



Akamai® Streaming

**(for On Demand Adobe® Flash® Video)
User Guide**

Flash Media Server 3.5 (Akamai Version 7)

Akamai Confidential
For Customer Use Under NDA Only

September 30, 2009

Akamai Technologies, Inc.

Akamai Customer Care: **1-877-425-2832** or, for routine requests, e-mail **ccare@akamai.com**

The EdgeControl® portal, for customers and resellers: **<http://control.akamai.com>**

US Headquarters
8 Cambridge Center
Cambridge, MA 02142

Tel: 617.444.3000
Fax: 617.444.3001

US Toll free 877.4AKAMAI (877.425.2624)

For a list of offices around the world, see:
<http://www.akamai.com/en/html/about/locations.html>

Akamai® Streaming (for On Demand Adobe® Flash® Video) User Guide

Copyright © 2005–2009 Akamai Technologies, Inc. All Rights Reserved.

Reproduction in whole or in part in any form or medium without express written permission is prohibited. Akamai, the Akamai wave logo, and the names of Akamai services referenced herein are trademarks of Akamai Technologies, Inc. Other trademarks contained herein are the property of their respective owners and are not used to imply endorsement of Akamai or its services. While every precaution has been taken in the preparation of this document, Akamai Technologies, Inc. assumes no responsibility for errors, omissions, or for damages resulting from the use of the information herein. The information in these documents is believed to be accurate as of the date of this publication but is subject to change without notice. The information in this document is subject to the confidentiality provisions of the Terms & Conditions governing your use of Akamai services.

Adobe, ActionScript, Macromedia, Flash, and AIR are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

All other product and service names mentioned herein are the trademarks of their respective owners.

Contents

PREFACE • 1

About This Document	1
---------------------------	---

CHAPTER 1. INTRODUCING AKAMAI STREAMING (FOR ON DEMAND ADOBE FLASH VIDEO) • 5

About Adobe Flash Streaming Video	5
How Akamai Streaming Works	5
Origin Server Requirements	6

CHAPTER 2. GUIDELINES AND BEST PRACTICES • 7

CHAPTER 3. PROVISIONING AKAMAI STREAMING (FOR ON DEMAND ADOBE FLASH VIDEO) • 13

Accessing Your Akamai Streaming Account	13
About the On-Demand Configurations Page	15
Creating New Flash Configurations	17
Configuring Secure Streaming	21
Testing Flash Configurations	25
Editing Flash Configurations	28
Modifying Secure Streaming	31

CHAPTER 4. INTEGRATING AKAMAI STREAMING (FOR ON DEMAND ADOBE FLASH VIDEO) • 35

Akamai Streaming (for On Demand Adobe Flash Video) Communications	35
Specifying Codecs with Akamai Streaming	36
Using Dynamic Streaming	37
Encoding Your Videos for Dynamic Streaming	38
Creating a Flash Playback Application for Dynamic Streaming	38
Creating a Flash Playback Application	38
Building a Playback Application Using the AkamaiConnection Class	39
Building a Playback Application Without the AkamaiConnection Class	39
A Note on the NetStream.onStatus Event Handler and “Stream Not Found” Behavior	42
Streaming MP3s	42
Expediting the Connection	43
Dealing with Problematic Client-Side Proxy Servers	44
Solving the Problem Using the AkamaiConnection Class	45
Solving the Problem Without the AkamaiConnection Class	45
Using the “ondemand” Server-Side Application	47
Automatic Bandwidth Detection	47
Stream Length Reporting	49
Using Bandwidth Detection and Stream Length Reporting Together	50
Using Adobe Flash’s Media Components	51
Using Flash Media Components with H.264- and HE-AAC-Encoded Media	51
Using the AkamaiNCManager ActionScript Class	51
Usage Notes	52

CHAPTER 5. USING SECURE STREAMING • 53

Secure Streaming Guidelines for On Demand Flash Video	53
Using an slist	54
Using Secure Streaming with Dynamic Streaming	55

Integrating Secure Streaming for On Demand Flash Video	56
Passing the Token in the connect() Method	56
Passing the Token in the Referer Header	57
Inserting the Token into HTML	57
Inserting the Token into JavaScript	58
CHAPTER 6. CHANGING FLASH VIDEO CONTENT • 61	
Purging the Video from the Akamai Streaming Cache	61
Constructing Purge URLs	61
Purging the Akamai Streaming Cache	62
APPENDIX A. USING THE FLASH SUPPORT PLAYER • 65	
Troubleshooting with the Flash Video Test Player	67
Using Trace Data.	67
Overriding the Server IP	68
APPENDIX B. USING THE PROGRESSIVE DOWNLOAD ALTERNATIVE • 69	



Preface

Akamai® Streaming (for On Demand Adobe® Flash® Video) combines the technologies of both Akamai and Adobe Systems Incorporated to offer customers a streaming Flash video solution that rounds out Akamai's existing streaming media suite.

About This Document

This guide provides an overview of Akamai Streaming (for On Demand Adobe Flash Video), as well as details regarding its setup and use. It is intended for customers who will be provisioning the service for use with their Web properties. It also provides information regarding the actual integration of the service into client-side Flash applications and will be of use to developers of those applications.

This document is organized into chapters as follows:

Chapter 1. Introducing Akamai Streaming (for On Demand Adobe Flash Video) gives an overview of Akamai Streaming.

Chapter 2. Guidelines and Best Practices provides tips, guidance, and suggestions regarding the Akamai Streaming (for On Demand Adobe Flash Video) service.

Chapter 3. Provisioning Akamai Streaming (for On Demand Adobe Flash Video) provides procedures for using the EdgeControl® portal to prepare your Akamai Streaming account for use.

Chapter 4. Integrating Akamai Streaming (for On Demand Adobe Flash Video) discusses options and methods for setting up your client-side Flash applications to receive streaming video from Akamai Streaming.

Chapter 5. Using Secure Streaming outlines the Secure Streaming feature as it applies to Flash video, providing you additional control over which end users access your content.

Chapter 6. Changing Flash Video Content walks you through procedures for purging Flash video content from Akamai Streaming so that replacement content may be streamed in its place.

Other Resources

Additional information regarding the following Akamai products can be accessed through the EdgeControl portal (<https://control.akamai.com>).

Akamai Streaming (for Adobe Flash Video) Tools ([On Demand Streams >> Documentation](#) (*or* [Documentation >> On Demand Streams](#)))

- AkamaiConnection ActionScript class files
- AkamaiNCManager ActionScript class file

Akamai Secure Streaming ([On Demand Streams >> Documentation](#) (*or* [Documentation >> On Demand Streams](#)))

- *Akamai Secure Streaming Integration Guide*

Akamai NetStorage ([NetStorage >> Documentation](#) (*or* [Documentation >> NetStorage](#)))

- *Akamai NetStorage User Guide*
- *Managing Akamai NetStorage Accounts*

Akamai Traffic Reports ([Documentation >> Traffic Management](#))

- “Reports Help”
- *Akamai Data Reference*

Documents and Articles Regarding Adobe Flash Software and Video

Adobe Flash CS3 and Flash 8

- Flash CS3 (<http://www.adobe.com/support/documentation/en/flash/>)
- Flash 8 (<http://www.adobe.com/support/documentation/en/flash/documentation.html>)

Dynamic Streaming

- *Dynamic Streaming in Flash Media Server 3.5—Part 1: Overview of the New Capabilities* (http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_advanced_pt1.html)
- *Dynamic Streaming in Flash Media Server 3.5—Part 2: ActionScript 3.0 Dynamic Stream API* (http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_advanced_pt2.html)—This document is for those who plan to build their own client dynamic code.
- *Dynamic Streaming in Flash Media Server 3.5—Part 3: Integrating Dynamic Streaming with Existing Video Players* (http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_advanced_pt3.html)—This document is for those who will incorporate their own code into an existing video player.

- *Dynamic Streaming On Demand with Flash Media Server 3.5* (http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_on_demand.html)—This document includes best encoding practices.

Supported Media Formats

- *List of Codecs Supported by Adobe Flash Player* (<http://kb.adobe.com/selfservice/viewContent.do?externalId=kb402866&sliceId=2>)
- *Flash Video Primer* (http://www.adobe.com/devnet/flash/articles/flash_flv.pdf)
- *Encoding Best Practices for Prerecorded Flash Video* (http://www.adobe.com/devnet/flash/articles/flv_encoding.html)
- *Flash Video Learning Guide* (http://www.adobe.com/devnet/flash/articles/video_guide.html)
- *Importing Video: Files and Encoding Guidelines* (http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_18990&sliceId=2)
- *Exploring Flash Player Support for High-Definition H.264 Video and AAC Audio* (http://www.adobe.com/devnet/flashplayer/articles/hd_video_flash_player.html)
- *Using the FLVPlayback Component with Flash Player 9 Update 3* (http://www.adobe.com/devnet/flash/articles/flvplayback_fplayer9u3.html)
- *MIME Type Registration for MPEG-4* (<http://www.rfc-archive.org/getrfc.php?rfc=4337>)
- *New File Extensions and MIME Types* (<http://www.kaourantin.net/2007/10/new-file-extensions-and-mime-types.html>)

Streaming Versus Progressive Download

- *Delivering Flash Video: Understanding the Difference Between Progressive Download and Streaming Video* (http://www.adobe.com/devnet/flash/articles/flv_download.html)

Chapter 1. Introducing Akamai Streaming (for On Demand Adobe Flash Video)

In This Chapter



About Adobe Flash Streaming Video • 5

How Akamai Streaming Works • 5

Akamai Streaming (for On Demand Adobe Flash Video) leverages Akamai's network of thousands of servers to deliver on-demand Flash video streams. The service transfers Flash videos from an origin server to the distributed Akamai Streaming service to enable delivery of streams to end users from the edge of the Internet.

About Adobe Flash Streaming Video

Flash video provides a high-quality video format for Web delivery. Streaming Flash video offers a number of advantages, including fast video playback and added security. These video files can be identified by their **.flv** file extension. In addition, Akamai Streaming (for Adobe Flash Video) supports MP3 audio, which uses the **.mp3** file extension, as well as the MPEG-4 standards H.264 video and HE-AAC audio. These latter two codecs have several possible file extensions, including **.mp4**, **.m4v**, **.m4a**, **.mov**, **.3gp**, **.f4v**, **.f4p**, **.f4a**, and **.f4b**.



*Note: Unless specified otherwise, the terms **Flash video** or **video** are used to collectively refer to all supported video and audio formats.*

In addition to the Flash video itself, the following are required to stream Flash video:

- **Client-Side Flash Application (a.k.a. Flash Movie or Flash Video Player).** This file is identified by its **.swf** file extension and is created by your Flash application developer. It resides on the Web server and is called by HTML in your Web page. This starts Adobe Flash Player, which reacts to script written into the application, establishes a connection with Akamai Streaming, and requests the video file.
- **Adobe Flash Player (Version 6 or Higher (Version 9.0.115.0 or Higher for H.264 Video streaming)).** This is a rich client that plays Flash applications and communicates with Akamai Streaming to enable Flash video streaming. It is downloaded by the end user and installed on his or her computer.

How Akamai Streaming Works

When an end user's Flash Player requests a Flash video, the request, using Akamai's intelligent routing technology, resolves to an Akamai Streaming server located optimally for that end user. If the Akamai Streaming server does not already have the

video in its cache, it requests it from your origin server; this download is performed using HTTP byte-range gets.

As soon as the first block of data is retrieved, the Akamai Streaming server begins streaming the video to the user. Once downloaded from the origin, the video remains cached on the Akamai Streaming server to handle further requests, which distributes the load on your server around the world. As a result, Akamai Streaming, with its global distributed network, removes a significant burden from your infrastructure.

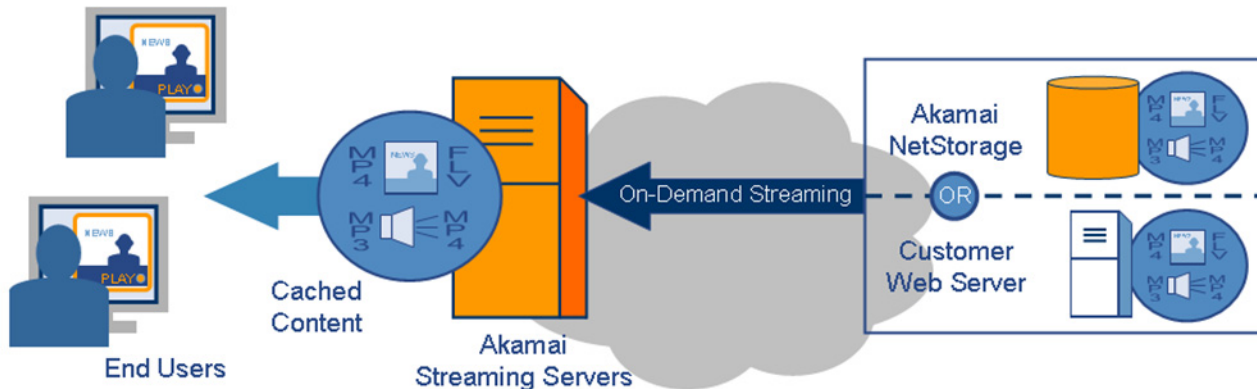


Figure 1-1. Akamai Streaming (for On Demand Adobe Flash Video)

Origin Server Requirements

If you host your Flash videos on your own server, you must have adequate resources provisioned so that Akamai's servers can retrieve the videos. Following are requirements for origin servers:

- Servers must be HTTP 1.1 compliant to support byte-range GET requests.
- Servers must return a Last-Modified header on every byte-range GET request; the header must be identical for all chunks of a given file and must not change unless the file is replaced.


► *Note: If your origin consists of multiple servers behind a load balancer, or if the origin host name returns multiple server IP addresses, you must ensure that, for a given file, the Last-Modified header is identical on each server and IP address.*

- Servers must return an updated **Last-Modified** header when a file is replaced. (You must also purge the file from Akamai Streaming Edge servers using the EdgeControl portal's Content Control Utility to ensure they fetch the new version (see Chapter 6).)
- Servers must have adequate processing power and provisioned bandwidth to handle the maximum expected simultaneous requests from Akamai's servers (note that this will never exceed, and is usually substantially less than, the maximum number of requests received when not using Akamai Streaming).

Of course, you may be able to eliminate the need to maintain your own origin server altogether by using Akamai's NetStorage service. Contact your Akamai representative for more information regarding this service.

Chapter 2. Guidelines and Best Practices

There are several important considerations you should take into account when using Akamai Streaming (for On Demand Adobe Flash Video).

 *Note: The following is not intended to be a complete list of all precautions you should take or considerations you should evaluate in using the service. If you have questions, please consult with an Akamai Customer Care representative.*

Use Akamai's AkamaiConnection class

The AkamaiConnection class is available as both an ActionScript 2.0 and ActionScript 3.0 class that you can download from the EdgeControl portal. It is designed to assist Flash application developers in establishing a robust connection with the Akamai Streaming (for Adobe Flash Video) service using current best practices for connection expedition, client-proxy penetration, and buffer management. Its use is strongly recommended and is discussed in Chapter 4.

If not using the AkamaiConnection class, include ActionScript to automate the connection expedition and client-side proxy solutions

If you decide not to use Akamai's AkamaiConnection class to build your client-side Flash application, it is strongly recommended you include ActionScript in your application to expedite Flash Player's connection to Akamai Streaming and to surmount any problematic client-side proxies. Chapter 4 discusses both of these solutions.

HTTP headers for ident requests must not be larger than four (4) kilobytes in size


Your client Flash applications can direct Flash Player to send ident requests to Akamai Streaming as a means of circumventing problematic proxy servers (see "Dealing with Problematic Client-Side Proxy Servers" on page 44). Some firewalls and proxy servers, however, can introduce large cookies into these requests' HTTP headers, adding considerable bulk. To allow for these, Akamai Streaming supports HTTP headers up to four (4) kilobytes in size. For efficiency's sake, headers larger than this are denied.

Akamai Streaming only supports streaming media

Flash Media Server can enable many different types of applications. Akamai Streaming, however, only supports streaming video and audio at this time.

Akamai Streaming (for On Demand Adobe Flash Video) supports several video and audio streaming media formats

Akamai Streaming supports Flash video (.flv) and MP3 (.mp3) audio, as well as MPEG-4 standard H.264 video and HE-AAC (High Efficiency Advanced Audio Coding), the file extensions of which include .mp4, .m4a, m4v, .mov, .3gp, .f4v, .f4p, .f4a, and .f4b.

 *Note: Support of Flash video (.flv) encoded with the On2 VP6 video codec was introduced with Adobe Flash Player 8, and support of the H.264 video and HE-AAC audio codecs was introduced with Adobe Flash Player 9 Update 3 (version 9.0.115.0).*

Akamai Streaming supports on-demand Flash video (.flv) encoded with the On2 VP6 codec

Support of the On2 VP6 video codec was introduced with Adobe Flash Player 8. Akamai Streaming supports on-demand Flash video encoded in this way, as well.

Be aware of potential high-bitrate issues for end users when encoding video with the H.264 video codec

The H.264 video codec is designed for high-quality, high-bitrate video, and many encoders, by default, create videos with bitrates that may be difficult for end users' bandwidth capacities to handle. You should, therefore, use care when encoding your videos to ensure you are not overpowering your audience's capabilities.

Akamai Streaming supports the enhanced seek feature

Normally, when an end user seeks within a streaming Flash video, Flash Player begins playback at the keyframe or seek point closest to the point the end user selected. The accuracy, therefore, is dependent on how frequently keyframes and seek points have been inserted in the video. Akamai Streaming, however, takes advantage of Flash Media Server's enhanced seek feature, which dynamically uncompresses the video, inserts a new keyframe or seek point at the user-selected point, recompresses the video, and resumes playback from that point.

Older Flash video encoders do not support metadata

Akamai recommends using one of the latest FLV encoders to create your Flash videos, as older versions may not support the inclusion of metadata within the video. Depending on how you use the exporter, metadata can include information such as the video's duration, width, height, video data rate, audio data rate, frame rate, and creation date. Of these, the one that may have the most impact is video duration, without which Flash application developers cannot include seek and progress bars in their video players.

Akamai Streaming automatically closes idle connections

By default, if an end-user connection remains idle (i.e., no data has streamed) for 20 minutes, Akamai Streaming automatically closes the connection. Instances in which this might occur include end users pausing playback on their client applications, or the stream terminating for some reason (e.g., end of stream).

If the default 20-minute threshold is impractical for your situation (e.g., you are serving advertisement streams), your Akamai representative can adjust it to your specification.

Akamai Streaming supports automatic bandwidth detection and stream length reporting

A server-side application provides for automatic bandwidth detection, allowing you to upload multiple bandwidth-optimized versions of a video and then stream the one most suitable to a user's Internet connection speed. The application can also report the video's duration (obtained from its metadata) to enable application features like progress bars. Both of these features must be requested by ActionScript included in your client-side application. Chapter 4 describes how to use these features.

Akamai Streaming only supports Akamai-provided server-side scripts and applications

Akamai does not support customer-authored scripts and applications for Flash Media Server.

Akamai Streaming does not support custom server configurations

This includes such things as log formats and server-side protocol/port rollover instructions.

If you are streaming media encoded with the H.264 or HE-AAC codecs, force end users to upgrade to a supported version of Adobe Flash Player

It is important to avoid streaming media encoded with either the H.264 video or HE-AAC audio codecs to end users having Adobe Flash Player versions older than 9.0.115.0. Due to a bug in the current version of Flash Media Server used on the Akamai Streaming platform, attempts to do so will cause the server to crash, thereby causing a denial of service.

As a precaution, if you are using these codecs you should force end users with older Flash Players to upgrade to the current version. There are methods available for detecting an end user's Flash Player version and producing a particular behavior based on the result, including forcing end users to upgrade their Flash Player or redirecting them to the Adobe Flash Player download Web page if no player is detected at all. These methods are discussed at http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_14526&sliceId=2. The Flash Player Detection Kit is available for assistance in setting this up at http://www.adobe.com/products/flashplayer/download/detection_kit/.

Lastly, a useful online tool is available that tells users which version of Flash Player is currently installed on their computers. (<http://www.adobe.com/products/flash/about/>).

When using SWF Verification, upload your SWF file to your SWF Path on Akamai NetStorage before making it available to end users to avoid denials of service

If your SWF file is available to your end users and they use it to request your video before you have uploaded the SWF to your SWF Path on NetStorage, the SWF Verification will fail, resulting in denials of service lasting 15 minutes (the Akamai Streaming server cache TTL).

When replacing SWF Verification files on Akamai NetStorage, temporarily maintain both the old and new SWFs to prevent denials of service to end users

If you are using the SWF Verification feature, use care when replacing existing SWF files with updated versions, as a misstep could result in denials of service to your end users. Specifically, it is important to temporarily keep both versions available on NetStorage for SWF Verification until such time as you are reasonably certain all end users are using your new SWF. The following scenario illustrates why this is important:

1. A customer uploads a SWF file, **player.swf**, to their SWF Path on NetStorage, followed by an upload of the same file to their HTTP origin Web site.

2. An end user requests **player.swf** from your Web origin, and it is subsequently delivered to his or her Web browser.
3. The Flash Player client creates a hash of the SWF file that is sent with the video request to an Akamai Streaming server.
4. The Streaming server looks in its cache to determine if it has a matching SWF hash. In this case it does not, so it requests a list of hashes kept in the appropriate SWF Path from NetStorage.
5. A matching hash is found and is cached at the Streaming server for 15 minutes.
6. With a valid hash in its cache, the Streaming server begins streaming the video to the end user.
7. After some time, the customer updates the SWF file, saving it to their NetStorage SWF Path with the same name, thus overwriting the original.
8. Before the updated SWF is uploaded to the HTTP origin Web site, an end user again requests **player.swf**, which is delivered from the origin. This is, however, the original SWF, not the updated version that is now on NetStorage. At the same time, the existing SWF hash in the Streaming server's cache expires.
9. Once again, the SWF sends a stream request (and the Flash Player-created hash) to the Streaming server.
10. Because the previous SWF hash has expired from the server's cache, the server again requests a list of hashes from NetStorage. This time, however, there is no match.
11. With no matching hash, the Streaming server caches a "no-match" for 15 minutes, prohibiting the video from streaming to any end user using the original SWF and resulting in a denial of service.

To avoid this situation, it is best to upload your updated SWF file to NetStorage using a name different from the original (the files on the SWF Path and on your Web site origin do not need to have the same name, only the same hash) to avoid overwriting and, thus, leaving the original available to any end users who may request your video before you are able to upload the updated SWF to your origin. You do not need to do the same with your origin's file; you can overwrite the original SWF file there with the same name. In fact, this is recommended so you do not need to update your Web page, as well.

Once you feel reasonably sure that enough time has passed to where your end users are no longer using the original SWF, you can delete it from your SWF path.

When using SWF Verification, keep the number and size of SWF files in your SWF Path location to a minimum to ensure optimal performance

Generally speaking, the more content you have in your SWF Path location, the longer it takes to perform the SWF Verification process. This may produce a delay in starting media playback for some users.

Following are SWF Path recommendations to help with this:

- Do not include any files not specific to SWF Verification.
- Remove old SWF files when they are no longer needed.
- Use the fewest number of SWF files you can and keep their sizes small.

Do not upload Adobe AIR™ (Adobe Integrated Runtime) files to the SWF Verification path

Adobe AIR files are essentially archive files (similar to .zip files) containing Flash application (SWF) files, which are extracted at their runtime. If you use the SWF Verification feature with Akamai Streaming, the server will attempt to authenticate the requesting SWF by comparing its hash with the hash of your authorized archived version in your SWF Path on Akamai NetStorage.

You should, therefore, not upload your AIR file to your NetStorage SWF Path. Rather, you should extract and upload the SWF files contained within its contents, which you can do by renaming the AIR file with the .zip file extension and opening it with the unzipping application of your choice.

Akamai Streaming does not support SSL streaming

If you are serving your Web content and SWFs using Secure Sockets Layer (SSL), be aware that any videos streamed into them from Akamai Streaming will be unencrypted. You can, however, stream your videos using Adobe's enhanced RTMP protocol (RTMPE and RTMPTE), which uses 128-bit encryption.

Use of the RTMPE/RTMPTE protocol is enforceable on a per-configuration basis

If you use the RTMPE and RTMPTE protocols exclusively to stream videos associated with a particular Flash configuration, it is recommended you ask your Akamai representative to enable RTMPE enforcement for that configuration to prevent unauthorized access of your video via regular RTMP. Since enforcement is applied on a per-configuration basis, there are situations in which you may not wish to do this. For example, if your Flash configuration has multiple RTMP streams that you are in the process of migrating to RTMPE, you should disable enforcement until the migration is complete else end users attempting to access unmigrated streams via RTMP will be denied.

Also, be aware that Flash player versions previous to 9.0.115 do not support RTMPE, so when creating your client Flash applications you may need to consider either forcing end users with older versions to upgrade or support them through the RTMP protocol.

Chapter 3. Provisioning Akamai Streaming (for On Demand Adobe Flash Video)

In This Chapter

- Accessing Your Akamai Streaming Account • 13
- Creating New Flash Configurations • 17
- Testing Flash Configurations • 25
- Editing Flash Configurations • 28

Provisioning Akamai Streaming begins with the initial activation of your account by Akamai. When completed, you will access your Akamai Streaming account through the EdgeControl portal and assign all parameters necessary to serve your Flash video stream. You will then be ready to configure your client-side Flash applications to receive video streams as discussed in Chapter 4.

Accessing Your Akamai Streaming Account

Access your Akamai Streaming account as follows:

1. Log in to the EdgeControl portal.
 - a. Launch your Web browser and open <https://control.akamai.com>.

The EdgeControl login page appears.

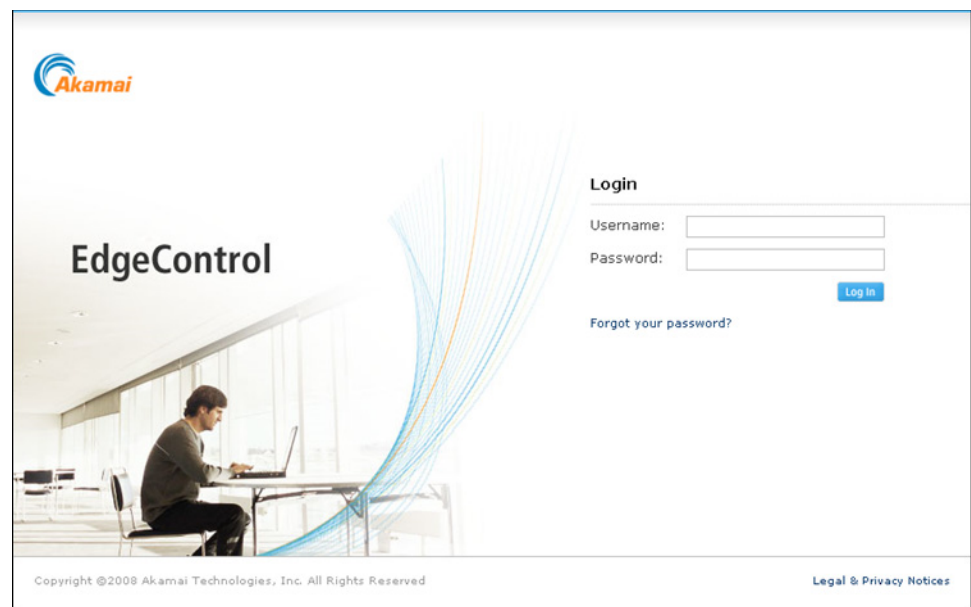


Figure 3-1. EdgeControl Login Page

- b. Enter your username and password, and click [Log In](#).

The EdgeControl **Welcome** page appears.

2. Navigate to the **Flash Configuration** page.

- a. In the left-hand navigation menu, click [On Demand Streams](#).

The **On Demand Streaming Traffic** page appears.

- b. In the left-hand navigation menu under [On Demand Streams](#), click [Configure Flash](#).

The **On-demand Configurations** page appears.

The screenshot displays the Akamai EdgeControl interface. The top header includes the Akamai logo, 'EdgeControl', and links for 'Support' and 'Logout'. The left-hand navigation menu is expanded, showing 'On Demand Streams' with sub-items like 'Traffic', 'Visitors', 'URLs', 'Subdirectories', 'Content Control Utility', 'Generate On Demand Link', 'Configure Flash', 'Tools', 'Log Delivery', 'Alerts', 'Recurring Reports', and 'Documentation'. The main content area is titled 'Flash Streaming' and 'On-demand Configurations'. It features a table with the following data:

CP Code	Stream Name	Options
9391	Product Demonstrations	View Details Edit Delete Test

Below the table, there is a link 'Add configuration: On-demand'. The footer of the page contains links for 'Contact Us', 'Feedback', 'Legal & Privacy', and a copyright notice: 'Copyright ©2008 Akamai Technologies, Inc. All Rights Reserved'.

Figure 3-2. On-Demand Configurations Page

About the On-Demand Configurations Page

There are several functions available on this page:

- **Create new configurations.** Click the [On-demand](#) link to display the **Add Flash Configuration** page (see “Creating New Flash Configurations” on page 17).
- **Show and hide configuration details.** Each configuration on the page is listed by its billing **CP Code** and **Stream Name**. You may expand any of the configurations to display additional information (see Figure 3-3 below).

To display configuration details, click the [View Details](#) link for an individual configuration. To hide configuration details, click a configuration’s [Hide Details](#) link.

- **Edit configurations.** To make changes to a Flash configuration, click its [Edit](#) link to display the **Edit Flash Configuration** page.
- **Delete configurations.** To delete a configuration, click its [Delete](#) link.
- **Test configurations.** To test a configuration’s video stream, click its [Test](#) link.

Figure 3-3 is an excerpt from the **Flash Configurations** page showing a configuration’s details.

CP Code	Stream Name	Options
9381	Product Demonstrations	Hide Details Edit Delete Test
<div> <div>Hostnames: cp9381.edgefcs.net (default) products.example.com</div> <div>Application Name Aliases: ondemand (default) stream vod</div> <div>Per Connect Secure Streaming: Enabled Configure Profiles</div> <div>SWF Authentication Enabled: Enabled</div> <div>SWF Path: example.download.akamai.com/9389/swfauth/productdemos/</div> <div>Virtual Path Name: /contraptions</div> <div>Actual Path: example.download.akamai.com/9389/contraptions</div> <div>Virtual Path Name: /gizmos</div> <div>Actual Path: www.example.com/gizmos</div> </div>		

Figure 3-3. Flash Configuration Details

The details display consists of the following elements:

- **CP Code.** The Content Provider code used to report and bill your Flash streams.
- **Stream Name.** The unique name you give to the configuration.

- **Hostnames.** The default hostname Akamai Streaming assigns when you initially create your Flash configuration, as well as any hostname aliases you create for use in lieu of the default.
- **Application Name Aliases.** The list of the Flash configuration's application names, including the default and any aliases you create.
- **Per Connect Secure Streaming.** The status of Akamai's Secure Streaming feature, which allows you control over who is able to view your Flash video streams associated with that particular Flash configuration. When enabled, a [Configure Profiles](#) link appears alongside, which opens the **Per Connect Secure Streaming Profiles** page, allowing you to configure the feature.
- **SWF Verification Enabled.** The status of the SWF Verification feature, which helps ensure only authorized client-side Flash applications (SWFs) can stream your video. If enabled, when your SWF file attempts to access a video associated with the Flash configuration, Akamai Streaming compares a hash of the SWF file with the hash of your authorized archived version, and if the two match it commences the stream. If the two do not match, a **NetConnection.Connect.Close** event is sent.
- **SWF Path.** This is only present if SWF Verification is enabled. It is the Akamai NetStorage location to which you will upload your authorized SWF files. Akamai Streaming uses these to verify the validity of the SWF files requesting videos associated with the Flash configuration.
- **Virtual Path Name.** The alias you create to represent the actual path to your Flash videos' origin server. The virtual path acts as a shortcut your Flash application developers use in lieu of the actual path.
- **Actual Path.** The full path to your Flash videos' origin server.

Creating New Flash Configurations

You may create as many Flash configurations as you have CP codes available, depending on how you would like to report on and bill your Flash video streams.

1. Display the **Add Flash Configuration** page.
 - a. Click the [On-demand](#) link.

The page appears.

Add Flash Configuration

[On-demand Configurations](#) | **Add Configuration**

To add a new Flash configuration, complete the form below and click **Save**.

Configuration Name:

CP Code:

Default Hostname: Please select a CP Code.

Host Aliases (optional):

Separate multiple aliases with a new line. To stream content on these hostnames, you must create them on your DNS server as CNAME records pointing to the default hostname above.

Per Connect Secure Streaming: ☐ Enable Per Connect Secure Streaming

Default Application Name:

Application Name Aliases (optional): (Alias 1)

(Alias 2)

(Alias 3)

You may specify up to three total. If not specified, the default application name is "ondemand". Only the following characters are allowed - "a-z" "A-Z" "0-9".

SWF Verification: ☐ Enable SWF Verification

Virtual Path Names and Locations

Path 1

Virtual Path Name: /

Origin Location:

☐
NetStorage
 Example: storage.download.akamai.com/1234/videos

☐
Customer's Hosted Origin Server with HTTP access to content
 Example: www.example.com/flash/demos

[Create another virtual path](#)


Figure 3-4. Add Flash Configuration Page

2. Enter a configuration name.
 - a. In the **Configuration Name** text box, enter a unique identifier.
3. Select the billing CP code.
 - a. From the **Billing CP Code** dropdown menu, select the CP code you would like to use for reporting and billing of the video stream or streams associated with this configuration.


Based on your selection, the **Default Hostname** field populates with the path your Flash application developers will use in their client-side applications to access Akamai Streaming. This is an Akamai-generated hostname consisting of the CP code you selected plus “edgefcs.net” (e.g., cp9391.edgefcs.net)

4. Enter one or more host aliases for the server-side application path (**optional**).
 - a. In the **Host Aliases** text box, enter any domain names you plan to CNAME to the default hostname, entering each alias on a new line.

If you have unique, registered domain names you would like to resolve to the Akamai-generated default hostname generated in step 3 above, enter them here. Your Flash application developers may use the aliases in their applications in place of the default hostname.

 *Note: You must set these CNAMEs up yourself; Akamai does not provide this service. The information you enter here is required by Akamai so that the appropriate server-side configurations can be made. If you plan to use host aliases and you do not populate this field, your streams will not work.*

5. Enable Per Connect Secure Streaming, if desired.
 - a. If you wish to enable secure streaming for all the configuration’s videos, click the **Per Configuration Secure Streaming** check box.

 *Note: Secure Streaming helps prevent theft or deep linking of your Flash video links. It is an additional service you must include in your Akamai contract to use. Otherwise it is unavailable, and the check box is not present.*

If you use Secure Streaming, you will later need to configure at least one Secure Streaming profile (see “Configuring Secure Streaming” on page 21).


6. Create application name aliases, if desired.

This action enables you to create aliases of your choice to use in place of the default name, “ondemand” (e.g., **www.example.com/stream/** instead of **www.example.com/ondemand/**). More information regarding “ondemand” is available in Chapter 4.

- a. In the **Application Name Aliases (optional)** text boxes (**Alias 1**, **Alias 2**, and **Alias 3**), enter up to three aliases to use in addition to the “ondemand” default.

7. Enable SWF Verification, if desired.

To use the SWF Verification feature, you must have an Akamai NetStorage account set up, and you must create your SWF path structure on that account before designating it here. Once complete, you must upload to this location all Flash playback application (SWF) files you intend to use to stream this Flash configuration's videos. Akamai Streaming will use these files to create hashes with which it can verify the authenticity of any SWF files requesting your streams. You may also use this location as the origin for the SWF files on your Web site, but if you choose not to, you must still upload copies of them here.

 *Note: It takes up to 15 minutes for new or replacement file uploads to NetStorage to become available for SWF Verification on the Akamai Streaming network. During this period, attempts at SWF Verification with these files may fail. Likewise, file deletions from NetStorage take up to 15 minutes to take effect.*

- a. If you wish to enable SWF Verification for all the configuration's videos, click the **Enable SWF Verification** check box.

The **SWF Path** dropdown menu and text box appear.

- b. From the **SWF Path** dropdown menu, select the Akamai NetStorage domain housing your authorized client-side Flash application SWF files.
- c. In the SWF Path text box, enter any subdirectories (if any) along the NetStorage domain path in which your authorized SWF files are kept.

8. Create the Flash video virtual path or paths.

A virtual path defines mapping between Akamai Streaming and your content origin. Your Flash developers will use this in their client-side application to represent the Flash video's location.

- a. In the **Virtual Path Name** text box, enter a name (containing no whitespace) for your virtual path.

This entry is required, and your Flash application developers will use it to point their client-side applications to the Flash video content.

- b. In the **Origin Location** area:
 - If you are using **Akamai NetStorage as your content origin**, select the **NetStorage** radio button, from the dropdown menu select the appropriate account, and enter any remaining path elements for your Flash videos' location in the accompanying text box.
 - If you are using your own **HTTP-accessible origin server**, select the **Customer's Hosted Origin Server** radio button, and enter the full path of your Flash videos' location, including the domain, directory, and subdirectories in the accompanying text box.
- c. If you wish to include additional paths in this configuration, click the [Create another virtual path](#) link and enter the appropriate information.

Be aware, you do not need to create a virtual path for each and every directory/subdirectory combination on your origin. Rather, the virtual path should be configured with the lowest-level directory allowing access to all videos in your directory structure. For example, if you are using an Akamai NetStorage origin **example.download.akamai.com/9389** containing the following structure:

- /Land/
 - /Bicycle/
 - MountainBike.flv
 - RoadBike.flv
 - /Snow/
 - Ski.flv
 - Sled.flv
- /Sea/
 - Sailboat.flv
 - Surf.flv
 - Speedboat.flv

Here, you would set up a virtual path and populate the **Origin Location** text box with **example.download.akamai.com /9389**, allowing you to stream all the directories' contents using the same configuration simply by appending the appropriate subdirectory and video name to the virtual path in your Flash application (see "Creating a Flash Playback Application" on page 38). So if your virtual path is called **Products** and you want to stream **Sled.flv**, you would enter **Products/Land/Snow/Sled** in your playback application.

9. Click .

A message appears confirming the configuration's creation. There is a short latency period from the time you create a configuration until the default hostname becomes available for use. You may monitor its progress by viewing the configuration's details—a status message will appear next to the hostname.



Note: Propagation of configurations to Akamai Streaming can take up to 30 minutes.

CP Code	Stream Name	Options
9391	Product Demonstrations	Hide Details Edit Delete Test
<div> <div>Hostnames: products.example.com cp9391.edgefcs.net (default)</div> <div>Application Name Aliases: ondemand (default)</div> <div>Secure Streaming Enabled: Enabled Configure Profiles</div> <div>SWF Authentication Enabled: Enabled</div> <div>SWF Path: example.download.akamai.com/9389/swfauth/productdemos/</div> <div>Virtual Path Name: /contraptions</div> <div>Actual Path: example.download.akamai.com/9389/contraptions</div> <div>Virtual Path Name: /gizmos</div> <div>Actual Path: www.example.com/gizmos</div> </div>		

Figure 3-5. Details for a Newly-Created Flash Configuration

10. Repeat for any additional configurations you wish to create.

Configuring Secure Streaming

If you have enabled Per Connect Secure Streaming for a Flash configuration, you must also create at least one Secure Streaming profile. This section will take you through this process. Refer to Chapter 5 for details regarding the full implementation of this feature.

1. Access the Flash Configuration's **Per Connect Secure Streaming Profiles** page.
 - a. On the **Flash Configuration** page, click [View Details](#) for the desired Flash configuration.

The configuration expands to display its details.

- b. Click [Configure Profiles](#).

The Per Connect Secure Streaming Profiles page opens.

The screenshot shows a web interface titled "Flash Configurations" with a "Help" link. Below the title is a breadcrumb "All Configurations | Per Connect Secure Streaming Profiles". The main heading is "Per Connect Secure Streaming Profile for CP Code 173055", with a link "Add additional profile" to its right. A table with three columns is shown: "Profile Name", "Token Type", and "IP Check". Below the table is a red warning message: "Please make sure to commit changes by clicking the 'Commit Changes' button after adding/modifying or deleting secure streaming profile." Below this is a note: "Note: You must modify the AIFP (Authentication Information Finger Print) parameter when you update any secure streaming profiles. Please refer to the streaming documentation to learn more about this." At the bottom right is a "Commit Changes" button.

Profile Name	Token Type	IP Check
<p>Please make sure to commit changes by clicking the "Commit Changes" button after adding/modifying or deleting secure streaming profile.</p> <p>Note: You must modify the AIFP (Authentication Information Finger Print) parameter when you update any secure streaming profiles. Please refer to the streaming documentation to learn more about this.</p>		

Figure 3-6. The Per Connect Secure Streaming Profiles Page

2. Create a Secure Streaming profile.
 - a. Click [Add additional profile](#).

The **Add Per Connect Secure Streaming Profile** page appears (see Figure 3-7 on the next page).

Flash Configurations
Help

All Configurations | Per Connect Secure Streaming Profiles | Add Per Connect Secure Streaming Profile

Authentication Profile for CP Code 9391
Hide Advanced Options

Profile name:
Password:

Advanced Options

Additional advanced options to configure per username.

Use E-type token:

Enabling E-type token requires inclusion of a Rijndael key in your token.

IP required in all tokens?:

Path required in all tokens?:

The path value in the token ties the token to a particular stream or piece of content. By removing the path from the token you are reducing the security of the token. By selecting "No" in this field you acknowledge that you are rejecting Akamai's strong recommendation against treating this parameter as "optional" and that you are introducing vulnerabilities to the security of your service by choosing to make this parameter optional. You are hereby on notice that, if you nevertheless choose to lower the security restriction, you bear all of the associated risks if any vulnerability occurs and release Akamai from any responsibility for any losses of any sort arising from taking this action.

Payload required in all tokens?:

Setting this value to Yes will reject any token that does not include a payload field

CIDR Block Restriction:

Use this option if you would like to restrict viewing of this stream to end users in a certain IP range or CIDR block. Entries are as follows: IP range restriction: a.b.c.d-w.x.y.z
CIDR block restriction: a.b.c.d/e
Single IP restriction: a.b.c.d-a.b.c.d

WARNING: The authentication profile added here applies to all the streams provisioned in this session.

Add Cancel


Figure 3-7. Add Per Connect Secure Streaming Profile Page (with Advanced Options Displayed)

- b. In the **Profile name** text box, type a name for the profile.

The profile name must be 24 characters or less, and can be made up of any alphanumeric characters, as well as asterisks (*) and underscores (_).

- c. In the **Password** text box, type a password.

Passwords must be 32 characters or less, made up of any characters except whitespaces and tabs.

 *Note: Make note of your password and store it safely. You will not be able to retrieve the password from the EdgeControl portal once you add the profile.*

3. If desired, set up advanced options for the configuration.

a. Click [Show Advanced Options](#).

The page expands to display the advanced options.

b. From the **Use E-type token** dropdown menu, select whether you would like to use an E-type token for authentication.

This creates a binary you must download and use in your token generator.

c. From the **IP required in all tokens?** dropdown menu, select whether you would like to require the end user's IP address be included in the token.

d. From the **Path required in all tokens?** dropdown menu, select whether you would like to require the content path be included in the token.



CAUTION: Akamai enables the path requirement by default, which ties the token to a specific stream or piece of content. Be aware, if you disable it you reduce the token's security, and you acknowledge that you are rejecting Akamai's strong recommendation against doing so, that you are introducing vulnerabilities to the security of your Secure Streaming service, and that you bear all associated risks should any vulnerability occur. Accordingly, you release Akamai from any responsibility for any losses of any sort arising from taking this action.

e. From the **Payload required in all tokens?** dropdown menu, select whether you would like to require a payload field be included in the token.

f. If you wish to restrict content access to end users within a particular range of IP addresses, type the parameters in the **CIDR Block Restriction** text box as follows:

- IP range restriction: **a.b.c.d-w.x.y.z**
- CIDR block restriction: **a.b.c.d/e**
- Single IP restriction: **a.b.c.d-a.b.c.d**

4. Submit the profile.

a. Click .

The **Per Connect Secure Streaming Profiles** page reappears with the new profile added.

5. Download the E-type token binary.

a. On the **Per Connect Secure Streaming Profiles** page, click ([Download E Token](#)) for the desired profile and save it to your desired location.

This step can be performed at a later time, if desired.

6. Commit the profile modifications.


- a. Click **Commit Secure Streaming Profile**.

The profile is committed and ready for use.

Testing Flash Configurations


After creating a Flash configuration, you can test it to ensure it is working correctly. To do so, the video must be available on the configuration's actual path (e.g., **example.download.akamai.com/9389/contraptions**).

In addition, if you have Per Connect Secure Streaming enabled for the configuration, and you have created a Secure Streaming profile, you may test your token-generation parameters to ensure they will work correctly in the real-world tokens you generate for your end users.

 *Note: Secure Streaming parameters only appear on the **Flash Configuration Testing** page if you have enabled the feature for the configuration.*

1. Access the **Flash Configuration Testing** page.
 - a. On the **Flash Configurations** page, click **Test** for the desired configuration.
The page appears.
2. Test the configuration.
 - a. From the **Host Name/Application** dropdown menu, select the hostname and application name you would like to test.
Any hostname and application aliases you created for the configuration will also be available here.
 - b. From the **Virtual Path** dropdown menu, select the virtual path that represents the actual path of the video you would like to test.
The actual path appears in the **Actual Path** field.
 - c. In the **Subdirectories / Filename** text box, type the desired video name (the file extension is required), including any applicable subdirectories on the actual path.
 - d. From the **Encoding Type** dropdown menu, select **flv**, **mp3**, or **H.264**, as appropriate to the type of media you are testing.
 - e. Enter required information for the Per Connect Secure Streaming token, if applicable.
 - i. From the **Token Type** dropdown menu, select either **D Type Token** or **E Type Token**.
 - ii. From the **Profile** dropdown menu, select the Secure Streaming profile with which you would like to test.

- iii. In the **Password** text box, enter the password for the selected Secure Streaming profile.
- iv. In the **Path** text box, enter the path along the virtual path of the streaming content (e.g., **virtual_path/subdirectory/video**).
- v. In the **AIFP** text box, enter a fingerprint of your choosing.
The AIFP parameter must be changed each time the Per Connect Secure Streaming profile changes to ensure the most recent version is used.
- vi. In the **Window** text box, enter the amount of time (in seconds) for which the token is valid.
- f. Enter any optional token information you wish to include:
 - i. In the **Time** text box, enter the time at which the token was created.
 - ii. In the **Payload** text box, enter any custom information you wish to include for tracking purposes.
 - iii. In the **IP Check** text box, enter the IP address from which the streaming request must originate.

 *Note: Additional information on Secure Streaming parameters is available in the Secure Streaming Integration Guide.*

- g. Click .

A video player appears at page bottom, which, assuming all entered information is valid, begins video playback. It also provides information such as the Edge server IP address and test results of various port and protocol connection combinations (see Figure 3-8 on the next page).

Flash Configuration Testing [? Help](#)

All Flash Configurations | Test Configurations

Host Name/Application:

Virtual Path:

Subdirectories / Filename:

Example: file.flv or file.mp3 or trailers/abcfilms/file.flv

Encoding Type:

Actual Path: example.download.akamai.com/9389

AKAMAI - Flash Video Streaming Service - Test Player Client IP address: 192.168.0.1

rtmp://cp9391.edgefcs.net/ondemand/9389/akamai.mp4

☐ Override server IP

☐ Use FastStart (on-demand only) ☐ Set buffer (seconds)

☐ Live Stream Auth

STEP 1: Requesting the IP address of the optimum Akamai server ✔ **208.49.199.133**

STEP 2: Testing all available port and protocol combinations

☐ Stop after first success

	1935	443	80
rtmp	✔	✔	✔
rtmpt	✔	✔	✔
rtmpe	✔	✔	✔
rtmpte	✔	✔	✔

(Bandwidth measured at 14501 kbps)

STEP 3: Playing the stream using the first good connection Success - buffer is full and video is streaming over rtmp:1935

VIDEO METADATA CUEPOINTS TRACE

Akamai Traffic Statistics

Network Capacity 1.016 2.750

Time: 00:00:17 | 00:22:11 Fps: 31 Buffer: 6s Size: 318x180

Video | Audio data rate unknown/unknown Codec: H.264

Figure 3-8. Flash Configuration Testing Page with Video Playback

In addition, using the player's control, you can pause and restart video playback, and you can use the **VIDEO**, **METADATA**, **CUEPOINTS**, and **TRACE** tabs to display their respective data.

In addition, once the player appears you can make changes to the playback parameters:

- If you wish to change the playback URL, you can do so in the yellow text box.
- If you wish to test against a specific Akamai Streaming server rather than using Akamai's normal mapping feature, select the **Override server IP** check box and enter the server's IP address in the resulting yellow **Enter IP here** text box.
- If you wish to use the FastStart feature, select the **Use FastStart (on-demand only)** check box.

With this feature enabled, the playback buffer is set to 0.5 seconds to quickly begin playback. As soon as the buffer fills completely, it resets to 3 seconds. You can override the 3-second default with the **Set Buffer (seconds)** check box.

- If you wish to override the player's default 3-second video buffer, select the **Set Buffer (seconds)** check box, and use the resulting spin box to specify the number of seconds to buffer.
- If you wish the player's **STEP 2** to limit itself to the first successful connection, foregoing connection attempts on remaining port and protocol combinations, select the **Stop after first success** check box.

Editing Flash Configurations

As previously stated, you may edit your Flash configurations to create application aliases. You may also change any of the other configuration parameters you wish.

1. Open the **Edit Flash Configuration** page.
 - a. On the **Flash Configurations** page, click the [Edit](#) link for the Flash configuration you want to change.

The page appears (see Figure 3-9 on the next page).

Edit Flash Configuration

[On-demand Configurations](#) | [Edit Configuration](#)

Configuration Name:

Billing CP Code: 9391

Default Hostname: cp9391.edgefcs.net

Host Aliases (optional):

Separate multiple aliases with a new line. To stream content on these hostnames, you must create them on your DNS server as CNAME records pointing to the default hostname above.

Per Connect Secure Streaming: ☒ Enable Per Connect Secure Streaming

Default Application Name: ondemand

Application Name Aliases (optional): (Alias 1)

(Alias 2)

(Alias 3)

You may specify up to three total. If not specified, the default application name is "ondemand".

SWF Authentication: ☒ Enable SWF Authentication

SWF Path:

Please note that SWF Path is required for enabling SWF Authentication feature and it needs to be a NetStorage location.

Example : To use "storage.download.akamai.com/1234/swfauth" as SWF Path, choose "storage.download.akamai.com/1234/" in the drop down list and enter "swfauth" in the text box.

Note: Subdirectories entered here must be first created at your origin site.

Virtual Path Names and Locations

Path 1

Virtual Path Name:

Origin Location:

☒
NetStorage
 Example: storage.download.akamai.com/1234/videos

☐
Customer's Hosted Origin Server with HTTP access to content
 Example: www.example.com/flash/demos

Path 2

Virtual Path Name:

Origin Location:

☐
NetStorage
 Example: storage.download.akamai.com/1234/videos

☒
Customer's Hosted Origin Server with HTTP access to content
 Example: www.example.com/flash/demos


☐ Remove the Virtual Path

[Create another virtual path](#)


Figure 3-9. Edit Flash Configuration Page

2. Change the configuration name, if desired.
 - a. Type a new identifier in the **Configuration Name** text box.
3. Add or delete host aliases, if desired.
 - a. In the **Host Aliases (optional)** text box, type a new alias or aliases (entering each on a new line) or delete unwanted aliases.
4. Enable or disable Secure Streaming, as desired.
 - a. Click the **Enable Per Connect Secure Streaming** check box to select or deselect it.
5. Create application name aliases, if desired.
 - a. In the **Application Name Aliases (optional)** text boxes (**Alias 1**, **Alias 2**, and **Alias 3**), enter up to three aliases to use in addition to the “ondemand” default.
6. Enable or disable SWF Verification as desired.

To use the SWF Verification feature, you must have an Akamai NetStorage account set up, and you must create your SWF path structure on that account before designating it here. Once complete, you must upload to this location all Flash playback application (SWF) files you intend to use to stream this Flash configuration's videos. Akamai Streaming will use these files to create hashes with which it can verify the authenticity of any SWF files requesting your streams. You may also use this location as the origin for the SWF files on your Web site, but if you choose not to, you must still upload copies of them here.

 *Note: It takes up to 15 minutes for new or replacement file uploads to NetStorage to become available for SWF Verification on the Akamai Streaming network. During this period, attempts at SWF Verification with these files may fail. Likewise, file deletions from NetStorage take up to 15 minutes to take effect.*

- Enable SWF Verification for the configuration's videos.
 - i. Click the **Enable SWF Verification** check box.
The **SWF Path** dropdown menu and text box appear.
 - ii. From the **SWF Path** dropdown menu, select the Akamai NetStorage domain in which the client-side Flash applications into which your videos will stream are kept.
 - iii. In the SWF Path text box, enter any subdirectories (if any) along your NetStorage domain path in which your SWFs are kept.
- Disable SWF Verification for the configuration's videos.
 - i. Clear the **Enable SWF Verification** check box.

7. Add, delete, or edit your virtual paths, as desired.
 - Add a virtual path.
 - i. Click the [Create another virtual path](#) link.
A new, blank path appears.
 - ii. In the **Virtual Path Name** text box, enter a name (containing no whitespace) for your virtual path.
 - iii. In the **Origin Location** area:
 - If you are using Akamai NetStorage as your content origin, select the **NetStorage** radio button, from the dropdown menu select the appropriate account, and enter any remaining path elements for your Flash videos' location in the accompanying text box.
 - If you are using your own HTTP-accessible origin server, select the **Customer's Hosted Origin Server** radio button, and enter the full path of your Flash videos' location, including the domain, directory, and subdirectories in the accompanying text box.
 - Delete a virtual path.
 - i. Select the path's **Remove the Virtual Path** radio button.
 - Edit a virtual path.
 - i. In an existing path, make any necessary changes to the **Virtual Path Name** and/or **Origin Location** entries.
8. Click .

The changes you made are entered into the Flash configuration. A confirmation message appears with a link to view [All Flash Configurations](#).

Modifying Secure Streaming

You may also make changes to your Per Connect Secure Streaming profiles.

1. Access the Flash Configuration's **Per Connect Secure Streaming Profiles** page.
 - a. On the **Flash Configuration** page, click [View Details](#) for the desired Flash configuration.
The configuration expands to display its details.
 - b. Click [Configure Profiles](#).
The **Per Connect Secure Streaming Profiles** page opens.
2. Delete or modify your profile(s), as desired.
 - Delete a profile.
 - i. Click [Delete](#) next to the profile you wish to remove.
The profile is deleted.

- Modify a profile.
 - i. Click [Modify](#) next to the profile you wish to edit.

The **Modify Per Connect Secure Streaming Profile** page appears.

Flash Configurations ? Help

[All Configurations](#) | [Per Connect Secure Streaming Profiles](#) | **Modify Per Connect Secure Streaming Profile**

Profile name : Example

Use E-type token: Create a new E Token

Client IP Check: No

Path required in all tokens?: Yes

The path value in the token ties the token to a particular stream or piece of content. By removing the path from the token you are reducing the security of the token. By selecting "No" in this field you acknowledge that you are rejecting Akamai's strong recommendation against treating this parameter as "optional" and that you are introducing vulnerabilities to the security of your service by choosing to make this parameter optional. You are hereby on notice that, if you nevertheless choose to lower the security restriction, you bear all of the associated risks if any vulnerability occurs and release Akamai from any responsibility for any losses of any sort arising from taking this action.

Payload required in all tokens?: No

Setting this value to Yes will reject any token that does not include a payload field

CIDR Block Restriction:

Use this option if you would like to restrict viewing of this stream to end users in a certain IP range or CIDR block. Entries are as follows: IP range restriction: a.b.c.d-w.x.y.z
 CIDR block restriction: a.b.c.d/e
 Single IP restriction: a.b.c.d-a.b.c.d

Update Secure Streaming Profile Cancel

Figure 3-10. The Modify Per Connect Secure Streaming Profile Page

- ii. From the **Use E-type Token** dropdown menu, select an action.
 - **Create a new E Token.** Enables the E-type token and generates a new E-type token binary for use with your token generator (if the token was already enabled, a new binary is generated).

You must download this new token from the **Per Connect Secure Streaming Profiles** page by clicking on the profile's ([Download E Token](#)) link.

- **Keep the same token** (only present if E-type token was already enabled). Make no changes to the E-type token (remain enabled with the same token binary).
 - **Disable E Token.** Disable use of the E-type token.
- iii. From the **Client IP Check** dropdown menu, select whether you would like to require the end user's IP address be included in the token.
 - iv. From the **Path required in all tokens?** dropdown menu, select whether you would like to require the content path be included in the token.



***CAUTION:** Akamai enables the path requirement by default, which ties the token to a specific stream or piece of content. Be aware, if you disable it you reduce the token's security, and you acknowledge that you are rejecting Akamai's strong recommendation against doing so, that you are introducing vulnerabilities to the security of your Secure Streaming service, and that you bear all associated risks should any vulnerability occur. Accordingly, you release Akamai from any responsibility for any losses of any sort arising from taking this action.*

- v. From the **Payload required in all tokens?** dropdown menu, select whether you would like to require a payload field be included in the token.
 - vi. If you wish to restrict content access to end users within a particular range of IP addresses, type the parameters in the **CIDR Block Restriction** text box as follows:
 - IP range restriction: **a.b.c.d-w.x.y.z**
 - CIDR block restriction: **a.b.c.d/e**
 - Single IP restriction: **a.b.c.d-a.b.c.d**
3. Submit the profile changes.
 - a. Click **Update Secure Streaming Profile**.

The **Per Connect Secure Streaming Profiles** page reappears.
 4. Download the new E-type token binary, if applicable.
 - a. On the **Per Connect Secure Streaming Profiles** page, click ([Download E Token](#)) for the desired profile and save it to your desired location.

This step can be performed at a later time, if desired.
 5. Commit the profile modifications.
 - a. Click **Commit Secure Streaming Profile**.

The profile is committed and ready for use.

The next chapter provides information and examples your Flash application developers will find useful in integrating the service into their Flash applications.

Chapter 4. Integrating Akamai Streaming (for On Demand Adobe Flash Video)

In This Chapter



Akamai Streaming (for On Demand Adobe Flash Video) Communications • 35

Specifying Codecs with Akamai Streaming • 36

Creating a Flash Playback Application • 38

Expediting the Connection • 43

Dealing with Problematic Client-Side Proxy Servers • 44

Using the “ondemand” Server-Side Application • 47

Using Adobe Flash’s Media Components • 51

The final step in setting up Akamai Streaming is to create a Flash playback application to allow end users to view your streaming video. For this you have three options:

- You may build it with the **AkamaiConnection** class.
- You may build it without the **AkamaiConnection** class.
- you may use the native Adobe Flash media components.

Each of these is discussed in this chapter.



Note: The ActionScript examples provided in this chapter are written in ActionScript 2.0. If you are using ActionScript 3.0, refer to the appropriate Adobe documentation for information on its usage.

Akamai Streaming (for On Demand Adobe Flash Video) Communications

When playing a video stream, Adobe Flash Player initiates communications with Akamai Streaming on port 1935 using either RTMP (Real Time Messaging Protocol) or RTMPE (Encrypted Real Time Messaging Protocol). If traffic is in some way blocked (an interceding firewall blocks the port, for example), Flash Player will make successive attempts on different ports using different protocols.

- RTMP (or RTMPE) on port 1935
- RTMP (or RTMPE) on port 443
- RTMP (or RTMPE) on port 80
- RTMPT¹ (or RTMPTE) on port 80

1. RTMPT is tunnelled RTMP (RTMP packets wrapped in HTTP) also called HTTP tunneling.

Be aware, progression through this sequence can cause a significant delay in video playback, and Akamai recommends using your client-side Flash application to override it (see “Expediting the Connection” on page 43). Also, the RTMP and RTMPE protocols are mutually exclusive where the connection attempt sequence is concerned. For example, if you designate RTMP for your connection, RTMPE connections will not be attempted, and vice versa. The exception to this is if you are using the AkamaiConnection class, which allows you to customize your port and protocol sequence.

- ▶ *Note: If you have implemented RTMP- and RTMPT-specific firewall rules, you may need to update them to include the RTMPE and RTMPTE protocols.*
- ▶ *Note: RTMPT is only available with Flash Player version 6.0.65 and higher; RTMPE and RTMPTE require version 9.0.115.0 and higher (use the online tool at <http://www.adobe.com/products/flash/about/> to check your version).*

Specifying Codecs with Akamai Streaming

As you will see later in this chapter, videos are specified to Akamai Streaming with a URL of a particular form. For example:

`rtmp://cp9391.edgefcs.net/ondemand/movies/productdemo.flv`

This is broken down as follows:

- **Protocol:** rtmp
- **Hostname:** cp9391.edgefcs.net
- **Application name:** ondemand
- **Virtual path:** movies
- **File name:** productdemo

Depending on how you set up your Flash playback application, you will pass the URL in one of two ways. If you are using the native Adobe Flash media components, the URL is passed in its complete form; if you are using ActionScript, you will break it into two parts containing the hostname and application name in the `netConnection()` method (e.g., `rtmp://cp9391.edgefcs.net/ondemand`) and the virtual path and file name in the `play()` method (e.g., `movies/productdemo.flv`).

In either case, if you are streaming a file format other than Flash video (.flv), you must prepend the virtual path/file name portion with the appropriate prefix to ensure Akamai Streaming uses the correct codec. In all, there are three possibilities:

- **No prefix**—Akamai Streaming assumes the file uses one of the supported Flash video codecs.
 - Flash Components: `rtmp://cp9391.edgefcs.net/ondemand/movies/productdemo.flv`
 - ActionScript: `play("movies/productdemo");`

With no prefix, Akamai Streaming searches for the designated file name with the .flv extension (e.g., productdemo.flv) and applies the FLV codec. As is pointed out later in this chapter, when calling a Flash-coded video with the `play()` method, you must omit the file extension.

- **mp4:**—Akamai Streaming assumes the file is encoded with the MPEG-4 standard H.264 or HE-AAC codecs.
 - Flash Components: `rtmp://cp9391.edgefcs.net/ondemand/mp4:movies/productdemo.mov`
 - ActionScript: `play("mp4:movies/productdemo.mov");`

Including the **mp4:** prefix prompts Akamai Streaming to search for the designated file name with the specified extension (e.g., productdemo.mp4) and applies the MP4 codec. Be aware, if no file extension is present in the `play()` method, Akamai Streaming will first search for the file name with no extension, then for the file with the .mp4 extension, and then will stop. It is, therefore, important to always include the file extension when using this prefix.

- **mp3:**—Akamai Streaming assumes the file is encoded with the MP3 codec.
 - Flash Components: `rtmp://cp9391.edgefcs.net/ondemand/mp3:movies/productdemo`
 - ActionScript: `play("mp3:movies/productdemo");`

Including the **mp3:** prefix prompts Akamai Streaming to search for the designated file name with the .mp3 extension (e.g., productdemo.mp3) and applies the MP3 codec.

Using Dynamic Streaming

Dynamic Streaming that enables seamless, on-the-fly switching—based on the end user’s bandwidth (quality of service)—between multiple versions of a video encoded in different bitrates. This feature is only supported as of the release of Adobe Flash Player 10, so to use it you must either require your end users to upgrade their Flash Players or provide an alternate experience for those with older players. While taking no action will not interfere with the video stream, it may result in an undesirable experience, as the video will not make the aforementioned quality of service adjustments. For example, if an end user is initially playing a high-bitrate version of the video and his or her available bandwidth subsequently decreases, an inordinate amount of rebuffering might occur.

The application of Dynamic Streaming is based on interactions between your Flash playback application and the Akamai Streaming server. As the video plays, your application continuously monitors quality of service, including the actual bandwidth (based on incoming data and the buffer state) and the end user’s device rendering capability (by monitoring the number of dropped frames). As conditions change, the ActionScript in your playback application notifies the Akamai Streaming server when

it becomes necessary to shift the bitrate up or down. The server then, on the same connection, seamlessly changes to the appropriate bitrate video at the next keyframe. This feature is advantageous in that it delivers a better end-user experience by accommodating wide variations in end-user bandwidth and device capabilities, and by gracefully handle dynamically changing conditions (e.g., in a wi-fi environment in which bandwidth can change mid-stream).

Encoding Your Videos for Dynamic Streaming

Dynamic Streaming supports the ON2 VP6, H.264, HE-AAC, and MP3 codecs, allowing the full range of Flash-compatible video and audio options. To use the feature, encode multiple versions of your video using different bitrates and save them on your origin as separate files.

When a mid-stream switch occurs between different versions of a video, it takes place at keyframe locations. To avoid video and/or audio artifacts during a switch each version of your video should be encoded with a constant (versus variable) keyframe interval (five (5) seconds is recommended), as well as an identical timeframe, which will help to avoid jumps in the video during a switch. Also, audio for the videos should be encoded at the same bitrate and sample rate.

► *Note: For lower bitrate streams, encoding the audio in mono at the same bitrate (i.e., half the stereo bitrate) is acceptable.*

Creating a Flash Playback Application for Dynamic Streaming

You have two choices in creating your playback application for Dynamic Streaming. You can either build it using Adobe's new ActionScript API that enables Dynamic Streaming features, or you can use Akamai's utility classes (Open Video Player), available at <http://openvideoplayer.sourceforge.net>. The latter is recommended, as it facilitates implementation since Akamai has already done the work of integrating Adobe's API to work correctly with Akamai Streaming.

► *Note: Open Video Player is provided as a reference and a best practice guide only. Your Flash developers may modify it or develop their own Dynamic Streaming heuristics, as desired.*

Creating a Flash Playback Application

The Flash application you create for end users to view your Flash stream can be as simple or as complex as you like. You can build the most basic application with Adobe Flash software simply by inserting an embedded video instance onto a blank stage, naming the instance, and adding an ActionScript.

► *Note: While streamed Flash videos are not cached on the client side, SWF files are. If your browser does not play the latest version of your application, clear its cache and retry the application. Simply refreshing the page will not work.*


Building a Playback Application Using the AkamaiConnection Class

For added convenience, Akamai provides the **AkamaiConnection** class—available in both ActionScript 2.0 and ActionScript 3.0—to assist your Flash application developers in establishing a robust connection with the Akamai Streaming (for On Demand Adobe Flash Video) service while also implementing the best practices and methods outlined later in this chapter:

- **NetStream.onStatus Event Handling.** See “A Note on the NetStream.onStatus Event Handler and “Stream Not Found” Behavior” on page 42.
- **Connection Expedition.** See “Expediting the Connection” on page 43.
- **Client-Side Proxy Penetration.** See “Dealing with Problematic Client-Side Proxy Servers” on page 44.
- **Automatic Bandwidth Detection.** See “Using the “ondemand” Server-Side Application” on page 47.
- **Stream Length Reporting.** See “Using the “ondemand” Server-Side Application” on page 47.

In addition, the class also offers other features such as **dynamic buffer management** and **event reporting**.

The class always creates a NetConnection and can optionally create a NetStream on that NetConnection. Connections are initiated by a public method, and events are used to notify the parent class of connection success, connection failure, errors, and NetStream- and NetConnection-specific events.

 *Note: The AkamaiConnection class focuses on the connection layer exclusively and does not facilitate user-interface elements.*

You can obtain the class file from the EdgeControl portal’s documentation area ([On Demand Streams >> Documentation](#)); it is packaged with a specification and usage document, which provides complete details on its use and features. The package also contains a series of detailed, working sample files that implement all the methods, events, and properties described in the specification. These serve as good ActionScript references for using the class in real-world Flash applications.

To use the ActionScript 2.0 version of the AkamaiConnection class, simply copy it to the same directory as your FLA file. When you publish the FLA, ActionScript imports the class and compiles it into the resulting SWF file. If you want to store the class in a different location, you can still pull it into your SWF by specifying a class path in your ActionScript.

Building a Playback Application Without the AkamaiConnection Class

The following describes the construction of a simple playback application that does not use the AkamaiConnection class.

1. In Flash CS3 or Flash 8, open a new Flash document.

- a. From the **File** menu, select **New**.
The **New Document** dialog box appears.
 - b. Select **Flash Document** (Flash 8) or **Flash File (ActionScript 2.0)** (Flash CS3) and click **OK**.
A new Flash document appears.
2. Create a new video instance.
 - a. If the **Library** pane is not present, select **Library** from the **Window** menu.
The pane appears.
 - b. From the pulldown menu on the right-hand side of the **Library** pane's title bar select **New Video**.
The **Video Properties** dialog box appears.
 - c. Enter a name in the **Symbol** text box (if desired), select the **Video (ActionScript-controlled)** radio button, and click **OK**.
A new **Video** instance appears in the **Library** pane.
 - d. Drag the **Video** instance on to the stage.
The instance appears on the stage.
3. Name the video instance.
 - a. If it is not already expanded, click the **Properties** pane's title bar.
The pane expands.
 - b. In the **<Instance Name>** text box, type a name for the video instance.
4. Add the ActionScript.
 - a. In the **Timeline** pane, click the first frame of **Layer 1**.
 - b. If it is not already expanded, click the **Actions** pane's title bar.
The pane expands.
 - c. In the **Actions** pane, type your ActionScript (see below).
5. Test the Flash application.
 - a. From the **Control** menu, select **Test Movie**.
A new pop-up window appears and your video begins playback.

Following is a breakdown of ActionScript you insert in the Flash application:

```

1  //Create a NetConnection object
2  var connection_object:NetConnection = new NetConnection();
3  //Create a local streaming connection
4  connection_object.connect("rtmp://hostname/application_name");
5  //create a NetStream object
6  var stream_object:NetStream = new NetStream(connection_object);
7  //Attach the NetStream video feed to the video instance
8  instance_name.attachVideo(stream_object);
9  //Set the amount of video to buffer before beginning playback
10 stream_object.setBufferTime(number_of_seconds);
11 //Begin playing the FLV file
12 stream_object.play("virtual_path/video_name");

```

Some of the items in this script are taken from the parameters you entered when you provisioned your Akamai Streaming account:

- **hostname.** This is either the Akamai-generated hostname (e.g., cp9391.edgefcs.net) or one of the host aliases you created.
- **application_name.** This is either the default application name, “ondemand,” or one of the application name aliases you created.
- **virtual_path.** This is the virtual path name you defined when you initially created your Flash configuration. If you are streaming media other than Flash video (FLV), you must preface this with the appropriate extension as outlined in “Specifying Codecs with Akamai Streaming” on page 36.

▶ *Note: Because of the way Akamai Streaming is configured, you must use a virtual path name here. Using the origin location’s full path will render the stream inoperative.*

- **video_name.** This is the actual name of the video itself. If you are streaming Flash video (FLV) you must omit the file extension (i.e., use **video**, not **video.flv**). For other media types (e.g., H.264 and MP3), you must include the extension.

▶ *Note: All these parameters are case sensitive. For example, the application name “ondemand” must be typed in lowercase letters. Variations like “onDemand” may cause errors.*

So, an actual ActionScript might appear as follows:

```

1  var nc:NetConnection = new NetConnection();
2  nc.connect("rtmp://cp9391.edgefcs.net/ondemand");
3  var ns:NetStream = new NetStream(nc);
4  myVideo.attachVideo(ns);
5  ns.setBufferTime(5);
6  ns.play("movies/productDemo");

```

A Note on the NetStream.onStatus Event Handler and “Stream Not Found” Behavior

ActionScript’s NetStream object has an event handler, “NetStream.onStatus”, that is useful for producing behaviors when status and error messages are posted for the object. If you use this event handler, be aware you must include an additional parameter in your “NetStream.play” method for the **NetStream.Play.StreamNotFound** error code to be correctly handled.

In the aforementioned simple Flash application, the “NetStream.play” method takes the form of:

```
ns.play("movies/productDemo");
```

When passed to Akamai Streaming, the server goes through a default search sequence to find the video content:

1. Search for a live stream with this name. If no stream exists, then...
2. Search for a prerecorded stream with this name. If no stream exists, then...
3. Create a new live stream with this name.

Since your stream is prerecorded, step 2 is the only one that will produce a result, as long as the video is actually available. If it is not, the server continues to step 3, sending the **NetStream.Play.Start** status for the “new live stream” and thus never producing the **NetStream.Play.StreamNotFound** error code.

To produce the desired behavior using the “NetStream.onStatus” event handler, the “NetStream.play” method needs an additional parameter to indicate to the server that the video in question is prerecorded. In this case, the method appears as follows:

```
ns.play("movies/productDemo", 0);
```

0 represents the time point, in seconds, at which to begin video playback. In this case, it starts at the beginning (time 0 seconds). Since starting at a specific point in a video is impossible with a live stream, the server will bypass step 1, and if the prerecorded video is unavailable, will return the **NetStream.Play.StreamNotFound** error code.

On the other hand, if the stream was live, a value of -1 would be used to indicate the stream should begin at whatever data point is available at that time. Doing so would cause the server to bypass step 2 in the sequence.

Streaming MP3s

Akamai Streaming can also stream MP3 audio files. The required ActionScript is nearly identical to that used for streaming video with a couple of minor alterations:

- Preface the virtual path and video name with **mp3:**
- Substitute the **attachVideo** method with **attachSound**

Your ActionScript would then appear as:

```
1 var nc:NetConnection = new NetConnection();
2 nc.connect("rtmp://cp9391.edgefcs.net/ondemand");
3 var ns:NetStream = new NetStream(nc);
4 myAudio.attachSound(ns);
5 ns.setBufferTime(5);
6 ns.play("mp3:audio/productDemo");
```

Expediting the Connection

As explained at the beginning of this chapter, Adobe Flash Player normally attempts to communicate with Akamai Streaming in a particular protocol/port progression—RTMP:1935, RTMP:443, RTMP:80, and RTMPT:80 (or RTMPE:1935, RTMPE:443, RTMPE:80, and RTMPTE:80). This default progression has a disadvantage, however, in that Flash Player can take considerable time to progress completely through the sequence, resulting in a potentially lengthy delay in starting video playback. As a best practice to overcome this, Akamai strongly recommends overriding the default sequence to attempt simultaneous connections and expedite playback.

To accomplish this, two connections of different types are attempted concurrently: one using the default protocol/port sequence and the other attempting only HTTP tunneling (RTMPT or RTMPTE) over port 80. Whichever connection succeeds first is the one used; the other is dropped.

The AkamaiConnection class performs this function automatically. The following ActionScript example illustrates how to implement it when not using the class:

```

1  // Create two NetConnection objects. One for the default protocol/
   port sequence and the other to try HTTP tunneling on port 80
   (RTMPT or RTMPTE).
2  var ncdefault:NetConnection = new NetConnection();
3  var nctunnel:NetConnection = new NetConnection();
4  // Create a flag object to ensure a race condition does not occur
   between the two connections.
5  var connSuccess = 0;
6  // Create two streaming connections. One for the default protocol/
   port sequence and the other to try HTTP tunneling on port 80
   (RTMPT or RTMPTE).
7  ncdefault.connect("rtmp://cp9391.edgefcs.net/ondemand");
8  nctunnel.connect("rtmpt://cp9391.edgefcs.net:80/ondemand");
9  // Create a NetStream object.
10 var ns:NetStream;
11 // If the default protocol/port sequence (ncdefault) connects
   successfully first, drop and close the RTMPT:80 (nctunnel)
   connection and begin playing the video.
12 ncdefault.onStatus = function(ncObj) {
13     if(ncObj.code == "NetConnection.Connect.Success") {
14         if(connSuccess == 0) {
15             connSuccess = 1;
16             if(nctunnel) {
17                 delete nctunnel;
18                 nctunnel = null;

```

```

19         }
20         netStream = new netStream(this);
21         myVideo.attachVideo(ns);
22         netStream.setBufferTime(5);
23         netStream.play("movies/productDemo");
24     }
25 }
26 }
27 // If the HTTP tunneling attempt (nctunnel) successfully connects
    // first, drop and close the default sequence (ncdefault)
    // connection and begin playing the video.
28 nctunnel.onStatus = function(ncObj) {
29     if(ncObj.code == "NetConnection.Connect.Success") {
30         if(connSuccess == 0) {
31             connSuccess = 1;
32             if(ncdefault) {
33                 delete ncdefault;
34                 ncdefault = null;
35             }
36             netStream = new netStream(this);
37             myVideo.attachVideo(ns);
38             netStream.setBufferTime(5);
39             netStream.play("movies/productDemo");
40         }
41     }
42 }

```

Dealing with Problematic Client-Side Proxy Servers

It is possible for Flash video streamed from Akamai Streaming to encounter problems when confronted by certain client-side proxy servers. The problems occur if the proxy server attempts multiple reconnections during the course of a single stream.

In brief, the proxy server allows Flash Player to make an initial connection to Akamai Streaming, which uses its intelligent routing technology to select the Streaming server optimal to the end user. If the proxy server attempts subsequent reconnections during the stream's playback, however, the routing process begins again, likely resulting in selection of a different server. Since the new server has no context for the streaming session, a failure results. To get around this problem, it is necessary to connect to a single Akamai Streaming server for the stream's duration. Circumventing Akamai's routing technology is not desirable, so the solution must account for both of these.

The following two sections discuss the solution both with and without using the AkamaiConnection class. As a best practice, Akamai highly recommends implementing this solution to avoid any potential problems.

Solving the Problem Using the AkamaiConnection Class

If you are using the AkamaiConnection class, a solution is implemented automatically. The class was created so that the playback application first sends a query through Akamai's routing process to find the optimal Streaming server. The query then asks the server to retrieve its XML-formatted **ident** file, which contains its IP address in the **<ip>** tag.

```
<?xml version="1.0" encoding="utf-8" ?>
<fcs><ip>Akamai_Streaming_server_IP_address</ip></fcs>
```

The contents of the ident XML file are, in turn, sent to the playback application, and the server's IP address is extracted and substituted for your hostname; Flash Player then attempts to connect to Akamai Streaming using the IP address. While the IP address is now being used for connection, the playback application retains your hostname for identification purposes to ensure the correct Akamai Streaming account and Flash video are accessed.


Solving the Problem Without the AkamaiConnection Class

If you are not using the AkamaiConnection class in your playback application, you can still overcome the client-side proxy server problem by including the appropriate ActionScript in your application. While deployed differently, this solution behaves identically to that described in the previous section.

The solution is facilitated by two variables—`_global.serverIP` and `_global.hostName`—shown in lines 2 and 3 in the ActionScript solution below. The former represents the Streaming server's IP address and the latter your hostname. When you use the `nc.connect()` method (line 28), you reference both variables:

```
nc.connect("rtmp://" + _global.serverIP + "ondemand?_fcs_vhost=" + _global.hostName);
```

This allows playback by providing Akamai Streaming with both a dedicated IP address and your hostname.

 *Note: Be certain to include the `_fcs_vhost` query string, else playback will not occur.*

```
1 // Establish global variables for customer hostname and Server IP
  address.
2 _global.hostName="cp9391.edgefcs.net";
3 _global.serverIP;
4 var nc:NetConnection = new NetConnection();
5 // Create a new NetStream object, but do not assign the
  NetConnection object to it yet.
6 var ns:NetStream;
```

```

7  // Retrieve the server's ident XML file by issuing an HTTP request.
8  var myXML:XML = new XML();
9  myXML.ignoreWhite = true;
10 myXML.onLoad = myLoad;
11 myXML.load("http://" + _global.hostName + "/fcs/ident");
12 // If retrieval of the ident XML file was successful, call this
    function to extract the IP address from the file.
13 function myLoad()
14 {
15     output.text += "myLoad Function !!\n";
16     if (this.firstChild.hasChildNodes())
17     {
18         var aNode:XMLNode;
19         // use firstChild to iterate through the child nodes of rootNode
20         for (aNode = this.firstChild.firstChild; aNode !=
            null; aNode = aNode.nextSibling)
21         {
22             output.text +=
23                 aNode.nodeName + ": " + aNode.firstChild.nodeValue;
24             if (aNode.nodeName == "ip")
25             {
26                 // Assign the IP address to the _global.serverIP variable.
27                 _global.serverIp = aNode.firstChild.nodeValue;
28                 // Substitute the IP address for the hostname, but retain the
                    hostname for reference.
29                 nc.connect("rtmp://" + _global.serverIp + "/"
                    ondemand?_fcs_vhost=_global.hostName);
30             }
31         }
32     }
33 }
34 // Finish creating the NetStream object.
35 ns = new NetStream(nc);
36 }
37 }
38 }
39 // Connect to the Akamai Streaming server using Flash Player's
    default protocol:port rollover sequence.
40 nc.onStatus = function(ncObj) {
41     output.text += "\nonStatus rtmp://" + _global.serverIp + ncObj.code;
42     if (ncObj.code == "NetConnection.Connect.Success")
43     {

```


```

40     myVideo.attachVideo(ns);
41     ns.setBufferTime(5);
42     ns.play("movies/productDemo");
43 }
44 }

```

Using the “ondemand” Server-Side Application

As you know, Akamai Streaming supplies a server-side application called “**ondemand**.” This is actually the name of a directory on the Akamai Streaming server that holds, among other things, an ActionScript file called **main.asc**. To avoid confusion, it will continue to be referred to here as the “ondemand” application.

 *Note: Inclusion of the application name or an alias in your hostname path (e.g., **cp9391.edgefcs.net/ondemand**) is mandatory when using Flash Media Server, regardless of whether you use its features.*

The ondemand application provides two services: automatic bandwidth detection and stream-length reporting. Use of these services is facilitated by the AkamaiConnection class. If you are not using the class, however, you can still add ActionScript to your client-side application to use them.

Automatic Bandwidth Detection

Automatic bandwidth detection allows you to better serve your end users by permitting them to view video streams optimized for their Internet connection speed. To use this feature you create multiple versions of your video, optimized for different bandwidths, and place them on your origin server. You then include ActionScript in your client-side Flash application to tell the “ondemand” application to calculate the bandwidth from Akamai Streaming to the end user’s client and then retrieve and stream the version most appropriate to their connection speed (based on parameters you enter into the script).

The following example adds to the previous examples to calculate bandwidth upon connecting with the server. This is only for instances in which you *are not* using the AkamaiConnection class. If you *are* using the class, refer to its specification and examples for usage instructions.

```

1  //Create a variable to allow bandwidth detection upon connecting
   with the server
2  var bandwidthVariable = true;
3  var nc:NetConnection = new NetConnection();
4  //Tell the "ondemand" application to check the bandwidth and return
   the value to the client
5  nc.onBWCheck = function(arg) {

```

```
6     return;
7 }
8 //Based on the returned value, choose the appropriate video stream
9 nc.onBWDone = function(functionParameter) {
10 //If functionParameter is greater than or equal to highBW_threshold
    (in kbps), play the high-bandwidth video
11     if(functionParameter >= highBW_threshold) {
12         ns.play("virtual_path/high-bandwidth_video")
13     }
14 //If functionParameter is greater than or equal to midBW_threshold
    (in kbps), play the medium-bandwidth video
15     else if (functionParameter >= midBW_threshold) {
16         ns.play("virtual_path/medium-bandwidth_video")
17     }
18 //If functionParameter is less than midBW_threshold (in kbps), play
    the low-bandwidth video
19     else {
20         ns.play("virtual_path/low-bandwidth_video")
21     }
22 }
23 //Connect to server and, on successful connection, tell the server
    to perform bandwidth detection and return the result for video
    selection
24 nc.connect("rtmp://cp9391.edgefcs.net/ondemand",
    bandwidthVariable);
25 var ns:NetStream = new NetStream(nc);
26 myVideo.attachVideo(ns);
27 ns.setBufferTime(5);
```

Using this example, the end product might look like this:

```

1  var computeBW = true;
2  var nc:NetConnection = new NetConnection();
3  nc.onBWCheck = function(arg) {
4      return;
5  }
6  nc.onBWDone = function(bw) {
7      if(bw >= 300) {
8          ns.play("movies/productDemo_hi")
9      }
10     else if (bw >= 100) {
11         ns.play("movies/productDemo_med")
12     }
13     else {
14         ns.play("movies/productDemo_lo")
15     }
16 }
17 nc.connect("rtmp://cp9391.edgefcs.net/ondemand", computeBW);
18 var ns:NetStream = new NetStream(nc);
19 myVideo.attachVideo(ns);
20 ns.setBufferTime(5);

```

Stream Length Reporting

The other “ondemand” feature is stream length reporting, which extracts the video’s length from its metadata and reports it back to your playback application, making it possible to include such things as a progress bar. As with bandwidth detection, you add ActionScript to your application to take advantage of this feature. If you *are not* using the AkamaiConnection class, the ActionScript appears as follows (if you *are* using the class, refer to its specification and examples for usage instructions):

```

1  //Create a variable referring to the video "productDemo.flv".
2  var videoVariable = "movies/productDemo";
3  //Create a new object and callback function to get the video's
   length
4  var callbackFunction:Object = new Object();
5  callbackFunction.onResult = function(functionParameter) {
6      //Insert the statement of your choice here, based on how you
       would like to use the result in your Flash application
7  }
8  var nc:NetConnection = new NetConnection();
9  nc.connect("rtmp://cp9391.edgefcs.net/ondemand");
10 var ns:NetStream = new NetStream(nc);
11 myVideo.attachVideo(ns);
12 ns.setBufferTime(5);
13 ns.play("movies/productDemo");
14 //Get the video's length and return the result.
15 nc.call("getStreamLength", callbackFunction, videoVariable);

```

The final product in this case might look something like this:

```

1  var video = "movies/productDemo";
2  var result:Object = new Object();
3  result.onResult = function(length) {
4      //Your statement
5  }
6  var nc:NetConnection = new NetConnection();
7  nc.connect("rtmp://cp9391.edgefcs.net/ondemand");
8  var ns:NetStream = new NetStream(nc);
9  myVideo.attachVideo(ns);
10 ns.setBufferTime(5);
11 ns.play("movies/productDemo");
12 nc.call("getStreamLength", result, video);

```

Using Bandwidth Detection and Stream Length Reporting Together

Using the two “ondemand” application features together is a relatively simple matter of combining their scripts as shown in the following example (for applications not using the AkamaiConnection class):


```

1  var computeBW = true;
2  var streamHi = "movies/productDemo_hi";
3  var streamMd = "movies/productDemo_med";
4  var streamLo = "movies/productDemo_lo";
5  var result = new Object();
6  result.onResult = function(length) {
7      //Your stream length statement
8  }
9  var nc:NetConnection = new NetConnection();
10 nc.onBWCheck = function(arg)
11 {
12     return;
13 }
14
15 nc.onBWDone = function(bw)
16 {
17     if (bw >= 300) {
18         ns.play("movies/productDemo_hi")
19         nc.call("getStreamLength", result, streamHi)
20     }
21     else if (bw >= 100) {
22         ns.play("movies/productDemo_med")
23         nc.call("getStreamLength", result, streamMd)
24     }
25     else {
26         ns.play("movies/productDemo_lo")
27         nc.call("getStreamLength", result, streamLo)
28     }
29 }
30 nc.connect("rtmp://cp9391.edgefcs.net/ondemand", computeBW);
31 var ns:NetStream = new NetStream(nc);
32 myVideo.attachVideo(ns);
33 ns.setBufferTime(5);

```

Using Adobe Flash's Media Components

Akamai Streaming (for On Demand Adobe Flash Video) supports the use of the native Flash media components—MediaController, MediaDisplay, MediaPlayer, and FLVPlayback. These provide ready means of video playback that do not require the use of ActionScript.

 *Note: If you use these components, you will be unable to take advantage of the recommended connection-expedition and client-side-proxy solutions discussed earlier in this chapter. The exception to this is the FLVPlayback component when used in conjunction with the AkamaiNCManager ActionScript class described on page 51.*

Using these components is a simple matter of placing them on the stage and populating their **URL** (MediaDisplay and MediaPlayer) or **contentPath** (FLVPlayback) parameter with the Akamai Streaming path in the form:

`rtmp://hostname/application_name/virtual_path/video_name.flv`

With the Flash media components, there is no need to break the path into two segments as in the previous sections' ActionScript examples (inclusion of the .flv file extension is optional), so the path might be entered as:

`rtmp://cp9391.edgefcs.net/ondemand/movies/productDemo.flv`

These examples assume are valid if you are streaming a Flash video (FLV). If, however, you are streaming media of another format, you must form these URLs as outlined in "Specifying Codecs with Akamai Streaming" on page 36.

Using Flash Media Components with H.264- and HE-AAC-Encoded Media

Of all the Flash media components, only the FLVPlayback component is able to stream media encoded with the H.264 video and HE-AAC audio codecs. To take advantage of this capability, however, you must first install Adobe Flash Player 9 Update 3 (version 9.0.115.0), which includes an update to the FLVPlayback component. You can obtain the update at http://www.adobe.com/shockwave/download/download.cgi?P1_Prod_Version=ShockwaveFlash.

Using the AkamaiNCManager ActionScript Class

The AkamaiNCManager class allows Flash developers to use the FLVPlayback 1.0.1+ component to connect to Akamai Streaming and Akamai HTTP Content Delivery to deliver streaming and progressive FLV files, respectively.

While, as previously stated, the FLVPlayback component can be used to connect to Akamai without any modification, using the AkamaiNCManager class in conjunction with the component provides advantages to the developer:

- It implements client-side proxy penetration by requesting the IP address request of the optimal server, thereby avoiding dynamic DNS problems as explained on page 44).
- It allows you to use progressive download and include parameters in the path.

- It allows you to use aliases in lieu of “ondemand” for your server-side application name.

You can obtain the class file from the EdgeControl portal’s documentation area ([On Demand Streams >> Documentation](#)). Deploying it is simple:

1. Place the AkamaiNCManager.as file in the same directory as the FLA file that will instantiate the FLVPlayback component.
2. Ensure you are using version 1.0.1 or higher of the FLVPlayback component.

To query the version, execute the ActionScript `trace()` method, thusly:

```
1 trace(mx.video.FLVPlayback.version)
```

If you need to update your version of the component, you can do so at <http://www.adobe.com/support/flash/downloads.html>

3. Include the following lines of ActionScript:

```
1 var __forceAkamaiNCManager:AkamaiNCManager;
2 mx.video.VideoPlayer.DEFAULT_INCMANAGER = "AkamaiNCManager";
```

4. Set the `contentPath` property, either included as a third line of ActionScript or in the component’s API, and proceed as normal (the component’s full API remains available).

Example ActionScript

This example assumes the FLVPlayback instance on the stage is named `player`.

```
1 var __forceAkamaiNCManager:AkamaiNCManager;
2 mx.video.VideoPlayer.DEFAULT_INCMANAGER = "AkamaiNCManager";
3 player.contentPath = "http://example.edgesuite.net/folder/
  myfile.flv?param1=abc";
```

Usage Notes

- The class does not currently support SMIL sources. Developers wishing to connect to these sources should use the AkamaiConnection class.
- The class supports Akamai URLs for the following formats:
 - Standard on-demand Streaming connection with default server-side application name:
`rtmp://cp9391.edgefcs.net/ondemand/folder/myfile.flv`
 - On-demand Streaming connection with server-side application name alias:
`rtmp://cp9391.edgefcs.net/alias_1/folder/myfile.flv`
 - Progressive download link with parameters:
`http://example.edgesuite.net/folder/myfile.flv?param1=abc`

Chapter 5. Using Secure Streaming

In This Chapter



Secure Streaming Guidelines for On Demand Flash Video • 53

Using an slist • 54

Integrating Secure Streaming for On Demand Flash Video • 56

Akamai Streaming offers a feature for Flash video called Secure Streaming that provides additional control over content by giving you the ability to prevent unauthorized access to your streams. Scenarios in which this is desirable include:

- Pay per view
- Deep linking prevention
- Internal Webcast/IP validation
- Affiliate tracking



Note: Secure Streaming is an additional service you must include in your Akamai contract to use. It is unavailable otherwise.

Before proceeding, it is strongly recommended you read the *Akamai Secure Streaming Integration Guide*. While the specifics of that document are oriented toward other streaming formats, it also contains important general information regarding Secure Streaming concepts. Familiarizing yourself with the document will prepare you for the information presented in this chapter, which focuses on areas specific to Flash video.

Secure Streaming Guidelines for On Demand Flash Video

Following are some Secure Streaming guidelines

- By default, Secure Streaming for Flash video is applied at the CP code/hostname level, not at directory or file levels as is the case with other formats. This is called Per Connect Secure Streaming, and if a Flash configuration has it enabled, all origin directories and videos associated with that particular configuration's hostname will have it applied to them. In addition, all aliases defined for the hostname will also require Secure Streaming authentication.
- You may apply Per Connect Secure Streaming to individual directories and files within the secured hostname by using an **slist** (see “Using an slist” below).
- Once an origin location (an origin server hostname and a specific associated path (e.g., **example.download.akamai.com/9389**)) is included Flash configuration enabled with Per Connect Secure Streaming, you may not add that location to a different configuration not having Secure Streaming. The opposite is also true.

You may, however, use the same origin server hostname in a non-secure Flash configuration if you specify a different path (be certain the paths do not in any way overlap). For example:

Virtual Path	Actual Path
secureflash	example.download.akamai.com/9389
nonsecureflash	example.download.akamai.com/9392


- Secure Streaming for Flash video supports D- and E-type tokens only, and these tokens must not exceed 511 bytes in length.
- All Secure Streaming tokens must include a fingerprint (**aifp**) parameter, which you must change each time you change the stream's Secure Streaming profile. Following is an example of a Secure Streaming token with its fingerprint parameter:

```
auth=damcPcIbVcscXdQcPbnd7d_dnbSaqaycgay-EDrBic=&aifp=1234
```

- Secure Streaming for Flash video does not support the rendering duration (playback-duration) token parameter; Flash Player will continue playback even after the specified duration is reached. In addition, it is unable to stream an alternative video should authentication fail.
- When including the path in token generation, the path should be either `/hostname/application_name` (e.g., `-p /flash.example.com/ondemand`) or the `slist` if using one (see “Using an slist” below).


Using an slist

An slist allows you to specify one or more particular directories and/or files to which you want Per Connect Secure Streaming applied. This adds an extra layer of security in that you include the slist in the token generator, and it is then passed in both the token and the Web page's markup. When the Akamai Streaming server detects the **slist** keyword in the markup, it recognizes that it should look for a match within the token. If no match is present, the content is not served.

 *Note: The parameters included in the token generator and those in the slist argument must match **exactly**, else a failure results.*

An example in which this feature is useful is when you are using the server-side ondemand application's automatic bandwidth detection feature (see “Automatic Bandwidth Detection” on page 47). In this case, more than one version of the same video, optimized for different bandwidths, is available, but only one is streamed based on the client's detected bandwidth. The video request is passed to the Akamai Streaming server before the client's bandwidth is known, however, so by passing all possible videos in an slist, the server applies Per Connect Secure Streaming to whichever is actually streamed.

The parameters passed in the slist for Per Connect Secure Streaming are identical in form to those passed to your playback application. That is, the slist parameters are given in relation to your virtual path. For example, you may have created a virtual path in a Flash Configuration that resolves the virtual path **movies** to the actual path **example.download.akamai.com/9389/movies/**. If you wanted to apply Secure Streaming to a particular video called “productDemo.flv”, you would pass **movies/productDemo** to your Flash playback application, and you would use the same value in your token and as your slist argument.

 *Note: For MP3, H.264, and HE-AAC files to work correctly in an slist, you must omit the codec prefix (**mp3:** and/or **mp4:**) from both the slist and the token. The **play()** call in your client-side playback application, however, must still contain these prefixes. Refer to “Specifying Codecs with Akamai Streaming” on page 36 for more information on this topic.*

*On a related note, the file extensions in your slist must exactly emulate those in your **play()** call.*

As previously stated, you can apply Per Connect Secure Streaming to both individual files and entire directories of files. For files, the parameter is passed in the token and slist argument as:

virtualpath/video or **virtualpath/directory/video**

Delimit multiple entries with semicolons (;), thusly:

virtualpath/directory/video1;virtualpath/directory/video2

If you wish to apply Per Connect Secure Streaming to the entire contents of a directory, you pass the parameter in a similar fashion, though omitting the video name at the end:

virtualpath/directory/

The trailing slash is not strictly required in this case, but is recommended for clarity. Again, delimit multiple directories with semicolons.

Using Secure Streaming with Dynamic Streaming

If you plan to use Secure Streaming alongside the Dynamic Streaming feature (see “Using Dynamic Streaming” on page 37), you must use an slist to specify each video encoded for use with the feature. If you do not, playback will be interrupted if Dynamic Streaming attempts to switch to a different bitrate version, since that version will not have been authorized by the Secure Streaming token. As previously stated, you can specify these videos by name, or by the directory in which they are housed.

Integrating Secure Streaming for On Demand Flash Video

Per Connect Secure Streaming integration results in passing the Secure Streaming authentication token to the Akamai Streaming service, allowing playback of the desired video. You can do this in two ways:

- Pass the token in your SWF's ActionScript's **connect()** method string.
- Pass the token in the referer header.

▶ *Note: Should tokens be passed via both methods for the same video request, the token in the **connect()** method takes precedence. Additionally, passing an invalid token results in a **NetConnection.Connect.Rejected** event.*

Passing the Token in the **connect()** Method

Akamai Streaming (for Adobe Flash Video) allows you to pass Per Connect Secure Streaming tokens in your ActionScripts' **connect()** methods, providing you more flexibility in the way you apply Secure Streaming. While Akamai also supports passing tokens in the referer header (see below), using the **connect()** method provides advantages over that, including:

- **Real-time authentication.** The token is passed to the server when the video is requested, not when the SWF is requested. This eliminates the possibility of the token expiring before the end user actually attempts to play a video.
- **Facilitating authenticated playlists.** Using the **connect()** method to pass your tokens is useful for playlists. In a typical scenario, the **XML.load()** method is used in the SWF's ActionScript to call an XML-formatted playlist from an origin server, the contents of which are used by the SWF to present a list of videos to the end user. When the end user chooses one to play, an authentication token is generated and is sent to Akamai Streaming in the **connect()** method string.

You can pass the token in the **connect()** method whether you are using the Akamai-Connection class or not. If you are *not* using it, the **connect()** method takes the following form:

```
connect("[hostname]/[application_name]?auth=[token]&aifp=[fingerprint]")
```

So, an actual **connect()** method might look like this:

```
connect("rtmp://cp9391.edgefcs.net/ondemand?auth=damcPcIbVcscXdQcPbnd7d_dnbSaqaycgay-EDrBic=&aifp=1234")
```

If you are using an slist simply append it to the token, thusly (the slist here includes three bandwidth-optimized videos):

```
connect("rtmp://cp9391.edgefcs.net/ondemand?auth=damcPcIbVcscXdQcPbnd7d_dnbSaqaycgay-EDrBic=&aifp=1234&slist=movies/productDemo_56;movies/productDemo_100;movies/productDemo_300")
```

If you *are* using the AkamaiConnection class, its specification provides information for adding the token to the **connect()** method.

Additionally, the AkamaiNCManager class allows you to pass the token in Flash's native FLVPlayback component. Refer to the class specification for details.

Passing the Token in the Referer Header

You can dynamically generate your Web pages to include the Per Connect Secure Streaming authentication token, either in the HTML markup or using a JavaScript. From there, Akamai Streaming passes the token in the referer header for validation. In both instances, the method for passing the token to the JavaScript or HTML markup is completely up to you.

Inserting the Token into HTML

The following example shows how to insert a Per Connect Secure Streaming token into HTML markup. It includes three bandwidth-optimized videos and uses the `slist` feature.

```

1  <html>
2  <body>
3  <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://download.macromedia.com/pub/shockwave/cabs/
    flash/swflash.cab#version=7,0,0,0" width="320" height="240"
    id="FLVPlayer" align="middle">
4  <param name="movie" value="http://www.example.com/movies/
    FlashPlayer.swf?auth=damcPcIbVcscXdQcPbnd7d_dnbSaqaycgay-
    EDrBic=&aifp=1234&slist=movies/productDemo_56;movies/
    productDemo_100;movies/productDemo_300&skinName=http://
    www.example.com/movies/FlashPlayer/
    blueSkin&serverName=cp9391.edgefcs.net&fallbackServerName=cp9
    391.edgefcs.net&appName=ondemand&streamName=movies/
    productDemo_56, 50, movies/productDemo_100, 150, movies/
    productDemo_300&isLive=false&bufferTime=5&autoPlay=true&autoR
    ewind=true&isFullScreen=false"/>
5  <param name="salign" value="lt"/>
6  <param name="quality" value="high"/>
7  <param name="scale" value="noscale"/>
8  <param name="bgcolor" value="#ffffff"/>
9  <embed src="http://www.example.com/movies/FLVPlayer/
    FlashPlayer.swf?auth=damcPcIbVcscXdQcPbnd7d_dnbSaqaycgay-
    EDrBic=&aifp=1234&slist=movies/productDemo_56;movies/
    productDemo_100;movies/productDemo_300&skinName=http://
    www.example.com/movies/FlashPlayer/
    blueSkin&serverName=cp9391.edgefcs.net&fallbackServerName=cp9
    391.edgefcs.net&appName=ondemand&streamName=movies/
    productDemo_56, 50, movies/productDemo_100, 150, movies/
    productDemo_300&isLive=false&bufferTime=5&autoPlay=true&autoR
    ewind=true&isFullScreen=false" quality="high" scale="noscale"
    bgcolor="#ffffff" width="320" height="240" name="FlashPlayer"
    salign="LT" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer"/>

```

```

10 </object>
11 </body>
12 </html>

```

Inserting the Token into JavaScript

Another way to pass parameters is using JavaScript. In this example, the `slist` feature is used to apply Per Connect Secure Streaming to an entire directory of files. Here, line 13 passes the token, line 14 passes the required fingerprint value (Refer to the *Akamai Secure Streaming Integration Guide*), and line 15 passes the `slist` argument.

```

1 <html>
2 <body>
3 <div align="center" id="videoDiv">
4 <script language="javascript">
5 <!--
6 function appendParameter(p_args, p_name, p_value) {
7   if (p_args == "")
8     return p_name+"="+escape(p_value);
9   else
10    return p_args+"&"+p_name+"="+escape(p_value);
11 }
12 var args = "";
13 args=appendParameter(args, "auth",
14   "damcPcIbVcscXdQcPbnd7d_dnbSaqaycgay-EDrBic");
14 args=appendParameter(args, "aifp", "1234");
15 args=appendParameter(args, "slist", "movies/");
16 args = appendParameter(args, "serverName", "cp9391.edgefcs.net");
17 args = appendParameter(args, "fallbackServerName",
18   "cp9391.edgefcs.net");
18 args = appendParameter(args, "skinName", "blueSkin");
19 args = appendParameter(args, "appName", "ondemand");
20 args = appendParameter(args, "streamName", "movies/productDemo");
21 args = appendParameter(args, "isLive", "false");
22 args = appendParameter(args, "bufferTime", "5");
23 args = appendParameter(args, "autoPlay", "true");
24 args = appendParameter(args, "autoRewind", "true");
25 args = appendParameter(args, "bgColor", "0xFFFFFF");
26 document.write('<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000" codebase="http://download.macromedia.com/pub/
shockwave/cabs/flash/swflash.cab#version=7,0,0,0" width="320"
height="240" id="FLVPlayer" align="" /> ');

```

```
27 document.write('<param name="movie"
    value="FlashPlayer.swf?'+args+' " /> ');
28 document.write('<param name="salign" value="lt" /> ');
29 document.write('<param name="quality" value="high" /> ');
30 document.write('<param name="scale" value="noscale" /> ');
31 document.write('<param name="bgcolor" value="#ffffff" /> ');
32 document.write('<embed src="FlashPlayer.swf?'+args+' "
    quality="high" scale="noscale" bgcolor="#ffffff" width="480"
    height="360" name="FlashPlayer" salign="LT" type="application/
    x-shockwave-flash" pluginspage="http://www.macromedia.com/go/
    getflashplayer" /> ');
33 document.write('</object>');
34 -->
35 </script>
36 </div>
37 </body>
38 </html>
```


Chapter 6. Changing Flash Video Content

In This Chapter



Purging the Video from the Akamai Streaming Cache • 61

There may be situations in which you will want to replace a Flash video in Akamai Streaming with another of the same file name. Doing this on your origin location is simply a matter of uploading the new video to take the place of the old. Akamai Streaming Edge servers, however, may retain the old video in their caches, requiring you to use Akamai's CCU (Content Control Utility) to purge those caches to allow end users to play the new video. This chapter discusses the process in more detail.



Note: If you are using the Dynamic Streaming feature (see “Using Dynamic Streaming” on page 37), you must purge every version of the video in question. Failure to do so can result in unpredictable behavior.

Purging the Video from the Akamai Streaming Cache

Before using the CCU to purge Akamai Streaming's cache, you must construct a purge URL for your content.

Constructing Purge URLs

The purge URL is fairly simple to construct, being a combination of elements with which you are already familiar: hostname, application name, virtual path, and file name. In short, you simply combine the information you entered into your SWF file's ActionScript parameters. So, for example, assume your ActionScript contains the following:

```
1 var nc:NetConnection = new NetConnection();
2 nc.connect("rtmp://cp9391.edgefcs.net/ondemand");
3 var ns:NetStream = new NetStream(nc);
4 myVideo.attachVideo(ns);
5 ns.setBufferTime(5);
6 ns.play("movies/productDemo");
```

You will combine information from lines 2 and 6, making a purge URL of:

`rtmp://cp9391.edgefcs.net/ondemand/movies/productDemo.flv`




*Note: Your purge URL must include the video's file extension **and** must omit the codec prefix discussed in “Specifying Codecs with Akamai Streaming” on page 36.*

If you don't have access to the ActionScript, you can obtain most of this information (except the video file name) from the appropriate Flash Configuration details display

on your Akamai Streaming account's **Flash Configurations** page as shown in the following example:

CP Code	Stream Name	Options
9381	Product Demonstrations	Hide Details Edit Delete Test
<div> <div>Hostnames: cp9381.edgefcs.net (default) products.example.com</div> <div>Application Name Aliases: ondemand (default) stream vod</div> <div>Per Connect Secure Streaming: Enabled Configure Profiles</div> <div>SWF Authentication Enabled: Enabled</div> <div>SWF Path: example.download.akamai.com/9389/swfauth/productdemos/</div> <div>Virtual Path Name: /contraptions</div> <div>Actual Path: example.download.akamai.com/9389/contraptions</div> <div>Virtual Path Name: /gizmos</div> <div>Actual Path: www.example.com/gizmos</div> </div>		

Figure 6-1. Flash Configuration Details

 *Note: Your purge URL may contain any of the host and application name aliases you created for your Flash configuration.*

Purging the Akamai Streaming Cache

You must log in to your Akamai Streaming account to purge the cache.

1. Access the CCU.
 - a. Log in to EdgeControl portal.
 - a. In the left-hand navigation menu, click [On Demand Streams](#).
The **On Demand Streaming Traffic** page appears.
 - b. In the left-hand navigation menu under [On Demand Streams](#), click [Content Control Utility](#).

The Purge Content page appears.

Content Control Utility

Refresh by URL | Refresh by Directory & File Extension

This utility allows you to quickly refresh the content cached in Akamai edge servers without waiting for expiration as specified in your edge server configuration. [Learn More.](#)

Note: URLs entered in the form or read from a file should be **full**, not partial, URLs and should **not** contain query strings.

Purging Flash Content?

Content to Refresh:

☒ **URLs/ARLs entered below**
 Enter one URL/ARL per line. URL or ARL entries should be full, not partial. You can specify http://, or https://, rtmp://, or mms://, as protocols; the default behavior is to refresh the objects without regard to the protocol specified.

 For example, if you specify http://www.example.com/graphics/logo.gif, Akamai will by default refresh that object and the object cached for https://www.example.com/graphics/logo.gif, if there is one.

☐ URLs/ARLs specified in a **File** >>

☐ [Use with **Caution!**] Refresh **ALL** URLs/ARLs Associated with Specific **CP Codes** >>

Refresh Method:

☒ **Purge**
 Remove the content from Akamai edge server caches. The next time the edge server receives a request for the content, it will retrieve the current version from the origin server. If it cannot retrieve a current version, it will follow instructions in your edge server configuration.

Note that "Purge" can increase the load on the origin more than "Invalidate". With 'invalidate', objects are not removed from cache and full objects are not retrieved from the origin unless they are newer than the cached versions.

☐ **Invalidate**
 Mark the cached content as invalid. The next time the Akamai edge server receives a request for the content, it will send an HTTP conditional get (If-Modified-Since) request to the origin. If the content has changed, the origin server will return a full fresh copy; otherwise, the origin normally will respond that the content has not changed, and Akamai can serve the already-cached content.

Notification:

☐ Don't notify me when this content has been refreshed

☒ Send an email to the following comma separated email address(es) when the content has been refreshed:

ccare2_nomail@akamai.com


Start Refreshing Content

Figure 6-2. The Content Control Utility

2. Purge the Flash video from the Akamai Streaming cache.

- a. In the **Content to Refresh** section, select the **URLs/ARLs entered below** radio button and enter your purge URL in the box; separate multiple purge URLs with a new line.
- b. In the **Refresh Method** section, select the **Purge** radio button.
- c. In the **Notification** section, select the desired radio button to either receive or not receive an e-mail notification upon removal of the Flash video file or files.

If you choose to receive notification, enter your e-mail address in the text box.

- d. Scroll to the bottom of the page and click .

The video will be purged from Akamai Streaming. Afterward, the next end-user request for the video will cause the service to download the new video from your origin location.



Note: Be aware that cache purges can take half an hour or longer to take effect.

Appendix A. Using the Flash Support Player

To assist you in troubleshooting your streaming Flash video, Akamai makes available the Flash Support Player, which is useful for testing and troubleshooting. The player is also publicly available, so you can use it to help resolve problems with your end users.

1. Access the Flash Support Player.
 - a. Start your Web browser and open <http://support.akamai.com/flash/index.html>.

The player appears.

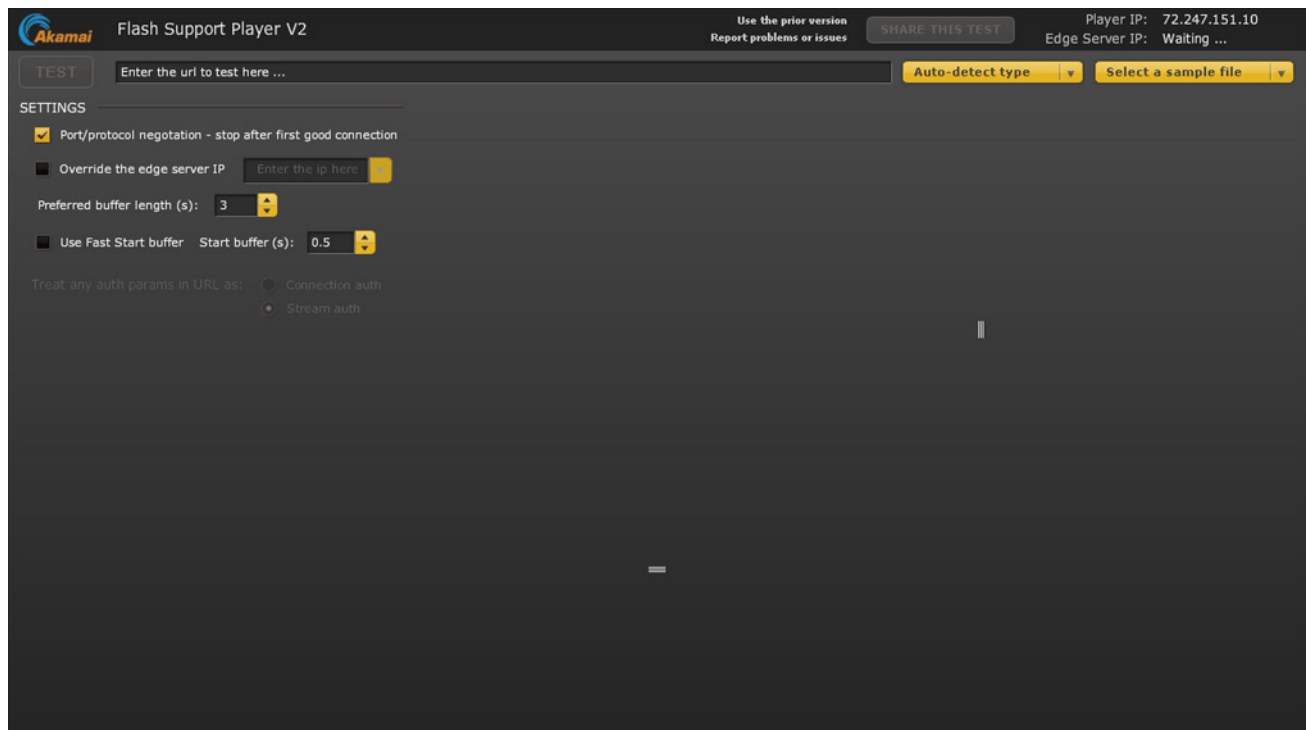


Figure A-1. Flash Support Player

2. Test the configuration.
 - a. In the **Enter the url to test here ...** text box, enter your video's full URL.

If your media is something other than a Flash video (.flv), you must include the appropriate **mp4:** or **mp3:** prefix as outlined in "Specifying Codecs with Akamai Streaming" on page 36.

- b. If you wish the player to limit itself to the first successful connection, foregoing subsequent connection attempts on remaining port and protocol combinations, select the **Port/protocol negotiation - stop after first good connection** check box.
- c. If you wish to test against a specific Akamai Streaming server rather than using Akamai's normal mapping feature, select the **Override the edge server IP** check box and enter the server's IP address in the accompanying text box. (Refer to "Overriding the Server IP" on page 68 for more information.)
- d. If you wish to override the player's default 3-second video buffer, use the **Preferred buffer length (s)** spin box to specify the number of seconds to buffer.
- e. If you wish to use the Fast Start feature, select the **Use Fast Start buffer** check box and use the **Start buffer (s)** spin box to specify the number of seconds (to a level of tenths of a second) to buffer.

With this feature enabled, the playback buffer is set to the specified number of seconds to quickly begin playback. As soon as the buffer fills completely, the buffer resets to the value entered in the **Preferred buffer length (s)** spin box.

- f. Click the **TEST** button.

A media player appears—assuming all entered information is valid—playing the media and providing various pieces of pertinent information such as the Edge server IP address, the test results of various port and protocol combinations, the activity trace, playback time, frame rate, video dimensions, video and audio bitrates, and the codec with which the media is encoded (see Figure A-2 on the next page).

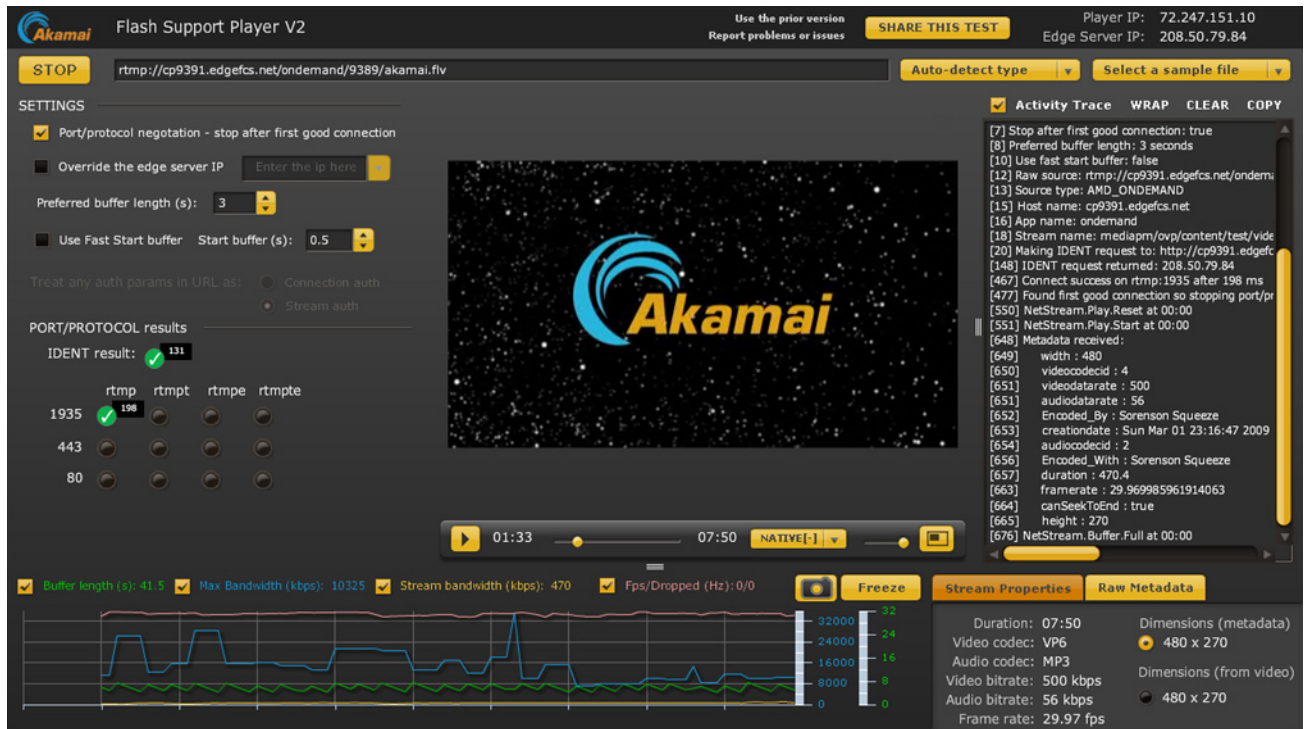


Figure A-2. Flash Support Player with Video Playback

If you wish to demonstrate your video's playback to others, click the **SHARE THIS TEST** button to open a new window containing a playback URL that you can copy and paste.

In addition, using the player's control, you can pause and restart video playback, change to a full-screen display, adjust the audio volume, view raw metadata, and start, freeze, and capture a graphical display of up to four (4) data sets (buffer length, maximum bandwidth, stream bandwidth, and frames per second dropped).

Troubleshooting with the Flash Video Test Player

The Flash video test player is also a useful troubleshooting tool, providing both stream data and the ability to specify streaming from a particular Akamai Streaming server. Using it as a go-to tool for both you and your end users can help expedite the problem-solving process.

Using Trace Data

The Flash video test player generates trace data each time you test a video stream. It is essentially a play-by-play report on your stream's status and can be viewed in the window on the player's right-hand side.

If you are experiencing problems with your stream, you can test it in the test player, click **COPY**, and paste the text into an e-mail to your Akamai representative. This will facilitate the troubleshooting process.

Overriding the Server IP

This feature of the Flash Support Player is helpful in determining whether a problem is occurring at the first or last mile of a stream.

Normally, when testing a stream with the Flash Support Player, the Akamai Streaming server is chosen dynamically, just as it would be in a real-world environment, with the result appearing in the Support Player's upper right-hand corner. You can, however, override this behavior to test a specific server:

1. Open the test player and set it up using the steps outlined earlier in this appendix, but do not click the **TEST** button yet.
2. Select the **Override the edge server IP** check box, and enter the server IP address in the accompanying field.

The server IP address can be obtained either using the Flash video test player or the **netstat** command.

3. Click the **TEST** button to initiate playback.

For example, if an end user experiences difficulties with your stream, they can provide you the IP address of the Akamai Streaming server to which they are connecting, and you can try to duplicate their problem by testing the stream using the same server. If you do not experience the same issues as the end user, the problem most likely resides at their end, rather than on the Akamai Streaming network.

Appendix B. Using the Progressive Download Alternative

While not part of Akamai Streaming (for On Demand Adobe Flash Video), progressive downloads of Flash videos are supported via Akamai's HTTP Content Delivery service. This can be useful should you encounter streaming obstacles (firewalls and proxy servers, for example) that you have been otherwise unable to overcome. This is possible because progressive download uses pure HTTP delivery, not RTMP, RTMPE, and HTTP tunneling.

Before using progressive download, however, there are a few issues to be aware of:

- Progressive download is only supported by Adobe Flash Player 7 or higher.
- Because it is downloaded, end users can only seek to portions of the video that they have actually received.
- The video is less secure since it is downloaded to end users' hard drives.

Since progressive download is not served from Akamai Streaming, you will need a separate Akamai HTTP Content Delivery account from which to serve your videos, though you may use the same origin server or servers as with Akamai Streaming. Setting up and configuring HTTP Content Delivery is beyond the scope of this document, but copious documentation on the topic is available in the EdgeControl portal ([HTTP Content Delivery >> Documentation](#) (or [Documentation >> HTTP Content Delivery](#))).

As is the case with streaming Flash video, you must use ActionScript in your client-side ActionScript to pass the hostname and video parameters. With progressive download, however, there is no need to pass these as two separate parameters. Here, you simply populate the "NetConnection.connect" method with a null value and pass the video's full path in the "NetStream.play" method as in the following example.

```
1 var nc:NetConnection = new NetConnection();
2 nc.connect(null);
3 var ns:NetStream = new NetStream(nc);
4 myVideo.attachVideo(ns);
5 ns.setBufferTime(5);
6 ns.play("http://a836.g.akamai.net/7/836/9391/e358f5db0045e9/
  movies.example.com/productDemo.flv");
```



Note: With progressive downloads, you must include the file's extension in the path.

Here, the path is an Akamaized® URL (ARL), but if you have an HTTP Content Delivery configuration file set up, you may use the DNS CNAME feature to resolve your domain to your HTTP Content Delivery hostname. The path then becomes **http://movies.example.com/productDemo.flv**. In either case, the optimal Edge server retrieves the video from your origin (if not already cached) for the end user.

