



TECHNICAL PAPER

Video content protection measures enabled by **Adobe® Flash® Media Interactive Server 3.5**

Kevin Towes
Senior Product Manager, Flash® Media Server

Tom Green
Professor, Interactive Multimedia
Humber Institute of Technology and Advanced Learning

Updated June, 2010

© 2010 Adobe Systems Incorporated. All rights reserved.

If this white paper 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.

This article is intended for North American audiences only.

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, ColdFusion, Flash, Flash Access, Flash Media Server, Flash Media Interactive Server, and Flash Player are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

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.

SUMMARY

This technical paper introduces you to some of the risks for delivering video on the Internet today. It illustrates how you can help protect your content using the built-in features of Adobe® Flash® Media Interactive Server 3.5 and Flash Media Streaming Server 3.5. Two major features include a new encryption protocol called RTMPE and SWF verification. Both features are available with the streaming and interactive versions of the server.

This paper also provides you with examples of how you can help ensure that your video—with increased protection measures applied—can be available for a large number of people to enjoy on the Internet. If you own Flash Media Interactive Server, you will learn some advanced ways to help protect your content.

TABLE OF CONTENTS

Summary.....	1
Table of contents	1
Introduction.....	2
How your video content can be captured.....	3
Flash Media Interactive Server 3.5 stream protection methods	5
Examples of how video is captured	6
Enable basic video protection with Flash Media Interactive Server 3.5	7
Basic configuration	7
How to change from progressive download to streaming.....	8
No client cache.....	9
RTMP protocol.....	10
SSL encryption	10
RTMPE real-time encryption.....	11
Content protection from Content Delivery Networks.....	11
Content protection with server-side programming	12
User authentication.....	12
Simple client verification using a unique token authentication key	13
User validation through an external resource.....	14
Private token-based system.....	15
The client object.....	16
Referrer and pageUrl check validation.....	17
IP address validation	18
White list domains	18
Flash Player version check	19
Using Client.agent.....	20
Access adaptor	20
SWF verification.....	21
Protecting your content from the “Replay”	22
Content protection.....	24
Media content protection from Adobe.....	25
Online resources	25
About the authors	26

INTRODUCTION

Video on the Internet has exploded in popularity. Video streamed using Adobe® Flash® Player software has raised the bar for content distributors to create a rich video experience. Compelling video content has driven the popularity of Internet-based high-quality video. Content created by everyone, which we call “user-generated content,” and content created by professionals—such as major broadcasters, filmmakers, and advertisers—position the Internet as a significant factor in delivering video today.

This video explosion can also be attributed to the increase in quality and bandwidth. The FLV format, used by Adobe Flash Player and Adobe® AIR, has improved the quality and performance significantly since it was introduced. The success is also due to a whole ecosystem of companies helping people and companies to encode, publish, manage, and deliver video. Adobe partners with many of these companies through our partner programs: Flash Video Streaming Service (FVSS) (adobe.com/go/fvss) and Flash Media Solution Provider (FMSP) (adobe.com/go/fmsp). Check them out.

With more and more compelling content online today, there are requirements to ensure that your video is protected from misuse or repurpose. You may not think that your home videos on a social media provider are important enough to worry about, but consider what people could do to your “innocent” home movies. If you are trying to monetize your video, you wouldn’t want people figuring out a way to remove the ways for you to profit. Even worse, someone could be making money from your video without your authorization.

The easiest way to help protect your content is to stream it.

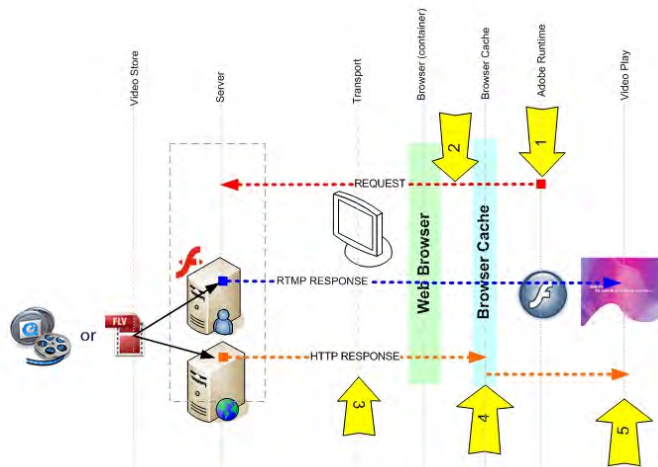
Streaming through Adobe Flash Media Interactive Server 3.5 is one easy method to protect your content. Built into Flash Media Interactive Server 3.5 are useful ways to ensure that your video is available to a wide audience and to control the experience and the actual video files.

Flash Media Development Server 3 is available for FREE today at adobe.com/go/fms. Included in this full-featured version are solutions that help to protect your content on the Internet but allow a limited group of people to watch the content at the same time.

HOW YOUR VIDEO CONTENT CAN BE CAPTURED

Before we review how to help to protect your video content, let's review ways that content can be captured today. The diagram below shows the end-to-end life of a video file from the point of delivery to playback. This list applies for all video formats including Windows Media (WMV), Apple QuickTime (MOV), MPEG-4 (MP4), and even Flash (FLV).

WHERE VIDEO IS CAPTURED



Lifetime of a video streamed on the Internet today

Video delivery options:

- True streaming (RTMP/RTSP)
- Progressive streaming/download (HTTP)

- 1) **Connection requests.** Capture technology listens for connection requests, and logs them. Later, they replay the connection request and capture the stream to a local disk, breaking the natural security received from streaming.
- 2) **Web browser.** "Listener" technology monitors the data flow between Flash Player and the network.
- 3) **Data transfer.** Capture technology listens for video formats being transferred and starts recording the bits transferred.
- 4) **Progressive download.** Video is captured from temporary Internet files (browser cache) and presented to the user for offline playback. Today, the majority of stream rippers use this method.
- 5) **Screen capture.** Screen capture technology record still frames or limited motion from the monitor.

METHOD OF MISUSE	HOW IT WORKS
Raiding the browser cache	<p>Video that is streamed progressively (progressive download) is cached to your web browser's disk cache. Just like HTML, JPEG, GIF, and even SWF files, video is stored in a temporary folder so that it's easier to access the second time. This technique is great for improving your web browsing experience. Video streamed progressively is actually downloaded to the computer requesting it.</p> <p>How do you know if your video is progressive? If your video is served from a web server, it is delivered progressively. If your video is in the same location as other web files—such as images, HTML, or other downloadable files—there is a likelihood that if you are using this method, someone has your video on his or her computer.</p>
Video URL access	<p>Capturing video can be easier if you expose the URL of the video. This reference is the online address of your progressive video. The typical place for this exposure is within the HTML of your web page. Technologies can copy this location and essentially cue up a download through typical HTTP or RTSP capturing.</p>
SWF re-serving	<p>The indirect way of ripping your video is to copy the SWF file that contains the video requests and reserve it from a different website or domain. A potentially more harmful situation is if someone could take the SWF file and learn where your content is and how to misuse it.</p>
Replay technologies	<p>Technology companies can leverage complex methods of network listening and other adverse techniques to misuse the "bits" as they are transmitted from server to client.</p> <p>Open protocols such as HTTP and RTSP make this easier. Traditionally these risks were not a major concern but now because of all that compelling video content online, the risk is increasing every day. There is a demand for people to acquire content from a server using these techniques.</p> <p>Companies that are currently offering this technology include:</p> <ul style="list-style-type: none"> • RealNetworks • Applian Technologies • Sothink Media

FLASH MEDIA SERVER 3.5 STREAM PROTECTION METHODS

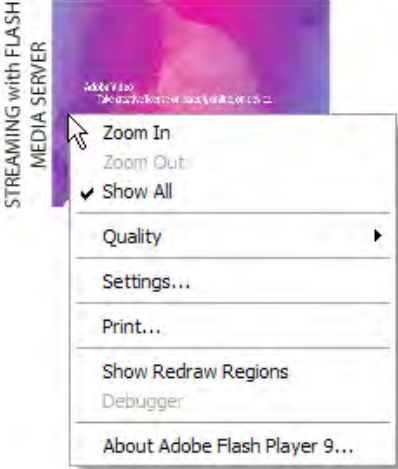
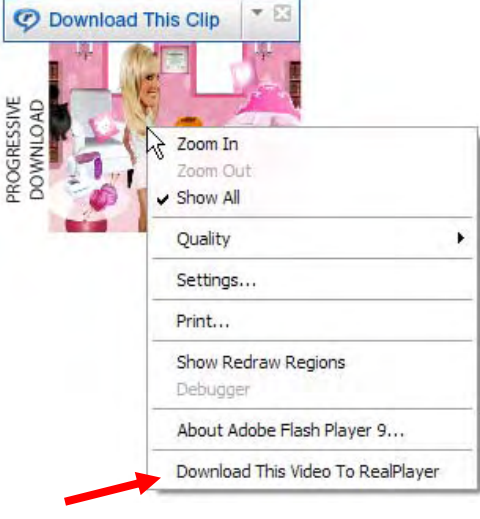
To start, let's first look at how Flash Media Interactive Server 3.5 addresses each of these methods of video misuse.

METHOD OF MISUSE	HOW FLASH MEDIA SERVER 3.5 MAY PROTECT YOUR CONTENT
<p>Raiding the browser cache</p>	<p>Streaming with Flash Media Interactive Server 3.5 does not download video to a browser cache. Instead, a buffer is created in the protected memory of Flash Player. When video bits have been viewed, they are discarded, making room for the next series of video bits.</p> <p>Flash Media Server offers pure video streaming on demand and live. This means that if your video content is 20 minutes long, it will take a little less than 20 minutes for your video to be delivered, depending on the size of the video buffer you set.</p>
<p>Video URL access</p>	<p>If someone were to misuse the URL or your video streamed from Flash Media Server, you could help to protect that video through special scripting on the server. The scripting is as basic as ActionScript or as complex as C++. It's really your choice.</p> <p>Because Flash Media Server does not use HTTP or RTSP, you have assistance against a larger array of video capture software currently available.</p>
<p>SWF re-serving</p>	<p>Flash Media Server can be configured to check that a SWF file is coming from the correct location, and can potentially block connection requests when the locations do not align.</p>
<p>Replay technologies</p>	<p>Flash Media Interactive Server 3.5 has new features to help protect against some of the currently available technologies because video is streamed using the Adobe proprietary protocol, RTMP, which supports custom scripting. The latest version also supports a fast, real-time encryption protocol, RTMPE. When combined with the new SWF verification feature, Flash Media Server provides the fundamental tools to help protect your content from the replay.</p> <p>Simply by streaming video from Flash Media Server versus progressively through a web server, you gain additional protection against companies like RealNetworks, Applian Technologies, and Sothink, whose technologies could make copies of your video content. By adding server-side scripting with Flash Media Server, you further help prevent content from being captured.</p> <p>If you need to add even more protection to the transfer of your video from server to client, you can enable 128-bit encryption using SSL. Alternatively, you can either encrypt the stream or use SWF verification to guard against these issues.</p>

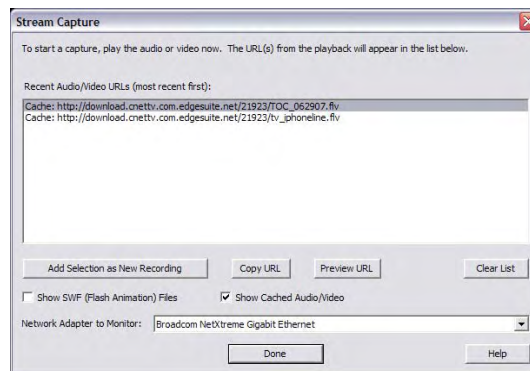
Examples of how video is captured

The following example shows how progressive downloaded video can be captured from disruptive technologies.

The video on the left is streamed from Flash Media Interactive Server 3.5. Notice how there is no way to download the video. The example on the right shows an option to download the video because it is a progressive download.

WITH FLASH MEDIA INTERACTIVE SERVER 3.5	WITHOUT FLASH MEDIA INTERACTIVE SERVER 3.5
	
<p><i>Video streaming to embedded Flash Player from Flash Media Interactive Server 3.5 can be unaffected from technologies like RealNetworks download routines.</i></p>	<p><i>Video streamed progressively to embedded Flash Player from a web server can be misused by RealNetworks download routines.</i></p>

The example below shows how a technology can access the browser's disk cache and then report the video files that are available. Video is available because it has already been downloaded. These tools enable the consumer to capture the files and then store them in a different location so they can be played back later.



© Applian Technologies

ENABLE BASIC VIDEO PROTECTION WITH FLASH MEDIA INTERACTIVE SERVER 3.5

Enabling basic video protection is easy with Flash Media Interactive Server 3.5. You don't need to change the encoding of your video or do anything at all. You can enhance your current video protection by downloading and installing Flash Media Interactive Server 3.5.

FLASH MEDIA INTERACTIVE SERVER 3.5 EDITION	WHAT IT CAN DO
Flash Media Development Server 3 (free)	<i>Full interactive features (up to 10 users)</i>
Flash Media Interactive Server	<i>High-volume streaming</i>
Flash Media Streaming Server	<i>Delivery of live and video-on-demand applications only</i>
Content Delivery Network (CDN) *	<i>High-volume and capacity-managed streaming</i>

* Available from an Adobe-authorized Flash Video Streaming Service provider

This section introduces you to the out-of-the-box content protection features that Flash Media Interactive Server 3.5 offers to help ensure that you know where your video is. Take a look at some of the protection features that Flash Media Interactive Server 3.5 offers:

- **Basic configuration** → Start streaming with Flash Media Server 3.5 and you're covered
- **No client cache** → Unlike progressive download, streaming has no client cache
- **RTMP protocol** → Unlike RTSP or HTTP, Flash Media Interactive Server 3.5 uses RTMP
- **RTMPE protocol (new)** → Proprietary protocol encrypts the stream
- **SWF verification (new)** → Matches currently playing SWF against the original and, if changes are detected, denies the stream
- **SSL encryption** → Encrypt the communication channel from server to client
- **CDN/platform protection** → CDNs offer advanced authentication including tokens

Basic configuration

Out of the box, Flash Media Interactive Server 3.5 is ready to go. All you need to do is create a publishing point, place your video files on it, and start streaming. Just by streaming video, you will help to protect against many of the capture technologies currently available.

After installing Flash Media Interactive Server 3.5, all you need to do is copy your video:

- 1) Browse to C:\Program Files\Adobe\Flash Media Server 3\applications\vod\media
- 2) Place all of your FLV or mp4 files in this folder and you are also ready to go.

You also have the option to connect to a remote file location, called VirtualDirectory. See the documentation at <http://livedocs.adobe.com/fms/2/docs/00000421.html>.

How to change from progressive download to streaming

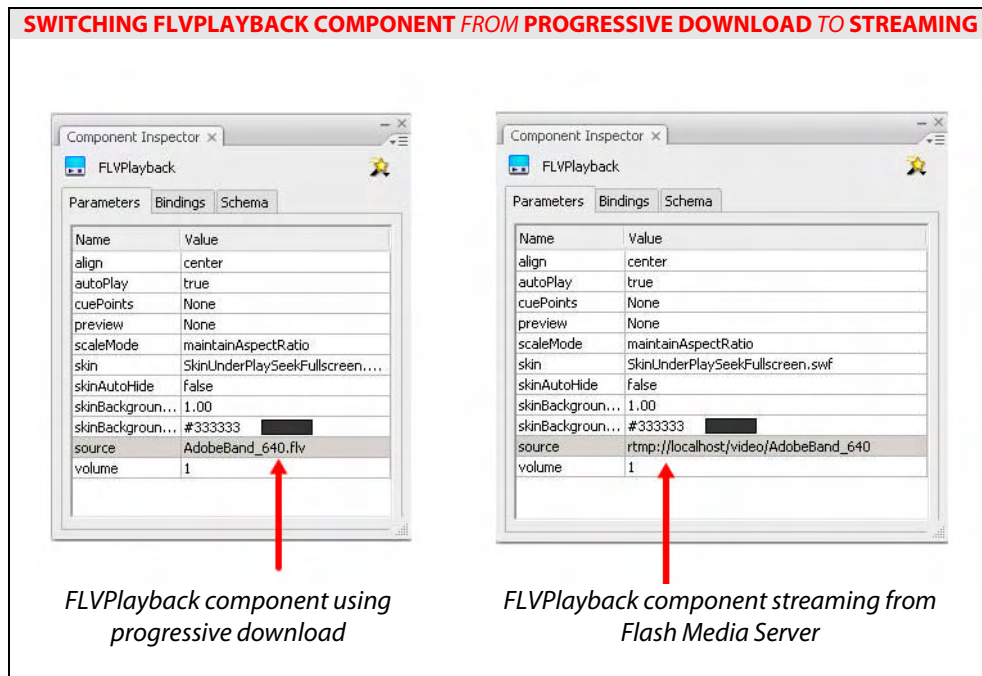
Now that you have the server set up and your video deployed you need to change your video player application in Flash CS3.

If you use progressive download video, you can do either of the following:

- Use the FLVPlayback component
- Have written something from scratch using ActionScript

For the FLVPlayback component, it is easy to change from progressive streaming to FMS streaming:

- 1) Open Flash CS3 Professional and select your FLVPlayback component.
- 2) In the Component inspector, locate the `contentPath` property (see figure below).
- 3) Change the content path to point to your Flash Media Server (include your video file, without the FLV extension): `rtmp://localhost/vod/AdobeBand_640`
- 4) Save and compile, and enjoy basic protected content streaming.



For the ActionScript method, follow these instructions:

- 1) Locate your `NetConnection.connect(null);` command. (This is what you used to indicate delivery from a web server.)
- 2) Change `null` to the name of your server:
`nc.connect("rtmp://localhost /vod/");`

- 3) Create an `onStatus` function for your `NetConnection` handling the Connection events:

```
nc.onStatus = function(pStatus:Object):Void {
    if (pStatus.code == "NetConnection.Connect.Success") {
        initStreams();
    }
}
```

assuming that `initStreams` is a function that wraps the `NetStream` class.

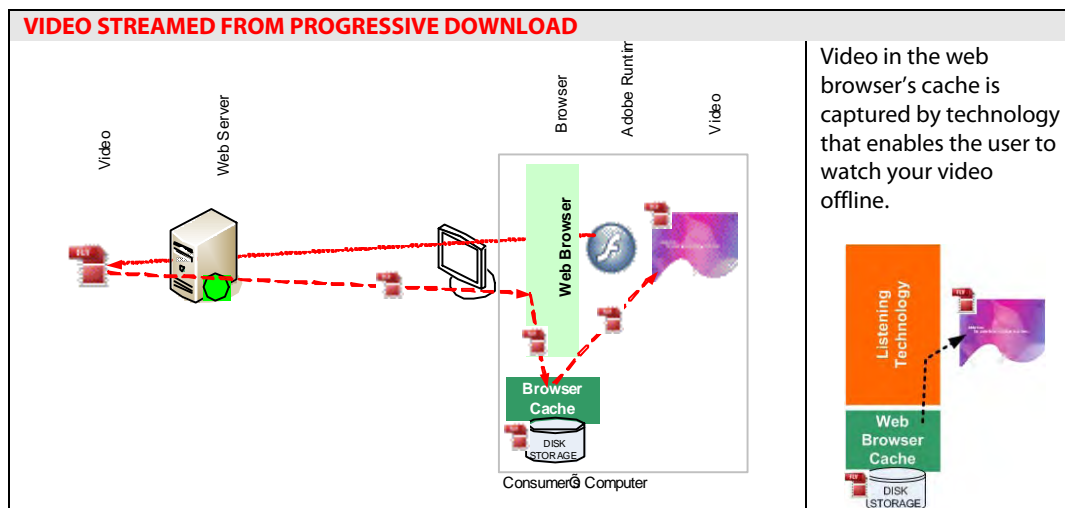
- 4) Execute the `NetStream` command after the connection has been established (within the `onStatus` function in Step 3 above).

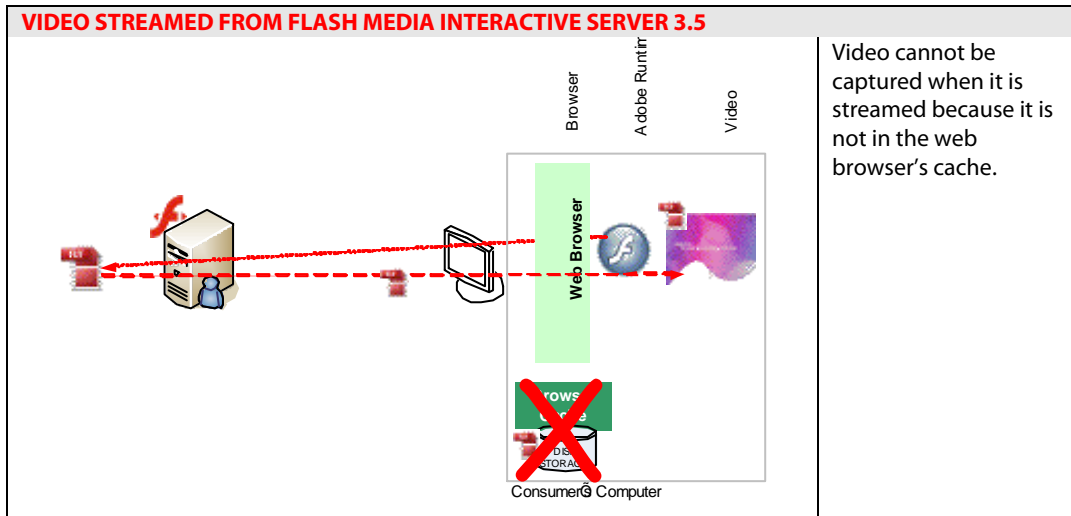
Everything else is the same.

No client cache

Progressive delivery (from a web server) actually downloads video to the hard disk. Video is requested exactly the same as requesting an image in a web page. A principal feature of Flash Media Interactive Server 3.5 streaming is that there is *no* client cache. When video is streamed from Flash Media Interactive Server 3.5, the bits are destroyed after they've been viewed. Accessing the client cache is an easy way for technologies to capture video and make it distributable.

The following diagrams illustrate the difference between progressive download/streaming and streaming video from Flash Media Interactive Server 3.5.





RTMP protocol

Why are we mentioning the RTMP protocol as part of a solution for content protection? RTMP is the proprietary protocol of Adobe and Flash Media Interactive Server 3.5. Unlike other streaming protocols like RTSP (<http://en.wikipedia.org/wiki/Rtsp>) or HTTP (<http://en.wikipedia.org/wiki/Http>), the RTMP protocol is proprietary, which makes it difficult for technologies to misuse the protocol and capture content streamed over it.

The native port (1935) that RTMP uses is an IANA-registered port for both TCP and UDP traffic. IANA is the Internet Assigned Numbers Authority (see http://en.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority). RTMP is bidirectional, which allows for an enhanced video experience, but if you stream with RTMP you will be more protected than if you stream with HTTP progressively from a web server.

SSL encryption

To help protect the transport of your video from Flash Media Interactive Server 3.5 to Flash Player, you can enable encryption. Flash Media Server can be configured with an SSL certificate and will provide up to 128-bit encryption strength. When you use SSL encryption, the protocol will become RTMPS. The RTMPS protocol actually uses HTTPS and usually communicates over port 443. SSL encryption requires Flash Player 8 or later.

To configure Flash Media Server to operate in SSL mode, you must configure a port on which it can listen. This is configured in `conf_defaultRoot_Adaptor.xml`. The tag `HostPort` describes all the ports that Flash Media Server will listen on. Ports defined with a minus sign will become the SSL ports. Here is an example:

```
<HostPort>:1935,80,-443</HostPort>
```

SSL can be configured at the server and the adaptor level. Private key files can be encrypted or clear and the `passphrase` required to open it can be added to the configuration file.

What Adobe provides, through this method, are the APIs and software that allow you to create custom authentication schemes such as validating the SSL connection before the content streams. When a connection is made, there is a basic handshake between the server and the client but you need to supply the authentication business rules.

Recognizing that no two SSL encryption scenarios are the same, Adobe provides only the mechanism for you to create customized authentication schemes.

For clients to connect using SSL, the URI string required will look like this:

```
rtmps://localhostlocalhost/video/AdobeBand_640
```

Detailed information on implementing SSL can be found in the documentation:

<http://livedocs.adobe.com/fms/2/docs/00000517.html>

RTMPE real-time encryption

This new method to help protect your stream is available starting with Flash Player 9,0,115,0. This encrypted protocol prevents third-party applications that have been developed to listen to the data transfer between the server and the client. In certain instances, this software can be used to “rip” the video from the stream. This feature is available by default, but using this feature is optional. If you choose to encrypt the data stream, you would use the following syntax:

```
nc.connect("rtmpe://localhost/vod/");
```

If you are using the source parameter of the FLVPlayback component, the path would be:

```
rtmpe://localhost/vod/flvFileName
```

By adding the letter “e”, you tell Flash Media Server to add real-time encryption to the stream. The file is encrypted as it moves from the server to the client and requires no keys to decrypt the file. Unlike the SSL solution, the RTMPE protocol can be terminated only by Flash Player. If an application uses RTMPE without specifying a port, Flash Player will scan the ports in the following order: 1935 (RTMPE), 443 (RTMPE), and 80 (RTMPE).

By default, RTMPE is enabled. To disable it, open the fms.ini file located in C:\Program Files\Adobe\Flash Media Server 3\conf and set the ADAPTOR.RTMPE_ENABLED parameter to off. When making this change, please note the server must be restarted.

Content protection from Content Delivery Networks

Another option that can add content protection to your video streaming is to use Adobe Flash Video Streaming Services through the Adobe Content Delivery Network (CDN) partners. Many of the Adobe FVSS partners offer restricted access streaming solutions and secure video.

To learn more about how a Content Delivery Network can help protect your content, please visit the Flash Video Streaming Service website at adobe.com/go/fvss.

Ask your CDN partner about protected streaming using RTMPE.

CONTENT PROTECTION WITH SERVER-SIDE PROGRAMMING

Flash Media Interactive Server 3 can be programmed using ActionScript 1.0 on the server. The ActionScript code is located in a file called main.asc. Both versions of Flash Media Server 3.5 ship with a prebuilt main.asc file. If you own Flash Media Interactive Server, you can change this file; if you own Flash Media Streaming Server, you are restricted from changing this file. Within the file, you can accept or reject a connection based on numerous challenges and conditions, including custom properties such as a login from the client or from information made available automatically such as the SWF filename.

This section explores methods to help protect your content including:

- User authentication
- Simple client verification
- Validation through external resources
- The client object
- The access adaptor

User authentication

There are numerous ways that you can authenticate a user with Flash Media Interactive Server 3 to help ensure that your content is distributed how you intended it. When a connection is made with Flash Media Interactive Server 3, data from the client to the server can be passed during the connection process. This is done by adding parameters to the `NetConnection.connect()` method. The first parameter is always the server location. Any parameter that follows is completely up to you.

This data can be challenged on Flash Media Interactive Server 3 to do either of the following:

- Ensure that your user is who they say they are
- Defend against content misuse and replay technologies

The information you can pass through the connection process could include the following:

- **User credentials** (login/password):
`NetConnection.connect("rtmp...", "kevin", "password");`
- **Secure hash token** (e.g. SHA-256 hash):
`NetConnection.connect("rtmp...", 6aef79f07bc8f23c38e8979f3630f436);`
- **Unique key**:
`NetConnection.connect("rtmp...", 349jh3k4324h9.234234098);`

The powerful ActionScript API gives you four ways to challenge credentials with external resources. External resources could validate the connection request against a database, LDAP server, or other access-granting service.

The external APIs available in Flash Media Server include:

- Web services (SOAP)
- Flash Remoting (NetServices)
- HTTP Post (LoadVars)
- XML Post
- File Read

Simple client verification using a unique token authentication key

This section introduces a simple client verification technique to help you increase your defense for replay technologies and non-authorized connections with Flash Media Interactive Server 3.

The client-side ActionScript creates a unique key. In this example, the key is made up of the millisecond time on the computer combined with a random number. That key is sent through the `NetConnection.connect()` method as the second parameter.

CLIENT-SIDE ACTIONSCRIPT

```
// Create a uniqueKey string for this client
var rNumber:String = String(Math.random());
var rDate:String = String(new Date().getTime());
var uniqueKey:String = rDate + rNumber;

// send the uniqueKey string to FMS
nc.connect("rtmp://server/secure1/", uniqueKey);
```

The server-side ActionScript receives the client data through the second argument, `uniqueKey` in the `Application.onConnect()` handler. If no unique key is found, the connection is rejected. The unique key is used as an index in an array. If the index already exists, the connection is rejected. This helps prevent replay technologies from capturing the connection sequence and replaying it.

SERVER-SIDE ACTIONSCRIPT

```
// this will store references of all clients, and ensure there are no replays
clientKeyList = new Object();

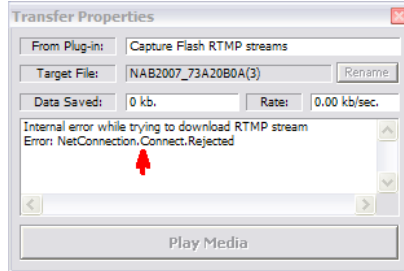
application.onConnect = function(pClient, uniqueKey) {

    if (uniqueKey != undefined) { // make sure there is always a uniqueKey
        if ( clientKeyList[uniqueKey] == undefined ) {

            //this client has never connected -- allow the connection
            pClient.uniqueKey = uniqueKey;
            clientKeyList[uniqueKey] = pClient;
            this.acceptConnection(pClient);
        } else {
            trace("Connection Failed");
            this.rejectConnection(pClient);
        }
    }
}

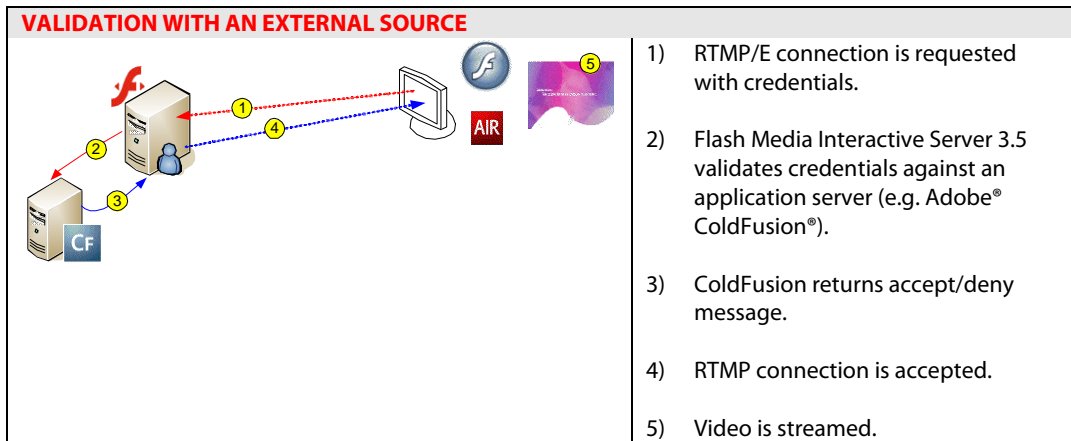
application.onDisconnect = function(pClient) {
    //clean up the keys
    delete clientKeyList[pClient.uniqueKey];
}
```

The `onDisconnect` handler will clear the client index when the connection is lost. The following screen shot shows an example of how this routine can help prevent replay technologies from capturing your video stream. The message `NetConnection.Connect.Rejected` is received because the technology cannot connect to your server and capture your video stream.



User validation through an external resource

This same technique as simple client verification could also be used to send in authentication credentials:



The client-side ActionScript can pass in login/password information through the `NetConnection.connect()` method:

CLIENT-SIDE ACTIONSCRIPT

```
// Create a uniqueKey string for this client
var sUsername:String = "myUsername";
var sPassword:String = "myPassword";

// send the credentials string to FMS
nc.connect("rtmp://server/secure1/", {username:sUsername, password:sPassword });
```

On the server running Flash Media Interactive Server 3, you can implement server-side ActionScript that accepts the credentials and challenges them against an external service through SOAP web services, Flash Remoting, XML, or an HTTP post.

The following server-side ActionScript is a template demonstrating how you can leverage the API to authenticate the client against an external service. The ActionScript will place the connection into a pending state until the results of the challenge are returned from the remote service.

```
SERVER-SIDE ACTIONSCRIPT
load("NetServices.asc"); // used for Flash Remoting
load("WebServices.asc"); // used for SOAP web services

pendingConnections = new Object();

application.onConnect = function(pClient, pUserName, pPassword) {
    // create a unique ID for the client
    pClient.FMSid = application.FMSid ++;

    // place the client into a pending array
    pendingConnections[FMSid] = pClient;

    if (pUserName!= undefined && pPassword !=undefined) {
        // issue the external call (3 examples below)
        loadVars.send("http://url?login=" + pUserName + "?password"+pPassword +
            "?FMSid"+FMSid);
        webService.authenticate(FMSid, pUserName, pPassword);
        netService.authenticate(FMSid, pUserName, pPassword);
    }
}

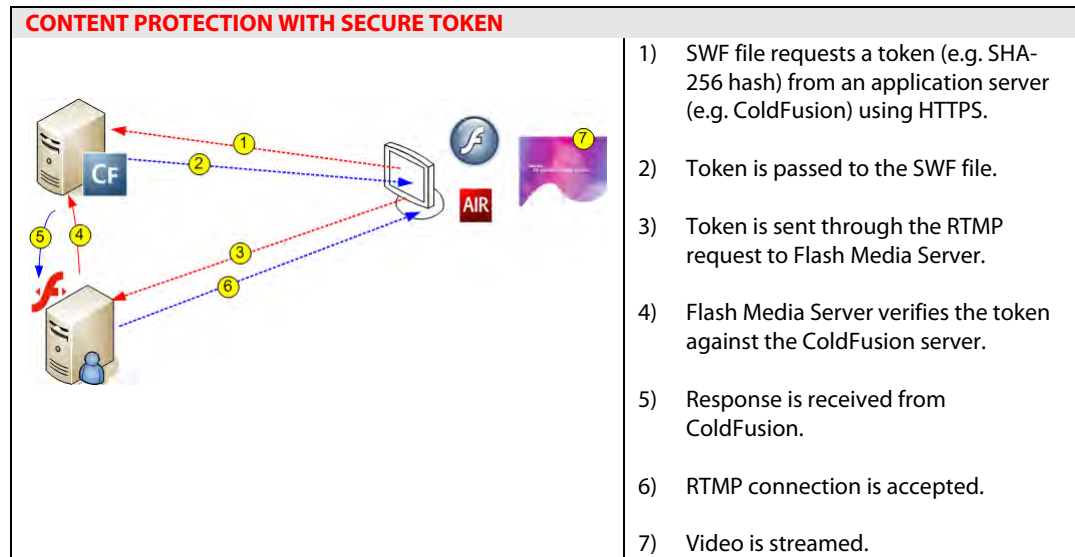
// the result handler (sample only, you will have to customize this)
// this command will return a true/false and the FMS client id
Authenticate.onResult =
loadVars.onData = function(FMSid,pData) {
    if (pData)
        application.acceptConnection( pendingConnections[FMSid] );
        delete pendingConnections[FMSid];
    } else {
        application.rejectConnection( pendingConnections[FMSid] );
        delete pendingConnections[FMSid];
    }
}
}
```

Private token-based system

To add security to access Flash Media Interactive Server 3 streams, you can use secret, time-based tokens delivered via secure hash such as SHA-256 (e.g. 6aef79f07bc8f23c38e8979f3630f436) can be used to request connections to Flash Media Server. To use this technique, the SWF file requests a secure ticket and then passes it with the connection request.

This feature is available only through Flash Media Interactive Server and requires Flash Player 6 or later.

The following diagram illustrates how a token-based system can be configured:



Read the following article to learn more about how you can implement a private token:

Using tickets and Flash Remoting to transmit secure information

adobe.com/devnet/flashcom/articles/ticket/fcs_secure_ticket.pdf

The client object

Each time someone connects to Flash Media Interactive Server 3, you have a chance to authenticate the connection. This authentication may protect you against unauthorized software clients such as SWF or other RTMP clones. If your video URL is compromised, the following techniques may restrict the connect request before a stream can even be accessed.

The following code is an example how to access the client data in server-side ActionScript.

SERVER-SIDE ACTIONSCRIPT

```
application.onConnect = function(pClient) {  
    for(var i in pClient)  
        trace('key: ' + i + ', value: ' + pClient[i]);  
}
```

One of the most effective ways to authenticate access is to use the Client object. The following properties are available each time a client makes a connection request to Flash Media Server.

Agent	WIN 9,0,115,0
IP	127.0.0.1
readAccess	/
writeAccess	/
Referrer	http://localhost/SimpleConnect.swf
Protocol	rtmp
URI	rtmp://towsfms.adobe.com/secureTest/
Secure	False
pageUrl	http://localhost/SimpleConnect.html
virtualKey	

The Client object can be accessed in server-side ActionScript in the `application.onConnect()` event handler. You can challenge properties in the Client object to protect your content in one of four ways:

- Flash Player version check
- Referrer and pageUrl validation
- IP address validation
- Virtual keys

Referrer and pageUrl check validation

If someone were to run your SWF file (Flash movie) on their website and the routines for playing video were inside the SWF, or within the embed/object tag, then you could end up seeing your video in places you never expected.

To protect against this misuse, there are two methods that may ensure that the SWF file is coming from the correct place: the `Client.referrer` property and the `Client.pageUrl` property.

Here is an example of how to use them. In your main.asc file integrate the following ActionScript:

```
SERVER-SIDE ACTIONSSCRIPT
var VALID_REFERRER = "http://localhost/SimpleConnect.swf";
var VALID_PAGEURL = "http://localhost/SimpleConnect.html";

application.onConnect = function(pClient) {
    if (pClient.referrer == VALID_REFERRER &&
        pClient.pageUrl == VALID_PAGEURL) {
        this.acceptConnection(pClient);
    } else {
        this.rejectConnection(pClient);
    }
}
}
```

IP address validation

If a client is trying to access your video from unauthorized or banned computers, you can restrict the client's IP address. Additionally, if multiple requests are coming from the same IP address very quickly, this can be an indication that some unauthorized activity is going on.

To protect against this misuse, you can validate the IP address of the client or put a temporary block on that IP address. A full block on an IP address may cause you problems with virtual IP addresses, so you could use a delay timer on connections from a single IP address. An effective example to block replay technologies is to have a unique identifier sent when the connection is made. Then validate that only a single unique identifier can connect at any time.

The following sample shows how a file listing banned IP addresses can be used to block IP address requests. The banned IP list is an external file that allows it to be modified in real time.

SERVER ACTIONSCRIPT	BANNEDIPLIST.TXT
<pre>function getBannedIPList() { var bannedIPFile = new File("bannedIPList.txt"); bannedIPFile.open("text","read"); application.bannedIPList = bannedIPFile.readAll(); bannedIPFile.close(); } application.onAppStart = function() { this.blockINT = setInterval(getBannedIPList, 30000); getBannedIPList(); } application.onConnect = function(pClient) { var isIPOK = true; for (var index=0; index<bannedIPList.length; index++) { var currentIP = this.bannedIPList[index]; if (pClient.ip == currentIP) { this.isIPOK = false; break; } } if (isIPOK) this.acceptConnection(pClient); else this.rejectConnection(pClient); }</pre>	<pre>192.168.0.1 128.493.33.0</pre>

White list domains

Flash Media Interactive Server 3 can be configured to restrict access to specific domains or IP addresses (also known as the *white list*). This is done in two configuration layers: Adaptor.xml and vHost.xml. The adaptor configuration lets you set a *black list* to deny requests from specific domains. You can add a comma-delimited list of domains and/or addresses in the Adaptor.xml or vHost.xml configuration files. Black lists are set using the <deny> tag in the configuration files. The vHost layer enables you to specify only domains that you allow.

Example #1 allows all connections except those connecting from `outlaw.adobe.com`:

1: CONFIGURE ADAPTOR.XML TO USE ALLOW/DENY RANGES

```
<allow></allow>
<deny>outlaw.adobe.com</deny>
```

Example #2 allows only connections from `adobe.com`, `macromedia.com`, and `allaire.com`:

2: CONFIGURE ADAPTOR.XML TO USE ALLOW/DENY RANGES

```
<allow>macromedia.com,adobe.com,allaire.com</allow>
<deny></deny>
```

Using this configuration will help you protect your content from unauthorized access without any server scripting.

Flash Player version check

Protecting against non-Flash Player clients or rogue clients is another method to protect your streams while also providing a better quality of service in the process. You can grant or deny access to the server based on the user agent string sent when a client connects.

When a client connects to Flash Media Server, it sends a string that identifies the platform and Flash Player version. Examples of these strings include:

- WIN 8,0,0,0
- MAC 9,0,45,0

You have two options to access these strings:

- **Client.agent:** Challenge the connection to Flash Media Server using ActionScript
- **Virtual keys:** Configure the server to remap the stream based on the Flash Player client

Virtual keys can be explored in the documentation or on Live Docs (<http://livedocs.adobe.com/fms/2/docs/00000423.html>).

Using Client.agent

You can use the same technique as Referrer and `pageUrl` to grant or deny access to specific Flash Player versions. The following ActionScript can be used to access each of the properties.

```
SERVER-SIDE ACTIONSCRIPT

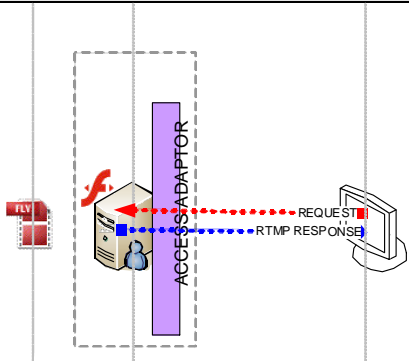
application.onConnect = function(pClient) {

    var platform          = pClient.agent.split(" ");
    var versionMajor     = platform[1].split(",")[0];
    var versionMinor     = platform[1].split(",")[1];
    var versionBuild     = platform[1].split(",")[2];
}

// Output Sample
// Client.agent:    WIN 9,0,45,0
// platform[0]:    "WIN"
// versionMajor:   9
// versionMinor:   0
// versionBuild:   45
```

Access adaptor

Flash Media Interactive Server 3.5 functionality and security can be extended with a plug-in architecture. The adaptor is used to provide greater protection to the Flash Media Server services on your server. Written in C++, the `access.dll` runs the Edge process while the File and Auth adaptors attach themselves to the `FMSCore.exe` service and handle connection routines before the application.

FLASH MEDIA INTERACTIVE SERVER 3.5 ACCESS ADAPTOR PLUG-IN	
	<p>The Access adaptor is a C++ plug-in module for Flash Media Interactive Server 3.5 that intercepts all connection requests and can accept/deny requests before passing to the Flash Media Server core service.</p> <p>The adaptor can validate against external applications such as SQL or LDAP.</p> <p>It is useful to prevent sites from deep-linking or attacking your server.</p>

For more information on the Access adaptor, visit the Adobe LiveDocs site entry for `Access.dll` at <http://livedocs.adobe.com/fms/2/docs/00000513.html>.

The Sample adaptor can be found at <http://livedocs.adobe.com/fms/2/docs/00000515.html>.

SWF verification

SWF verification is another powerful, new method of helping to protect your SWF file and the information within it from decompiling and changes. This feature is available in both Flash Media Streaming Server 3 and Flash Media Interactive Server 3. When you enable this feature, a copy of the original SWF is placed on the Flash Media Server. Each time the SWF plays, Flash Media Server compares the structure of the SWF currently playing in the user's browser with the one on the server. If any changes are detected whatsoever, the server will deny the stream to the client.

SWF verification requires Flash Player 9,0,115,0 or later, or Adobe AIR, and will work with both the streaming and interactive versions of Flash Media Server.

SWF verification is relatively easy to enable. A copy of the SWF to be verified is placed in a folder at the root level of the application and the Application.xml document found in `C:\Program Files\Adobe\Flash Media Server 3\conf_defaultRoot_defaultVHost_` is opened. By default, the name of this folder is `SWFS`. You can create a single folder or a semicolon-delimited list of folders containing copies of the client files.

To verify SWF files for application instances, create instance folders in the `SWFS` folder. Keep in mind that the SWF files found in instance folders will be verified only for that specific instance.

You need to open this document because, by default, SWF verification is disabled. Locate the `<SWFVerification enabled = "false">` tag and change the value to `true`. You will need to restart the server after you have made your changes.

One other feature of SWF verification is the ability to tell the client how long the SWF will live in the memory cache. This duration is measured in minutes. The default value in the Application.xml document is 1440 minutes (24 hours). The value is found between the `<TTL></TTL>` tags (TTL stands for "Time To Live"). You can also configure how often Flash Media Server checks for updated, verified SWF files.

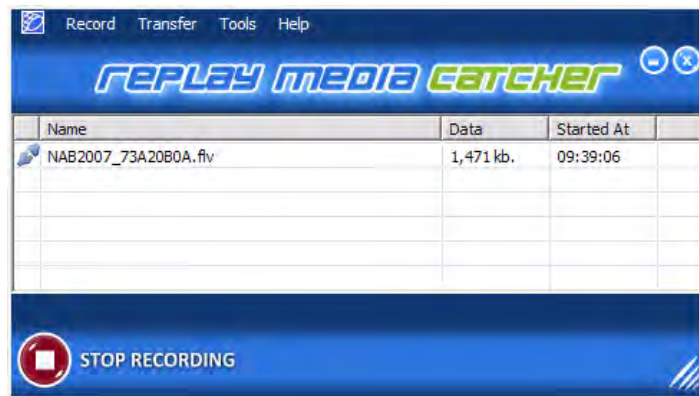
You can find further information regarding SWF verification in the Configuration and Administration Guide found in the Flash Media Server documentation folder.

PROTECTING YOUR CONTENT FROM THE “REPLAY”

The best way to help protect your content against the “replay” is by using RTMPE plus SWF verification. You can go even further, however, using some server-side ActionScript with Flash Media Interactive Server 3.

Companies like Applian Technologies use listening techniques to listen to network messages from Flash Player, and then record it. The technology will “replay” the network messages and will record the video bytes to disk when they are returned. Flash Media Server is “spoofed” and it appears as the same connection as the authorized player.

The protection routines discussed earlier in this paper do not prevent this spoofing. The following image shows the technology capturing the video stream.



There *are* ways to help prevent this. Your first defense is SSL or the new RTMPE protocol. By encrypting the connection between server and client, or encrypting the stream itself, this technology will most likely not succeed.

Your second defense is to verify that the client is authorized to play the video back. To do this, you need to place an additional line of ActionScript in your video player. This ActionScript will respond to a request from Flash Media Server to verify a unique string sent from the server.

Place this function property as an extension of your NetConnection class instance:

```
nc.verifyClient = function(pKey:Object):Object { return pKey; }
```

Your complete client-side code may look like this:

CLIENT-SIDE ACTIONSCRIPT

```
var nc:NetConnection = new NetConnection();
nc.onStatus = function(pStatus){ trace(pStatus.code); };

// RTMP Ripper protection
nc.verifyClient = function(pKey:Object):Object { return pKey; }

nc.connect("rtmp://localhost/onDemand/");
```

Next, place some ActionScript in your main.asc file on Flash Media Interactive Server 3. This ActionScript will ask for the client to verify itself after it has connected. If the client doesn't respond within your set timeout, then the connection is closed by the server.

Integrate the following server-side ActionScript to your main.asc file. If you do not have a main.asc file, copy the following code and save it into a file called application\ondemand\main.asc, where onDemand is your application on Flash Media Server.

SERVER-SIDE ACTIONSCRIPT

```
application.VERIFY_TIMEOUT_VALUE = 2000;

Client.prototype.verifyTimeOut = function() {
    trace(">>>> Closing Connection")
    clearInterval(this.$verifyTimeOut);
    application.disconnect(this);
}

function VerifyClientHandler(pClient) {
    this.onResult = function(pClientRet){
        // if the client returns the correct key, then clear timer
        if (pClientRet.key == pClient.verifyKey.key){
            trace("Connection Passed");
            clearInterval(pClient.$verifyTimeOut);
        }
    }
}

application.onConnect = function(pClient) {

    this.acceptConnection(pClient);

    // create a random key and package within an Object
    pClient.verifyKey = ({key: Math.random() });

    // send the key to the client
    pClient.call("verifyClient",
        new VerifyClientHandler(pClient),
        pClient.verifyKey );

    // set a wait timer
    pClient.$verifyTimeOut = setInterval(pClient,
        "verifyTimeOut",
        this.VERIFY_TIMEOUT_VALUE,
        pClient);

}

application.onDisconnect = function(pClient) {
    clearInterval(pClient.$verifyTimeOut);
}
```

This implementation helps protect against RTMP stream rippers and is customizable. To further assist in protecting your video streams, consider customizing this routine to match your installation or adding SSL to the connection or using the RTMPE protocol.

A final solution is to use the SWF verification features of Flash Media Interactive Server 3.5 presented earlier in this paper.

Content protection

The two fundamentals of DRM (digital rights management) are encryption and access control. As you have discovered throughout this document, there are only two methods of delivering video to a user over the Internet: streaming the content or downloading the content. The first fundamental—encryption—occurs in real time either through the use of the RTMPS (SSL) or the new RTMPE protocols presented earlier. The second fundamental—access control—is easily accomplished using the new SWF verification feature of Flash Media Interactive Server 3.5.

Conversely, you can take advantage of the new plug-in architecture of Flash Media Interactive Server 3, along with a server-side application layer. Using web services (SOAP), Flash Remoting, or even XML, you can create a custom system using secure tokens that provide access control over the content.

MEDIA CONTENT PROTECTION FROM ADOBE

Adobe has numerous ways to help protect your valuable media content. Adobe Flash Media Server provides session-based encryption and access controls including SWF verification. Combined, these technologies provide a great and easy way to ensure that your content is protected.

Adobe Flash® Access™ software is a robust content protection and monetization solution that lets content owners, distributors, and advertisers realize new sources of revenue by providing seamless access to premium content. Flash Access supports a wide range of business models, including video on demand, rental and electronic sell-through. Enterprises can also use Flash Access to protect the integrity and privacy of their training or announcements delivered via video. You can distribute content protected with Flash Access by streaming through Adobe Flash Media Server software, the new HTTP dynamic streaming, progressive download, or permitting downloads to a content library for local playback at the consumer's convenience.

For more information, visit the product page for Adobe Flash Access at adobe.com/go/flashaccess.

ONLINE RESOURCES

Flash Video Learning Guide: Progressive and streaming video

adobe.com/devnet/flash/articles/video_guide_02.html

Scaling and securing streaming media applications with Flash Media Server 3

adobe.com/devnet/flashmediaserver/articles/scaling_securing_fms3.html

DRM and digital media protection with Flash Media Server

adobe.com/devnet/flashmediaserver/articles/digital_media_protection.html

ABOUT THE AUTHORS

Kevin Towes is the senior product manager for Flash Media Server at Adobe Systems and is responsible for defining, delivering, and supporting Adobe streaming video products and services. Before joining Adobe, Kevin spent 13 years working to enable customers with Flash based interactive video streaming solutions using Flash Media Server. His Flash Media Server Live Video work with the Canadian Broadcasting Corporation (CBC) led to an Emmy nomination in 2004.

Tom Green is a professor of interactive multimedia at the Humber Institute of Technology and Advanced Learning in Toronto, Ontario. He is the author of several best-selling books in the area of Flash and Flash technologies. His latest book is *Foundation Flash CS4 for Designers*, coauthored with David Stiller, and he updated *Foundation Flash CS3 Video* with Adam Thomas. Tom has completed DVD videos for Lynda.com and Adobe Systems, and is a partner at Community MX and a regular contributor to Digital-Web.com and Layersmagazine.com. He is also an active member of the Adobe Community Experts Group, speaking at conferences and seminars around the world and contributing regularly to the Adobe Developer Connection in the areas of Flash authoring and video technologies.



Adobe Systems Incorporated
345 Park Avenue, San Jose, CA 95110-2704 USA
www.adobe.com

Adobe, the Adobe logo, Adobe AIR, ColdFusion, Flash, Flash Access, Flash Media Server, Flash Media Interactive Server, and Flash Player 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.

© 2010 Adobe Systems Incorporated. All rights reserved. 06/10