.

Geographical location can affect latency, which results in a lower quality of service. The higher the latency between server and client, the more limitations in throughput, which may result in a higher instance of rebuffering. Also, different countries have different limits on the public Internet, such as firewalls, bandwidth limitations, and blocked ports.

Typical deployments can consist of anywhere from 2 to 12 regions. A region may be defined as an area as large as the east or west coast of the United States. Some content delivery networks (CDNs) have global coverage, but most will tend to specialize in one area, such as North America, Europe, or Asia-Pacific regions.
Finding an ideal international dedicated server or colocation provider can be a challenge. Make sure the support representatives speak a language familiar to you, and that they are available during your business hours.

**Understanding Core Processes**

FMS 3.5 is a 32-bit process that runs on 64-bit operating systems in 32-bit mode. It does support file lengths longer than 2GB, exceeding the typical file addressing limitation of 32-bit processes.

Flash Media Server operates with multiple processes. With the default installation, there will be four processes running: master, edge, core, and admin. The master process is a monitor that starts core processes when necessary. There can only be one master process running at a time, but there can be many core processes running at a time.

Flash Media Interactive Server 3.5 offers flexibility in configuring multiple server processes. "Process scopes" help distribute Flash Media Server operations, optimizing server performance and improving quality of service. Proper configuration of process scopes can provide more efficient connection handling and memory usage, as well as isolation of malformed scripts and malicious attacks. This means that even if one process crashes, the others are not affected in any way. Splitting your processes:

- Allows Flash Media Server to accept connections faster
- Allows Flash Media Server to store more video/audio data in RAM
- Expands the 2GB memory limit
- Isolates the process in the event of malformed scripts or DoS attacks.

Scope preferences are set in the Application.xml file. You can specify the number of processes, the process scope, how long a process runs, and the number of process failures allowed before a core process is disabled. There are five different process scope options: Adaptor, VHost, Application, Instance, and None. Core processes and process scopes are covered more thoroughly in the Adobe Flash Media Server 3.5 technical white paper.

- **nc.connect**("rtmp://streaming.adobe.com/application/instance/")
- **streaming.adobe.com**
- **streaming.adobe.com**

Configuring VHosts is straightforward. Each VHost has its own directory inside the adaptor directory on the FMS machine. Each VHost must be mapped to a Domain Name Server (DNS) entry or another name resolution such as a Windows Internet Name Service (WINS) address, or a hosts file that specifies an IP address on the server. Each adaptor must also contain a _defaultVHost_directory in addition to the custom VHost directories. If a client tries to connect to a VHost that does not exist, the server will attempt to connect it to _defaultVHost_.

**Deployment Architecture**

Flash Media Server provides a range of large-scale deployment options, from virtually plug-and-play deployment with Edge and Origin, to robust, highly-configurable solutions using Origin-Only Cluster deployments.
Understanding Origin and Edge

To choose the best deployment architecture for your network, it is important to first understand the basic concepts behind Edge and Origin load balancing.

**Edge and Origin configuration** is a great solution for large internal deployments, such as enterprises or small CDNs. This configuration is easy to set up, as it takes advantage of built-in cache management. A simple configuration change is all that is required to configure Flash Media Interactive Server (FMIS) as an Edge server (also called a proxy). Edge (proxy) configurations will automatically manage local disk and memory cache, so you don’t have to. Caches are managed on demand, so the most popular media segments are available quickly when requested. Caching ultimately brings the content closer to the viewer, increasing response time, decreasing latency and improving the overall quality of service.

Edge servers build their caches from an FMIS identified as an Origin. The Origin configuration (the default) is the central execution point and handles file storage. All logical processing and data storage occurs at the Origin. Edge servers connect to the Origin and manage connections, CPU and bandwidth. For example, the Edge can be used to redistribute a stream to thousands of users, while only requiring a single connection to the Origin. This protects the Origin, and saves bandwidth, CPU load, and memory. Edge servers can also manage authentication using custom C++ plug-ins.

Edge and Origin configurations support multiple datacenter locations, which add stability and redundancy to the distribution network. This helps to ensure increased uptime, reliability and overall quality of service. The distributed nature of Edge and Origin means failover servers can be located apart from each other, ensuring that a catastrophic datacenter failure (power outage, natural disaster, fire, etc.) will not cause a complete service outage.

Live streaming using an Edge and Origin configuration increases overall network capacity, as the Edge servers require only one connection to the Origin, from which the live content is rebroadcast. For a more detailed discussion, see the Adobe Flash Media Server 3.5 Documentation at [http://www.adobe.com/support/flashmediaserver](http://www.adobe.com/support/flashmediaserver) for more information and tutorials on Edge and Origin connectivity.

**Origin-Only Cluster** deployments are appropriate for one- or two-server deployments or for massive-scale CDN deployment, with the support of a specialized technical staff. Origin servers can be used to perform Edge roles that are custom to the delivery network. This configuration requires more network expertise, including load balancing and cache management. Good working knowledge of Flash Media Interactive Server operations, including knowledge of C++ plug-ins and custom cache management, will help scale FMIS efficiently.

Like Edge and Origin configurations, Origin-Only Cluster deployments can be established globally or from a single datacenter. Cluster deployments give you the ability to manage increased capacity in selected locations through smart caching and localized distribution points (Points of Presence or POPs). FMIS can operate on a variety of server hardware configurations and platforms to help manage costs in low-demand regions.

Origin-Only Cluster deployments can be used for very low-volume deployments, but are also recommended for massive scale because of the granular control over the cache they provide.
### DEPLOYMENT TYPE PROS CONS

| Edge and Origin | • Servers can easily be deployed in multiple locations and connected together.  
• Single points of failure can be minimized using failover features.  
• Utilization of FMS internal caching | • Manual Cache loading not supported  
• Not optimal for massive scale |
| Origin-Only Cluster | • Massive scale capacity for CDN deployment.  
• Custom load balancing solutions can be created  
• Customized server management and monitoring can be implemented  
• Custom cache control | • In-depth knowledge of cache management is required  
• C++ and/or Server Side ActionScript must be used to manage caching  
• Advanced knowledge of authentication is required |

A principal advantage of Edge and Origin configuration is Adobe’s highly efficient cache management and garbage collection solution that helps customers get popular content closer to the user. This, in turn, reduces latency, reduces the reliance on disk access, and uses the network more efficiently.

Edge and Origin server configurations improve performance by distributing the server load among many computers on a network. With an Edge and Origin deployment strategy, all connection requests from clients are redirected to an Edge server. This configuration also lets you maximize your resources if you are supporting a large private network. With Edge servers placed in remote office locations, media files will be cached locally on the Edges so each stream does not need to access the Origin (host) server for each request.

Typically Edge and Origin deployments are best used with one-way streaming services. When using custom server-side applications to enable real-time communication, the Edge server strictly handles the connection requests on behalf of the Origin server. Client connections then make roundtrips to the Origin server to run the application.

When a client request is received, the Edge server will handle the tasks it can, then will make a connection to the Origin server for any additional data required. When the Origin server fulfills the request, the data is sent back to the Edge server, then on to the client. To the client, it appears that the connection is made directly to the application running on the Origin server.

The Edge server serves as a “traffic cop,” handling connection overhead, authentication, and other administrative duties to free up valuable system and network resources for the Origin server. Every connection and connection attempt consumes resources over and above the actual stream data flowing through the connection. As the number and frequency of connections increase, the load can be excessive, adversely affecting server performance. The Edge server greatly reduces this load by aggregating connections from a large number of clients into one connection to the Origin server. All communications between Edge and Origin servers are transparent to clients.

The Edge server also stores the prerecorded media content received from the Origin server in a cache, which is then made available to other clients that connect to the Edge server, further reducing the load on the Origin server.

Configuration decisions should be based on your knowledge and skills for content caching and ability to use C++. Edge server technology is available if you do not need custom control over your caching.
How Caching Works with Edge and Origin

In Edge and Origin configurations, the Edge server maintains a local disk cache, populated with only the parts (segments) of the video that have been watched. By caching only watched segments and not the entire files (as was the case in previous versions of FMS), the disk cache is managed more efficiently. These cached segments are requested from the Origin over a private RTMP channel, in a method similar to byte-range requests found in HTTP 1.1. This form of cache management is important, especially with high-quality, multi-bitrate video that could easily be 1GB in total file size. With files of this size, the disk cache would quickly be filled.

For example, as a stream is being played, parts of the stream (segments) are requested by the client. The FMS Edge server will look for the segment in its in-memory cache. If the segment is not found in the cache, it will try to load it from the source file. If the local file is not on that Edge server, it will request it from the Origin. Then, the Edge will try to insert the segment into its segment cache. If the segment cache is full, then it will try to remove something to make room for the new segment. Segments are removed on a Least Recently Used (LRU) basis until enough space is cleared for the new segment. If a segment is currently in use, it cannot be removed. If all segments are in use, and not enough space can be cleared to add the segment, the segment fails to load, and the stream play that requested that segment will prematurely stop.

FMS relies on garbage collection to manage segment removal. To manage disk cache garbage collection, the FMS administrator can specify a maximum cache size on each Edge server. Once that size is reached, FMS begins garbage collecting files based on approximate Least Recently Used (LRU) ordering. The file segments are grouped into subdirectories, or “buckets,” which are then sorted by LRU. When garbage collection is initiated, the oldest buckets are deleted first. The number of buckets is configurable, with the default being eight. More buckets will yield more granular deletions, but also will result in more shuffling of files between buckets, resulting in higher disk I/O.

For more information about caching, refer to the “Edge Cache” section later in this document.

Configuration Options

There are two configuration options for large-scale Flash Media Server deployments: Edge and Origin and Origin-Only Cluster.

Edge and Origin

Flash Media Interactive Server features built-in Edge and Origin functionality.

Edge and Origin is best suited for large deployments in organizations such as enterprises or small CDNs. It is often used in conjunction with less mature back-end storage systems, using built-in file access functionality. The Edge and Origin solution is a great turn-key solution for most customers. However, for customers who have advanced knowledge of cache management and would like granular control over managing the load on servers, it’s recommended that an Origin-only Cluster solution be used in combination with custom plug-ins that perform caching and access control tasks. (Adobe recommends that only experienced stream managers or massive-scale CDNs use Origin-Only Cluster approach for large-scale deployments.)

In Edge and Origin configurations, the operating system and disk format should be identical for all Origin and Edge servers to avoid file naming conflicts that can break your deployment. The recommended ratio of Edge to Origin servers is generally 5 to 1. It is recommended that failover Origin servers be implemented as well.

One significant benefit to the Edge and Origin configuration is the ease of setup. To configure an Edge server, you simply edit your vhost.xml configuration file in a text editor, making the following changes:
1. Find the Mode setting, and change the value to **remote**.

2. Find the Anonymous setting, and change the value to **true**. This configures the FMIS server in an implicit installation. For explicit installations, set this to **false**. (Implicit installations are usually recommended, since they require no changes to the URL presented to the end user.)

3. Find the RouteEntry setting, and change the values. The first value is your Edge server virtual server IP address and port, and the second is the Origin server virtual server IP address and port. For example, if you are using the RTMPS protocol, your Edge virtual IP is 10.20.0.0, and your Origin Virtual IP Address is 10.234.56.78, the RouteEntry would be:

   ```xml
   <RouteEntry>10.20.0.0:443;10.234.56.78:443</RouteEntry>
   ```

4. If you wish to turn on the local Edge cache, find the CacheDir setting and change the value to **true**. This setting is optional.

For more in-depth information about configuring Edge and Origin deployments, refer to the Flash Media Server 3.5 documentation.

**Origin-only Cluster**

Origin-Only Cluster configuration allows managed scalability in highly customized deployments. With this option, multiple Origin servers are deployed behind a load balancer to distribute client traffic. Deploying multiple Origin servers enables reliable application scaling and creates redundancy, which eliminates single points of failure.

The Origin-Only Cluster approach is generally best for live or VOD streaming, where clients do not need to communicate with each other from within specific application instances. Clustering can be achieved using either Flash Media Streaming Server or Flash Media Interactive Server.

For these types of configurations, you are required to implement and develop your own cache management solution on Flash Media Server using the C++ plug-ins. All content may be stored in either an internal or remote storage location. A custom File plug-in can be developed to fetch the files (or file segments) and cache to provide faster access to the content. This solution would be suitable for networks with robust back-end storage systems, and an IT staff with caching and file storage expertise.

Although FMS configuration adjustments with your specific load balancer will vary, you will need to adjust some basic settings: the amount of time idle connections are kept open (keep-alive time), cookie support, and the route entry.

By default, FMS will keep idle connections alive indefinitely. While this persistence is generally beneficial for the functionality of connection-based applications, it is not ideal for large-scale deployments. Each unused open connection is a drain on server resources, so it is recommended to close idle connections in a reasonable amount of time. To do so, edit the server.xml file. Change AutoCloseIdleClients to **true**, and adjust the CheckInterval and MaxIdleTime settings to the optimal times for your applications. (An interval of 20 is good to start with; adjust as necessary for best performance.)

In Origin-Only Cluster deployments, individual users need to be mapped back to the correct host using cookies. In order to achieve this session persistence, FMIS must have cookie support turned on. By default, FMS has cookie support turned off. To turn cookie support on, open the adaptor.xml file and change the SetCookie setting to **true**.

Finally, you’ll need to configure the HOST header that the FMS server sends in response to a client request. In Origin-Only Cluster deployments, this header must be set to the IP address of the load balancer’s virtual server. To make this change, open the vhost.xml file and change the RouteEntry setting to match the IP address and port number of the load balancer’s virtual server.

**TYPICAL LOAD BALANCING APPROACHES**

**DNS Load balancing** allows you to direct requests to a server closest to the consumer’s DNS server.

**Geo Load balancing** allows you to direct requests to a server relative to IP address — this may be more effective in today’s Internet because office workers typically use a common DNS server.

**Content aware load balancing** allows you to direct the request directly to the server storing the content.
Deployment Type Comparison

**Edge-Origin Configuration**
Connections are received and accepted by Edge servers. The Origin server(s) host applications and streams, with these Edge servers acting as proxies.

**Origin-only Cluster Configuration**
A load balancer such as Windows Network Load Balancing, a Linux-based load balancer, or a load-balancing appliance distributes incoming connections evenly to all servers in the cluster.

**Multiple Tenant Provisioning**
Flash Media Server includes powerful virtualization features, which makes provisioning multiple, separate user accounts simple. Adobe recommends provisioning at the application level for multiple tenants. Each application directory on the server contains its own configuration file, allowing you to define customized settings for each application. Assigning multiple client accounts to their own applications is also recommended if they have server access privileges. It keeps each client’s content separate and secure, and custom server-side code “sandboxed.”

Limiting each core process to a specific number of applications can limit the number of applications that can be taken down by a single bad one. Configuring multiple tenants at the application layer allows per-user control of:

- Client bandwidth throttling
- Process scopes
- Available protocols

Adobe recommends provisioning at the application level for multiple tenants.
• Content protection (ACL/SWF Verification/RTMPE)
• Automatic script loading
• Memory allocation
• Virtual directory locations
• Cache
• Performance tuning
• DVR controls
• Shared object controls

You can also allocate server resources across multiple tenants based on process scope settings. Selection of a scope depends upon how you provision your customer accounts. Flash Media Server can be configured to spawn core processes in the following scopes:

• **Adaptor:** Best for stateful applications where clients need to communicate with each other (chat, live video, gaming, or data-sharing solutions).

• **VHost:** Useful for applying unique settings for users in different subdomains.

• **Application:** The recommended setting, this runs each application within its own process. Useful if you have lots of memory and heavy connection requests from different applications.

• **Instance:** Provides the most granular scope distribution.

Each process can use as much as 4GB of virtual memory, so spawning too many processes may not be the best choice. Refer to the Adobe Flash Media Server 3.5 technical white paper for more information on process scopes.

**Access Control and Authentication**

In addition to the inherent security of streaming, Flash Media Server offers many features that allow granular access control. The following chart and diagram illustrate the various approaches available.

**Flash Media Server Security**

![Flash Media Server Security Diagram]

FOR MORE INFORMATION

Refer to the official Adobe Flash Media Server 3.5 white paper at [http://www.adobe.com/go/fms_whitepaper](http://www.adobe.com/go/fms_whitepaper) for more information on process scopes.
Firewalls and Port Filtering

Firewalls are valuable additions to any server configuration. They can help stop hackers dead in their tracks, and keep them from accessing your valuable data or attacking your servers.

It is recommended that all unused TCP ports are blocked. If you don’t have access to a firewall or don’t wish to use one, Windows and Linux both offer the ability to filter ports, which closes off connections and deters port scanners.

The default ports that should be opened for Flash Media Server are 80, 443, and 1935. Port 1111 needs to be open for remote admin server access. It is very important that port 3389 is kept open on Windows servers if you are using Terminal Services. Linux servers should have port 22 open for SSH terminal access. Other common ports include 873 for rsync, 20 and 21 for FTP, and 22 for SFTP.

Common attacks include port scanning, Denial of Service (DoS) attacks, Distributed Denial of Service (DDoS) attacks, brute force login scripts, and viruses. These attacks can all be negated with basic security planning.

User Authentication via SSAS, Access Plug-in or Authorization Plug-in

Several methods of user authentication are available with Flash Media Server 3.5: custom Server-Side ActionScript, writing an Access plug-in, or using Authorization plug-ins.

Server-Side ActionScript allows you to implement a simple username/password, an encrypted token (MD5 Hash), or a unique key. Then, on the server side, Flash Media Server would be able to integrate with web services (SOAP), Flash Remoting, XML, HTTP Post or simple file access to validate the client based on the data sent. This authentication scheme could be as simple as checking login information against a database, or as complex as creating an SSL-based token system using ColdFusion.

Using an Access plug-in allows you to incept connections and authenticate users before they reach the server’s script layer, implementing custom logic to perform tasks such as database queries and updates, accept only a specified number of connections, or allow access to only certain files or directories. There can be only one Access plug-in per Flash Media Interactive Server installation.

Authorization plug-ins authorize client access to server events, and can be “stacked” to perform sequential actions on incoming connection requests. Once the connection has been established, but before it is accepted, Authorization plug-ins can allow you to implement logic as basic as

<table>
<thead>
<tr>
<th><strong>ACCESS CONTROL METHOD</strong></th>
<th><strong>BENEFITS</strong></th>
<th><strong>IMPLEMENTATION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewalls and port filtering</td>
<td>Prevents port scanning, Denial of Service (DoS) attacks, brute force login scripts, and viruses.</td>
<td>Use a firewall or port filtering in the server operating system to block all unused TCP ports on servers running FMS.</td>
</tr>
<tr>
<td>Username/password authentication</td>
<td>Allows you to set specific user access levels and record user access data.</td>
<td>Validate against user database using ActionScript and backend scripts such as ColdFusion or PHP.</td>
</tr>
<tr>
<td>Token-based authentication</td>
<td>Provides secure authentication based on token handshake Can be a simple access control solution.</td>
<td>Integrate with web services (SOAP), Flash Remoting, XML, HTTP POST or simple file access to validate the client</td>
</tr>
<tr>
<td>Access plug-in</td>
<td>Intercepts requests before they are connected, potentially conserving server resources and providing extra security. Allows you to accept, reject, or redirect connections based on high-level server criteria such as how many users are currently connected, or can even authenticate against a database.</td>
<td>Develop an Access plug-in.</td>
</tr>
<tr>
<td>Authorization plug-in(s)</td>
<td>Intercepts requests after they are connected, but before they are accepted and code is executed. Gives you the opportunity to accept, reject, or redirect the connection, and call custom server-side ActionScript functions.</td>
<td>Develop one or many Authorization plug-ins.</td>
</tr>
<tr>
<td>Domain blocking</td>
<td>Strict control over who accesses Flash Media Server streams and resources.</td>
<td>Create a whitelist (or blacklist) of domains in the Adaptor or VHost configuration files.</td>
</tr>
<tr>
<td>SWF verification</td>
<td>Restricts FMS access to authorized SWFs, preventing unauthorized attempts to access content and server resources.</td>
<td>Place copies of authorized SWF files in the Flash Media Server SWF directory for your application, and turn on the feature in Application.xml..</td>
</tr>
</tbody>
</table>
Limit Access by Domain or IP

Preventing unauthorized domains and networks from utilizing your resources is one of the first steps in locking down your server. By default, a client can connect to Flash Media Server from any domain or IP address, which can be a security risk. You can create a whitelist of allowed host names, domain names and full or partial IP addresses in either the Adaptor or VHost configuration files.

SWF Verification

SWF Verification is a security feature of Flash Media Server 3.5 that gives you control over which SWF files can connect to your server. Without using this feature, unauthorized clients such as Replay Media Catcher software, an open source RTMP client or proxy, or any SWF file (with the proper connection string and application name) could freely connect to your server — accessing your streams and consuming your server resources. SWF Verification allows you to designate specific SWF files with permission to connect to your server. To enable this feature, simply store a copy of the approved SWF file in your FMS application directory, and then turn on the feature in the Application.xml file. Then, when a SWF file attempts a connection, FMS will verify that the file exactly matches one of the SWF files in your application directory before accepting the connection. You can configure the server to check the length of time the verification data is held in cache, how often the server should check for updated approved SWF files, and if there should be any exceptions (such as Flash Media Live Encoder).

Server Health and Monitoring

Flash Media Server provides a number of different ways to monitor its health and performance. The simplest checks will verify that:

• The server operating system is up and running.
• FMS is running.
• Connections are being accepted.

Deeper health checks would provide information about:

• Memory usage
• CPU utilization
• Disk access
• Network performance and utilization
• Current number of connections
• Cache utilization
• Number of streams currently playing

These server details can be revealed using plug-ins, the Administration Console, the Administration API, and the FMSCheck Utility. You can also use third-party software to monitor FMS. Nagios, Zenoss, Groundwork Open Source, and Hyperic HQ are open source and free. You can purchase “Enterprise” versions that include support, additional features, training, limited consulting, and more. These applications offer various methods such as SNMP queries, agent applications, WMI queries, and pings to track server, application, and service statuses, and can automatically notify server administrators upon detecting any server issues.

Several standard tools are also recommended to efficiently manage a large-scale FMS deployment. Active Directory can help you pull together all of your servers under a single administrative policy, enabling you to enact network-wide policy changes quickly and efficiently. Management interfaces, KVM over IP, terminal services, VNC and the SSH command line are valuable tools for day-to-day management of your servers.

MONITORING SYSTEMS

Nagios
http://www.nagios.org

Zenoss
http://www.zenoss.com

Groundwork Open Source
http://www.groundworkopensource.com

Hyperic HQ
http://www.hyperic.com
Plug-ins

Flash Media Server 3.5 features a plug-in architecture that can be used by administrators to make maintenance and administration more efficient. C++ plug-ins can be developed to limit or modify access to applications, create an authentication layer for streams and services, and implement a file I/O system. For more information, tutorials, and sample plug-ins, see the Flash Media Server 3.5 documentation, or the Understanding Plug-ins section later in this document.

Administration Console

The Flash Media Server Administration Console is a SWF file that is useful for viewing the current statistics of a server or virtual host. It provides a live log of trace events, useful for debugging applications, as well as lists of active Clients, Shared Objects, and Streams. The performance section allows an administrator to view real-time statistics on active connections, bandwidth usage, and CPU/memory statistics, which can be used to determine server efficiency and utilization. The Administration Console is included with the installation of Flash Media Server and can be run locally on any machine or hosted on a web server. It utilizes the Administration API and connects to the FMS Administration Server on port 1111 by default.

Administration API

The FMS Administration API consists of 50 commands that enable you to create your own custom administration tools for monitoring, managing, and controlling FMS. Custom applications can allow local or remote access to perform many administrative tasks. These tools can run in an Adobe Flash Player or Adobe AIR client over RTMP or RTMPE (via default port 1111) or from a web client over HTTP. With the Administration API, you can execute server functions such as managing administrator and virtual host accounts, creating and deleting applications, starting and stopping the server, loading and unloading applications and instances, garbage collection, and more. To use the Administration API, both Flash Media Server and Flash Media Administration Server must be running. Refer to the Flash Media Server Administration API Reference for more information.

FMSCheck Utility

The FMSCheck utility is a free scriptable command-line tool available for Windows and Linux that can be used to determine server status and diagnose problems. Because it is a command-line tool, it can be integrated into any automated system using scripting languages such as C script, perl, bash, csh, VBScript, or Python. Use this utility to perform checks such as:

- Publish or play a stream
- Play a server-side stream
- Determine whether the server is running or not
• Measure current server response time
• Determine if any fmscore processes are not responding
• Connect to an instance of a single application, or to all active application instances.

To access the FMSCheck utility, navigate to the tools subfolder of the Flash Media Server 3.5 installation from a command prompt, and invoke FMScheck.exe (or ./fmscheck on Linux) with the proper parameters; for example:

```
fmscheck --host localhost --port 1935 --app myApp --logfile "C:\Program Files\Adobe Flash Media Server 3.5\fmsCheck01.log" --play "C:\Program Files\Flash Media Server 3.5\Test.flv" 0 20
```

Results can be seen in the FMSCheck log file or in the FMS Administration Console. For more information about the FMSCheck utility and required parameters, see the Flash Media Server 3.5 Configuration and Administration Guide.

Logging
With Flash Media Server 3.5, you have powerful logging capabilities that can be highly customized for your specific application. Typical uses include billing, quality-of-service monitoring, and troubleshooting. All log files use the W3C extended log file format and can be parsed with standard tools. You can watch them in real time, parse them with a log analysis tool, or import the files into a spreadsheet program.

There are four different types of log files:

• **Access logs.** Track user requests for access to server resources. By default, there is one access log per server, but they can be configured to be specific to each virtual host.

• **Application logs.** These are used primarily for debugging applications. A separate log file is created for each application instance.

• **Diagnostic logs.** These logs track server operations and processes.

• **Plug-in logs.** Logs messages and events.

You can also configure custom logging using the C++ plug-ins. Custom logs could report on critical issues with the server, or advanced per-client log information, or even reference a publishing system ID.

Located at the root level of the configuration directory, the Logger.xml file controls settings for Flash Media Server log files. You can edit this file to specify the data that is logged, how the log files are named, where they are saved, and how often they are rotated. The default location for the log files is in the logs directory in your server installation directory (RootInstall/logs).

The Logging section in Server.xml enables or disables the log files; Logger.xml contains the actual log file settings. In typical deployments, FMS can produce several gigabytes of logs every second, so a robust log management system should be part of your capacity planning. Log management systems are available from third-party providers, or can be custom built for your needs.

Understanding Plug-ins and Applications
Unlike many other web video delivery technologies, which just present pre-branded players to your viewers, the Flash platform allows you to create completely customized interfaces with rich interactive features. Real-time data sharing, server-side plug-ins, logging, and monitoring application programming interfaces (APIs) provide developers and IT teams with the tools they need to develop and administer large-scale rich media applications.

An Flash Media Server “application” consists of a client SWF file and an associated service or custom application on the server. The client SWF file can run in Flash Player in a browser, AIR on the desktop, or in Flash Lite on devices. FMS ships with two pre-built applications, or services: VOD for video on demand, and Live for live broadcasts. These services enable you to begin streaming via FMS right out of the box, with no server-side programming required. With Flash Media Interactive Server, you can also write your own Server-Side ActionScript code to create custom applications.

In addition to custom applications, Flash Media Interactive Server allows plug-ins written in
C++ that enable you to extend the functionality of the server itself. Plug-ins can perform access security checks, allow geographical targeting of content, track statistical data about clients, and execute network-based file operations. Flash Media Server 3.5 ships with new sample plug-ins that make it even easier to get started.

Some typical use-cases include:

<table>
<thead>
<tr>
<th>USES FOR C++ PLUG-INS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve a remote file</td>
<td>Access files not local on the network. Can be implemented by developing solutions using HTTP byte-range requests (for segmented loading) or FTP.</td>
</tr>
<tr>
<td>Cache management</td>
<td>Can be used to manage garbage collection and monitoring cache-hits. Can also be used to create your own custom Edge server by managing memory and disk cache for Origin servers on the edge of the network.</td>
</tr>
<tr>
<td>Access controls (ACL)</td>
<td>Leverage LDAP, Active Directory or custom provisioning solutions including timed tokens to restrict access to streams</td>
</tr>
<tr>
<td>Custom logging</td>
<td>Extend the base logging functions of FMS and allow tracking of additional critical issues, warnings or alarms</td>
</tr>
<tr>
<td>Server monitoring</td>
<td>Monitor server health and create a management and notification solution to handle errors and failures (e.g. create alarms, establish memory management routines, or inform routers).</td>
</tr>
<tr>
<td>Deep client/connection monitoring</td>
<td>Monitor and manage quality of service at the connection level. Could be used to allow/disallow certain actions such as seeking, access to a particular stream, Dynamic Streaming, or DVR max duration.</td>
</tr>
</tbody>
</table>

There are three plug-in classifications: File, Authorization, and Access:

<table>
<thead>
<tr>
<th>PLUG-IN CLASSIFICATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Provides an asynchronous interface between FMS and the operating system's file I/O mechanism, giving you complete control over where and how the server reads content from the file system.</td>
</tr>
<tr>
<td>Authorization</td>
<td>Used for a variety of security and user administration functions, as well as Quality of Service (QoS) monitoring. Multiple Authorization plug-ins may be used, and are executed in sequence. Some uses include authorizing playing, publishing or seeking in a stream; connecting/disconnecting clients; calling methods in Server-Side ActionScript; delivering streams based on geographic location or subscription level; limiting time and duration of user’s access to specific streams; writing QoS data to external logs; and more. Authorization plug-ins can be used with both Edge and Origin deployments, but the functionality is different in each case. Refer to the Flash Media Server documentation for more details.</td>
</tr>
<tr>
<td>Access</td>
<td>Adds another layer of security to the server. Unlike Authorization plug-ins which run in the Core process (fmscore), the Access plug-in runs in the Edge process (fmsedge), allowing it to intercept connections before they hit the server's script layer. You can code the plug-in to accept, reject, or redirect requests based on criteria like how many users are connected to the server, the amount of bandwidth being consumed or if the user is in your database. Other uses include setting read and write access for files and folders on the server, allowing access to video bitmap data, and inspecting client properties. There can only be one Access plug-in per server.</td>
</tr>
</tbody>
</table>

Using an Access plug-in on an Edge server enables you to intercept connections before they reach the Origin server’s script layer. An Access plug-in is a server plug-in written in C++ that determines if the intercepted connections should be accepted, rejected, or redirected. You can implement custom logic in the Access plug-in to handle client connection requests. For example, you can query your account database upon client login, and then update the database record after the client connection is accepted, or accept a request only if there are fewer than a certain number of clients currently connected. You can also set read and write access for files and folders on the server, set permissions to access audio and video bitmap data, and inspect client properties through the Access plug-in. The Access plug-in is available with Flash Media Interactive Server, and requires Flash Player 6 or later on the client side. There can only be one Access plug-in per Flash Media Interactive Server installation.

Authorization plug-ins are also server plug-ins written in C++ that authorize client access to server events. These plug-ins come into play once the connection has been established, but before it is accepted. Authorization plug-ins can:

- Authorize connections to the server
- Disconnect clients from the server
- Authorize playing, publishing, or seeking in a stream
- Call custom methods in server-side ActionScript
- Deliver content to clients according to their geographic location, subscription level, or stream origin
• Limit time and duration of a user’s access to specific streams

• Map a logical stream path to a physical stream path. For example, a client requests the stream “high.flv,” but since he is not a premium member of the service, he should only receive the low-quality version of that content, so he is actually served “low.flv.”

Unlike the Access plug-in, you can use multiple Authorization plug-ins to sequentially perform actions on the incoming event. For example, the first plug-in could authorize the client connection; the next plug-in could then authorize playback of a high-definition version of a stream rather than a low-quality version.

Storage and Cache
Proper planning of file storage systems and content caching is essential in large-scale deployments. There are many ways to store your content files, and many factors to consider when building file delivery architecture for large scale deployments.

Storage Options
There are two main approaches to content storage for FMS deployments: local storage or remote storage.

Local storage either uses the default FMS storage location or uses configuration files to map to a central storage directory on the server or local network. Media files for an application can be stored in the following locations:

• The default “applicationName\streams\instanceName” folder
• An alternate location defined in the <storageDir> tag in Application.xml
• Any additional locations defined in the <VirtualDirectory> tag in Vhost.xml or Application.xml

This configuration is the simplest to deploy but is often insufficient in large scale scenarios where FMS is supporting large volumes of content. Local storage can present challenges for segregating content by customer, and tends to be inflexible and difficult to scale. For these reasons, most CDNs employ a remote storage solution.

Remote storage provides a more robust and flexible approach. In this scenario, all servers request content from a central external content repository. There are two types of remote storage: “near” and “far”. Near storage can simply be mounted as a network drive, while far storage requires a File plug-in to deploy. It is very important that the File plug-in be fine-tuned for optimal performance with your network. Some important best practices include:

• Do not block threads.
• Request and return file data to FMS as quickly as possible.
• Use HTTP range requests.
• Perform large read-ahead and cache as much data as possible locally so it is ready on demand.

Refer to the Flash Media Server 3.5 documentation for further details about configuring content storage and programming the File plug-in.

Edge Cache
Content caching is a crucial element of a large-scale deployment, as it can have a profound impact on the final playback experience. The ideal caching solution would be to cache 100% of your content in memory on every server in your entire network. Of course, this is unrealistic; so the goal is to store the most frequently requested content as close to users as possible, thus limiting the number requests for content from the file system.

Content can be cached at multiple points in an FMS deployment. The built-in FLV cache in FMS is the most basic cache point, storing content that has already been served in server memory. This decreases the frequency of disk access, but is bound by total amount of server RAM and the 4GB virtual memory space limitation of 32-bit processes in Windows. Configuring how applications are assigned to core processes can help optimize FLV caching performance.
The next level of cache available with FMS is the Edge cache. Edge servers can be employed to cache stream segments closer to end users, which can decrease latency and improve QoS. Stream segments are stored in server memory and can also be configured to save to disk. Edge cache deployments require you to create policies for purging stale content, however, and cache smearing can occur.

Origin-Only cache handling requires nothing but the FLV cache to configure, along with a strong back-end storage system and a well-tuned File plug-in on each Origin server.

**Memory Management**
Flash Media Server 3.5 features enhanced cache behavior, improving memory management and overall server performance. You can set a ceiling on the amount of RAM utilized by the cache by changing the SERVER.FLVCACHE_MAXSIZE parameter in the server's fms.ini file. The default value is 500MB.

In addition, the cache is based on segments, or chunks, rather than entire files, resulting in much more efficient caching behavior. When the cache is full, the server removes unused chunks starting with the least recently used. The size of these chunks can be set in the APP.DEFAULT_CHUNKSIZE parameter of the fms.ini file. Larger values will reduce CPU usage, but can also slow performance for clients on lower bandwidth connections and can affect server performance. For example, if the cache size is greater than the available memory, or the server process exceeds the OS memory limit, the server process could be terminated. Alternatively, if it is set too low, all chunks could theoretically be in use and unable to be exchanged for new stream chunks. In this case, the stream requesting the new chunk will stop playing.

For more information about fine-tuning memory management, refer to the Flash Media Server documentation.

**Live Streaming**
Live streaming is a powerful feature of Flash Media Server, but can pose some interesting challenges in large-scale deployments. Adobe offers some solutions, including multipoint publishing and the LiveStreamCast application.

**Ingest Points**
Flash Media Server provides a live service that acts as an ingest point, allowing you to broadcast your live streams right out of the box, with no custom scripting or server configuration. This live service is located in the FMS applications folder, and would be accessed using a connection URI in ActionScript such as:

```
rtmp://myFMSHost.com/live
```

Live streams can be broadcast via Flash Media Server 3.5 from either a webcam source in Flash Player, from Flash Media Live Encoder, or from a third-party live encoder. In standard deployments, the stream is sent from the source directly to a server running FMS. Clients then connect to that same server to view the stream. This limits the scale of the broadcast, and is not feasible for large scale deployments. To achieve greater flexibility and scalability for live broadcasts, you can implement the multipoint publishing feature available in Flash Media Interactive Server 3 and later. With multipoint publishing, a Flash Media Server or Flash Media Live Encoder can be used to control the feed to the Origin server, which then handles the large scale broadcast. This allows the broadcaster to inject data messages into the stream, and allows implementation of server-side code without deploying custom applications to CDN servers.

**Using LiveStreamCast**
LiveStreamCast is a prebuilt application structure that helps create out-of-the-box scalability for live Flash Media Interactive Server applications. It consists of an Origin Server Node, an Intermediate Server Node, and an Edge Server Node. Each node in this three-layer structure is an FMS application, written in Server-Side ActionScript (.asc), and runs on its own FMS server, configured in Origin-Only mode.

LiveStreamCast is a free download from http://www.adobe.com/go/fms_tools
LiveStreamCast with multiple Origin groups and multiple encoders. In this scenario, a secondary Origin is available from Encoders 1 and 2, but not from Encoder 3.

LiveStreamCast with optional intermediate node. In this scenario, an intermediate server is installed with the Edge Server node application. Note that the Edge Node references the primary and secondary server as the same server.

Because of the added complexity involved in scaling live applications, subscribing directly through the use of `NetStream.publish()` and `NetStream.play()` commands may not be reliable. The distributed node structure of LiveStreamCast addresses this by allowing servers to keep track of all live streams being published and pass the publishing information down to the subscriber. This is a much more efficient and reliable approach for large-scale broadcasts. The following chart details the functionality of each element of the LiveStreamCast application structure:
<table>
<thead>
<tr>
<th>NODE</th>
<th>DESCRIPTION</th>
<th>FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin Server Node</td>
<td>This server-side FMS application receives the video stream from the publisher and distributes the data to multiple Intermediate Server Nodes.</td>
<td>Ability to accept connections from publishers and maintain a list of all publishers Accepts connections from Intermediate nodes and notifies Intermediate nodes when a publisher is added or removed Accepts play command from the Intermediate nodes and distributes data to the Intermediate nodes</td>
</tr>
<tr>
<td>Intermediate Server Node</td>
<td>This server-side FMS application receives video streams from the Origin Server Nodes and passes the data to multiple Edge Server Nodes. It is the middle layer of the distribution network.</td>
<td>Ability to read Origin IPs from a configuration table in host.ini Connects to all Origin nodes in the configuration table and reconnects if connections drop Receives notifications from Origin nodes and maintains a list of active publishers for each Origin node Sends play command to and receives data from Origin nodes Reports connection/stream errors to Edge nodes</td>
</tr>
<tr>
<td>Edge Server Node</td>
<td>This server-side FMS application receives connection requests from clients and delivers the data from Intermediate Server Nodes to the clients.</td>
<td>Ability to read Intermediate IPs from a configuration table in host.ini Connects to all Intermediate nodes in the configuration table and reconnects if connections drop Receives notifications from Intermediate nodes and maintains a list of active publishers for each Intermediate node Sends play command to and receives data from Intermediate nodes Reports connection/stream errors to Edge nodes</td>
</tr>
</tbody>
</table>

**Content Protection Measures**

While streaming provides some inherent content protection, Flash Media Server features additional security options.

**Deploying RTMPE and Limiting RTMP**

Encrypted RTMP (RTMPE) is enabled on Flash Media Server by default, and enables you to send streams over an encrypted connection without requiring certificate management. Offering secure 128-bit encryption, RTMPE is generally the recommended form of encryption, as it is easier to deploy and is much faster than SSL. (However, some load balancers do offload SSL processes, which could make SSL a more efficient option by taking the encryption load off of the FMS machine entirely.) RTMPE is supported in Flash Player 9 or later.

To implement stream encryption for your content, simply specify the protocol when you connect to your application:

- **SSL**
  ```javascript
  NetConnection.connect("rtmps://yourFMSserver.com");
  ```

- **Tunneled SSL**
  ```javascript
  NetConnection.connect("rtmpts://yourFMSserver.com");
  ```

- **Encrypted RTMP**
  ```javascript
  NetConnection.connect("rtmpe://yourFMSserver.com");
  ```

- **Tunneled encrypted RTMP**
  ```javascript
  NetConnection.connect("rtmpte://yourFMSserver.com");
  ```

RTMPE can be enabled/disabled at the Adaptor level, but cannot be used to encrypt connections between Edge and Origin servers.

It is very important to disable standard RTMP access when using RTMPE to prohibit unencrypted connections.

**SWF Verification**

Another level of security for your FMS servers is SWF verification. This feature prevents content from being accessed by unauthorized client applications. Copies of authorized SWF files are placed on the server (or in a remote file repository); connection requests are then only accepted from SWF files that match one of the authorized files. You can configure the location of the

**CONTENT PROTECTION RESOURCES**

How to protect video content (Flash Media Server) TechNote
http://www.adobe.com/go/kb405456

Video content protection measures enabled by FMS 3 white paper
http://www.adobe.com/go/fms_protectvideo
SWFs folder, agent exception lists (e.g. Flash Media Live Encoder), scan interval for the SWFs folder, and SWF cache duration.

The default configuration for SWF verification does present some challenges for large-scale deployments. Managing the distribution of authorized SWF files for multiple applications across multiple servers can be daunting. To simplify this task, a global SWF verification folder can be configured in the server.xml file. Also, you may want to consider decreasing the default value for the cache duration to facilitate quicker changes.

One pervasive security risk, stream ripping, enables the viewer to directly access and record the data of a stream. Use RTMPE in conjunction with SWF verification to prevent stream ripping.

Troubleshooting

<table>
<thead>
<tr>
<th>ISSUE</th>
<th>DESCRIPTION</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video Lag/Synch Issues</td>
<td>The video is choppy, slow, or audio is out of sync.</td>
<td>CPU, memory or network limits may have been reached. Video and audio lag and de-synchronization can also be caused by poor encoding practices, excessively high bit rates or poorly architected applications. If a server’s CPU becomes heavily loaded, new processes will increasingly queue up, increasing wait times and latency. Multi-core, multi-thread processors perform better than single-core processors, of course, but CPU overload can still occur when all cores are consistently at high utilization. Physical memory can begin to run out when a large number of unique videos are requested or shared object usage is heavy, requiring memory to be paged, which is very slow and inefficient, and can even cause the server to crash. Networking limitations can also become an issue when streaming large amounts of video. If the server’s network card is saturated, packets may be dropped and streaming performance will suffer. Multiple network cards can help balance the network load and increase the server’s streaming capacity. Proper encoding practices are absolutely essential to ensure smooth video playback. If the bit rate is too high, the server or the client may not be able to handle the bandwidth, resulting in choppy playback. In addition, it is recommended that streaming video be encoded using constant bit rate (CBR). Using the Dynamic Streaming feature in FMS 3.5 can also improve overall QoS by detecting the viewer’s bandwidth and serving them a stream at the appropriate bit rate, dynamically adjusting to their current network conditions. Proper application architecture is critical to ensuring optimized server performance. A poorly coded client application can cause excessive CPU usage, inefficient memory allocation, and general poor performance. For example, it is important to avoid unnecessary loops, properly enable and disable event listeners, and close streams when they are no longer needed.</td>
</tr>
<tr>
<td>Video Lag/Synch Issues</td>
<td>My live stream is lagging by several seconds.</td>
<td>Live stream publishing issues are usually due to insufficient bandwidth from the publisher to the server. It is recommended that you publish over the fastest, hard-wired Internet connection available. Be sure it is a dedicated connection, as shared connections can be unreliable. Latency can also be caused by an overloaded server.</td>
</tr>
<tr>
<td>Video Lag/Synch Issues</td>
<td>My on-demand stream is stuttering, stopping, or will not play.</td>
<td>Be sure the video codec is compatible with the client’s version of Flash Player. H.264 video will only play in Flash Player 9 or later. If the failure occurs on only specific videos, be sure they are not corrupt by inspecting them with the FLVCheck utility. This utility will also tell you if the video’s codec profile is compatible with Flash Player. Also check the possible causes of poor performance previously discussed.</td>
</tr>
<tr>
<td>Video Lag/Synch Issues</td>
<td>Only some clients have trouble with video playback; others are receiving streams just fine.</td>
<td>In order to use FMS-enabled applications, a user should have at least a broadband connection. Generally, at least a 512 Kbps (download) speed is recommended to receive streaming videos smoothly and participate in video chats. Since FMS does offer bandwidth detection and dynamic streaming, lower-quality videos can be provided for users on slower connections, meaning a wide variety of audiences with varying connection speeds can be served.</td>
</tr>
<tr>
<td>ISSUE</td>
<td>DESCRIPTION</td>
<td>SOLUTION</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Server Crashing</td>
<td>The server freezes or crashes.</td>
<td>Server crashes are typically caused by memory leaks or corrupt media files. To determine the root cause of the server crash, begin by inspecting all FMS log files. Note the time when the issue occurred, and document logged events from when issues began to when the server crashed. The operating system logs can also reveal details of a crash (Event Log in Windows, or /var/log/messages in RedHat Linux distributions). Obtaining crash or hang dump data can also be helpful. In Windows, use ADPlus (<a href="http://support.microsoft.com/kb/286530">http://support.microsoft.com/kb/286530</a>); in Linux, use pstack. Collecting performance data can also reveal details of a server hang or crash. Memory utilization, CPU utilization, number of connections and swap space are all valuable to determining the root cause. Document any behaviors or events that lead up to the failure.</td>
</tr>
<tr>
<td>Server Troubleshooting</td>
<td>The application(s) have stopped responding, or a connection cannot be made.</td>
<td>Begin by logging into the FMS Administration Console; check to see if there are any active connections. If so, restart the Virtual Host that you are having problems with. If there are no active connections, try stopping and restarting the Virtual Host. If this doesn't work, check the FMS log files for anything that might reveal details about the unresponsive behavior. Next, check the Event Log in Windows, or /var/log/messages in Linux, to see if any connections are being rejected, and any reasons logged. A common reason for rejecting connections can be resource limit violations. Resource limit violation errors indicate that the server has exceeded its maximum number of connections, streams, shared objects, or instances. If you have set limits on your Virtual Host, you may need to raise them. Other errors in the application logs usually indicate an issue with the application itself, or with some user input into the application. Utilize trace functions and/or external logging in your applications, and check the FMS Administration Console for live trace feedback. SWF verification may also be a culprit; if you're using this feature, be sure that your client SWF file matches the authorized SWF file.</td>
</tr>
<tr>
<td>Server Troubleshooting</td>
<td>The server is slow to respond, and RDP/VNC/SSH is very sluggish.</td>
<td>Check the CPU, Memory, and Network usage. If any of these are high, the server will become slow and unresponsive. It may need to be restarted, or the offending application closed. If the resource usage remains high after a restart, or seems to spike up to maximum instantly, there may be an issue with a specific FMS application or with some unrelated, external program. This should be investigated immediately to resolve the issue, as prolonged high usage may damage the hardware over time. You may need to run hardware tests if all other options are exhausted.</td>
</tr>
<tr>
<td>Server Troubleshooting</td>
<td>Client connections are being accepted, but very slowly, or are never resolved.</td>
<td>Connection issues may be observed when there are network issues such as malfunctioning load balancers or a massive number of simultaneous connection requests. On the client end, there can be Internet connection problems, or a firewall blocking ports. The FMS access.log file can reveal details about connection issues. Logging is turned on by default in FMS 3 and later. FMS Health Check utility (FMSCheck) is another useful in diagnosing connection issues. ActionScript traces of event handler status codes can also be helpful in diagnosing connection problems (onStatus in AS2, netStatus in AS3). You can also try connection monitoring utilities such as Ethereal Wireshark, tcpdump, or Charles. Often, connection “length” can have an impact on connection performance; geographical location, number of hops, and overall latency can all play a part. Note also that RTMP tunneling (RTMPT) connections are generally slower than standard RTMP connections.</td>
</tr>
</tbody>
</table>

**Contacting Adobe**

Adobe support is very useful for professional assistance with your deployment. These guidelines will be followed when you contact the support team:

- All incoming customers are sent to Technical Support.
- If further assistance is needed, the customer is sent to a Worldwide Escalations Engineer for further analysis.
- The issue will be logged in Adobe’s bug reporting system, and escalated to Engineering, where it will be evaluated and handled by the appropriate Engineering department.
- The highest-level inquiries will be handled by Product Management.

Adobe offers several premium support plans from which to choose, including Bronze, Silver, Gold, and Platinum. Visit [http://adobe.com/support/programs](http://adobe.com/support/programs) for features and pricing or contact Adobe sales for more information.

When preparing to contact Adobe support, it is important that you first gather as much information about your issue as possible. The minimum information you should provide includes:
• FMS version and exact build number
• Platform, operating system and service pack (Windows) or kernel (Linux)
• Confirmation that the system requirements for FMS are met (http://www.adobe.com/go/fms_sysreqs)
• Is the problem with a live, VOD or interactive application?

Some additional details that may be helpful include:
• Is this a new or existing deployment?
• Were any changes made recently that may have introduced this issue?
• What is the configuration (load balancers, NIC, Edge-Origin)?
• What are the detailed steps required to reproduce the problem?

It is also very helpful to provide Adobe with log files, crash or hang dumps, performance data, FLA files and ActionScript for client applications, Server-Side ActionScript, and FMS configuration files, if they are available.

Glossary

Basic Technical Definitions

Adobe AIR: A cross-platform tool that enables developers to use their existing web development skills in HTML, Ajax, Flash, and Flex to build and deploy rich Internet applications to the desktop.

Cache smearing: Occurs when the same piece of content is unnecessarily repeated across many servers. For example, ten users want to play a 4 GB video, and are directed to ten different servers by a content-unaware load balancer. As a result, each of these ten servers has to maintain a copy of the 4 GB file, taking up 40 GB of total disk space. Had these 10 users been directed to one server, then they would have taken up only 4 GB space on that server, saving 36 GB of space on the other nine servers. Cache smearing lowers the cache utilization or cache effectiveness due to wasted cache space.

Configuration options: Defined in XML files. These options enable the server administrator to fine-tune the performance and functionality of Flash Media Server.

Content Delivery Network (CDN): Companies that offer streaming services and bandwidth so customers do not need to set up and install servers of their own.

Client: A SWF file running in Flash Player, Adobe AIR, or Flash Lite 3 that connects to Flash Media Server.

Codec: Short for “code/decode,” the format in which a video or audio file is encoded. Flash uses Sorenson Spark, On2 VP6-S, On2 VP6-E, and H.264 codecs for video; Nellymoser, MP3, and AAC for audio. The decoding part a given codec must be present in the player to play back video using that codec.

Connection: When a client is streaming video it consumes one connection. When multiple clients are streaming at the same time, they are referred to as simultaneous connections.

Content: Video or audio data streamed from Flash Media Server.

Digital Rights Management (DRM): Usage rules and protections applied to media to prevent stealing and unauthorized sharing.

Flash Lite 3: The mobile Flash player that supports VP6 and Spark codecs and allows RTMP connections to Flash Media Server.

Flex: Adobe Flex is a cross-platform, open-source framework for creating rich Internet applications that run in Flash Player and AIR. These applications run identically in all major browsers and operating systems.

Flash Media Live Encoder: A free Windows-based desktop application that connects to Flash Media Server, allowing you to stream live video and audio to Flash Player.

Live: Live Flash streaming using Flash Media Live Encoder or Flash Player.
Plug-in: A program designed to extend the capabilities of another program by interfacing with it at the code level. FMS allows for the creation and integration of C++ plug-ins using the built-in C++ API. This allows server administrators and developers to extend the functionality of FMS.

Protocol: A convention or standard that allows communication between two computing endpoints. RTMP is a protocol that allows communication between the SWF client and FMS.

Port: An application- or process-specific communication endpoint that allows a transport protocol to define which service communications should be received by the application or process. FMS communicates over port 1935.

Publishing Point: A directory on Flash Media Server where customers can place video or audio content and publish live video.


RTMPE: Encrypted RTMP, the next-generation Real Time Messaging Protocol (RTMP) that increases security and performance over SSL-encrypted protocols.

Quality of Service (QOS): Refers to stream playback performance, which affects the quality of the consumer’s playback experience.

Security Options: The options in FMS that allow the server administrator to effectively secure the server. The options include SWF verification, referring domain controls, RTMPE, and more.

Server-Side ActionScript (SSAS): A JavaScript 1.5-based language that allows developers to exploit a wide variety of features on the FMS server, and to add a further degree of control. Generally used for interactive, complex applications to enhance functionality.

SSL (Secure Sockets Layer): A cryptographic protocol that provides secure communications over the internet. SSL is utilized by RTMPS to encrypt streams and data.

SWF Verification: Allows the administrator to provide a reference SWF application file to Flash Media Server for the purpose of client application verification. The reference SWF file’s bytecode is compared with the client SWF file that is making a connection request, and if the bytecode matches, a connection is allowed. If the bytecode does not match, the connection is automatically rejected.

Video on Demand (VOD): A term used to describe the delivery of prerecorded Flash video streaming.