Using
ADOBE® HTTP DYNAMIC STREAMING

© 2010 Adobe Systems Incorporated and its licensors. All rights reserved.

Using Adobe® HTTP Dynamic Streaming

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

This user guide is licensed for use under the terms of the Creative Commons Attribution Non-Commercial 3.0 License. This License allows users to copy, distribute, and transmit the user guide for noncommercial purposes only so long as (1) proper attribution to Adobe is given as the owner of the user guide; and (2) any reuse or distribution of the user guide contains a notice that use of the user guide is governed by these terms. The best way to provide notice is to include the following link. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/

Adobe, the Adobe logo, ActionScript, Flash, Flash Access, and the HTTP Dynamic Streaming logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

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

Updated Information/Additional Third Party Code Information available at http://www.adobe.com/go/thirdparty.

Portions include software under the following terms:

This product contains either BSAFE and/or TIPEM software by RSA Security, Inc.

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

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

Last updated 5/19/2011

Contents

Chapter 1: Getting Started streaming on-demand media over HTTP

Overview 1

Tutorial 4

Chapter 2: Installing and configuring the HTTP Origin Module

About the HTTP Origin Module 7

Installing and configuring the HTTP Origin Module 8

Logging 10

Chapter 3: Packaging on-demand media

Packaging on-demand media 11

File Packager reference 15

Chapter 1: Getting Started streaming on-demand media over HTTP

Adobe® HTTP Dynamic Streaming is a solution that allows you to stream live and on-demand content over HTTP to Adobe® Flash® Player 10.1 and AIR 2. When content streams over HTTP, clients can seek quickly to any location. HTTP Dynamic Streaming supports adaptive (multi-bitrate) streaming, DVR functionality, and Adobe® Flash® Access™ protection.

This document contains information about on-demand HTTP Dynamic Streaming only.

To stream live content over HTTP, use Flash Media Server 4.0 and later. See Streaming media over HTTP in the *Flash Media Server Developer's Guide*.

Overview

On-demand HTTP Dynamic Streaming Components

Note: *For information about live HTTP Dynamic Streaming, see Streaming media over HTTP in the Flash Media Server Developer's Guide.*

The Adobe HTTP Dynamic Streaming solution contains the following components for on-demand streaming:

File Packager A command-line tool that translates on-demand media files into fragments and writes the fragments to F4F files. The File Packager is an offline tool. You can use the File Packager to encrypt files for use with Flash Access. For more information, see "Packaging on-demand media" on page 11.

The File Packager is available from adobe.com and is installed with Adobe® Flash® Media Server to the *rootinstall*/tools/f4fpackager folder.

HTTP Origin Module An Apache HTTP Server module that serves the on-demand files created by the File Packager and the live streams created by Flash Media Server. For more information, see "Installing and configuring the HTTP Origin Module" on page 7.

The HTTP Origin Module is available from adobe.com and is installed with Adobe® Flash® Media Server 4.0 and later.

OSMF media players Built on the Open Source Media Framework (OSMF), these media players runs in Flash Player 10.1. You can use Flash Media Playback, Strobe Media Playback, and the OSMF Sample Player for HTTP Dynamic Streaming.

For more information and to download the media players, see osmf.org.

F4F File Format Specification The F4F file format describes how to divide media content into segments and fragments. Each fragment has its own bootstrap information that provides cache management and fast seeking. For more information, see F4F File Format Specification.

F4M File Format Specification The Flash Media Manifest file format contains information about a package of files that the HTTP Origin Module can serve. Manifest information includes codecs, resolutions, and the availability of files encoded at multiple bit rates. Manifest information also includes DRM data. For more information, see F4M File Format Specification.

Flash Access Flash Access delivers protected media to Flash Player. To use HTTP Dynamic Streaming with Flash Access, use the File Packager and Flash Media Server to both package and encrypt content. For more information, see Protecting content with Flash Access.

System requirements

The following are the system requirements for on-demand HTTP Dynamic Streaming:

Component	Requirement
File Packager	The File Packager is a 32-bit binary that runs on the following operating systems: • Microsoft® Windows Server® 2008 32-bit • Red Hat® Linux® 5.1 32-bit
HTTP Origin Module	The HTTP Origin Module is a 32-bit binary extension of Apache HTTP Server version 2.2, 32-bit. The HTTP Origin Module and Apache HTTP Server run on the following operating systems: • Microsoft® Windows Server® 2008 32-bit • Red Hat® Linux® 5.1 32-bit The stand-alone HTTP Origin Module supports on-demand HTTP Dynamic Streaming only. It does not support live streaming. **Important:***To run the stand-alone HTTP Origin Module on Windows, download and install Microsoft Visual C++ 2005 Redistributable Package and Microsoft Visual C++ 2008 Redistributable Package.*
OSMF media players	OSMF 1.0
Runtimes	Flash Player 10.1, AIR 2
Flash Access	2.0
File formats and codecs	The File Packager and Live Packager convert F4V/MP4 and FLV files to F4F files. HTTP Dynamic Streaming supports the H.264, VP6, MP3, and AAC codecs that Flash Player supports. If a file plays in Flash Player, HTTP Dynamic Streaming (with or without Flash Access) can package it and serve it. **Important:** *HTTP Dynamic Streaming 1.0 does not support the MP3 file format or Speex audio.*

Use cases

On-demand HTTP Dynamic Streaming supports the following use cases:

- On-demand content
- On-demand multi-bitrate content (also called “adaptive streaming” and “dynamic streaming”)
- Flash Access content protection

On-demand workflow

The following is the workflow for all on-demand streaming use cases:

1 Install and configure the HTTP Origin Module. Do one of the following:

- Install Flash Media Interactive Server 4.0.
- Install the HTTP Origin Module to any installation of Apache HTTP Server 2.2.

See "Installing and configuring the HTTP Origin Module" on page 7.

2 Run the File Packager to process media files. Do one of the following:

- To package a single media file, see "Package content on Windows" on page 12 or "Package content on Linux" on page 12.
- To package multi-bitrate content, use the File Packager to create a manifest file that contains information about each encoded media file. See "Packaging multi-bitrate content" on page 13.
- To encrypt content for Flash Access, specify encryption options. See "Packaging and encrypting content" on page 14.

3 Copy the files output from the File Packager to a location on the Apache HTTP server under the DocumentRoot folder.

The DocumentRoot folder is the folder from which Apache serves files.

4 Create a crossdomain.xml file and copy it to the Apache DocumentRoot folder.

A crossdomain.xml file allows Flash Player clients hosted on other domains to access data from this domain. For more information, see Website controls (policy files) in the ActionScript 3.0 Developer's Guide.

5 Use the OSMF Player to play the media in Flash Player 10.1.

See Using the OSMF Player

Protecting content with Flash Access

To use deliver content with HTTP Dynamic Streaming and protect it with Flash Access, use the following tools:

- Flash Access Server for Protected Streaming

  This server is a Flash Access license server implementation optimized for use with HTTP Dynamic Streaming. See *Adobe Flash Access Protecting Content*.

  Note: *The Flash Access SDK and the Flash Access license server reference implementation can also issue licenses for HTTP Dynamic Streaming.*

- HTTP Dynamic Streaming File Packager

  Use this tool to package content offline for on-demand streaming.

The certificates that you use with the File Packager and the Live Packager must match the license server.

Important: *Do not use the Flash Access packaging tools to encrypt content. The HTTP Dynamic Streaming packagers cannot fragment content that has already been encrypted. Use the HTTP Dynamic Streaming packagers to both encrypt and fragment content.*

Workflow for using Flash Access with on-demand HTTP Dynamic Streaming

1 Use the File Packager to package and encrypt content. See "File Packager reference" on page 15.

2 Copy the packaged files to an Apache HTTP Server that uses the HTTP Dynamic Streaming Origin Module.

3 The OSMF Player does the following:

   a Requests the file fragments and plays them back seamlessly.

   b Obtains a content license from Flash Access Server for Protected Streaming. The content license contains the key required to play the content to legitimate clients.

   c Decrypts the content in real-time.

Tutorial

Tutorial: On-demand HTTP Dynamic Streaming

Requirements

To complete this tutorial, download the following software:

- OSMF Player for HTTP Dynamic Streaming
- Flash Player 10.1

On-demand HTTP Dynamic Streaming requires Apache 2.2. Apache 2.2 is included with Flash Media Server 3.5 and later. Download one of the following:

- Apache 2.2
- Flash Media Server 3.5
- Flash Media Server 4.0

On-demand HTTP Dynamic Streaming requires the following software tools. These tools are included with Flash Media Server 4.0. If you're using Apache 2.2 without Flash Media Server or with Flash Media Server 3.5, download the tools:

- File Packager and HTTP Origin Module

Follow these steps to create your first end-to-end, on-demand, multi-bitrate HTTP Dynamic Streaming application:

1 Do one of the following to install Apache:

   - Install Flash Media Server 4.0 or later.

     This version of the server includes Apache and the HTTP Origin Module. On Windows, it includes the Microsoft Visual C++ 2005 Redistributable Package and the Microsoft Visual C++ 2008 Redistributable Package. On Linux, it includes the Linux libraries.

   - Install Flash Media Server 3.5.

     This version of the server includes Apache. This version does not include the HTTP Dynamic Streaming Origin Module, the Visual C++ runtime packages, or the Linux libraries.

   - Install Apache HTTP Server version 2.2.

     This version of Apache doesn't include the HTTP Origin Module, the Visual C++ runtime packages, or the Linux libraries.

2 If you're using Apache with Flash Media Server, "Configure Apache and Flash Media Server to run on the same computer" on page 8.

3 If you're not using Flash Media Server 4.0, do the following:

   a On Windows, download and install Microsoft Visual C++ 2005 Redistributable Package and Microsoft Visual C++ 2008 Redistributable Package.

      On Linux, download and install the following libraries: OpenSSL, Expat, and Netscape Portable Runtime (NSPR).

   b Install the HTTP Origin Module.

      On Windows, copy the following files to the Apache \modules folder: mod_f4fhttp.so, adbe_F4V.dll, and libexpat.dll.

      On Linux, copy the following files to the Apache \modules folder: mod_f4fhttp.so, and libF4V.so file.

   c Create a "vod" folder in the Apache DocumentRoot folder. The vod folder will hold the packaged media files.

      On Windows, the default location is C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\vod.

      On Linux, the default location is /usr/local/apache/htdocs/vod.

   d To configure the HTTP Origin Module, do the following:

      e Open the Apache \conf\httpd.conf file in a text editor.

      f Locate the `LoadModule` section and add the following line:

```
LoadModule f4fhttp_module modules/mod_f4fhttp.so
```

         Note: *If there isn't a `LoadModule` section, add the line anywhere in the file.*

      g Locate the `<Location>` section and add the following:

```
<Location /vod>
    HttpStreamingEnabled true
    HttpStreamingContentPath "C:\Program Files\Apache Software
Foundation\Apache2.2\htdocs\vod"
</Location>
```

         Note: *Set the `HttpStreamingContentPath` directive to the full path of the "vod" folder you created in step 2c.*

      h Save and close the file.

      i Restart Apache HTTP Server.

4 To package files, complete the task "Package multi-bitrate files" on page 13 and return to this tutorial.

5 Copy the packaged files to the "vod" folder you created in step 3c.

   On Flash Media Server, the "vod" folder is in the following locations:

   - On Windows, C:\Program Files\Adobe\Flash Media Server 4.0\webroot\vod.
   - On Linux, /opt/adobe/fms/webroot/vod.

6 To play the media in OSMF Player, do the following:

   a Copy the OSMF Player files (including the \scripts, \images, and \assets folders and their files) to the DocumentRoot folder. This folder is the parent of the "vod" folder.

      Note: *If the DocumentRoot folder already contains an "images" folder, copy the files in the OSMF Player \images folder to the DocumentRoot\images folder.*

   b Open a web browser and enter the address of the OSMF Player.

      This tutorial uses **http://localhost/OSMFPlayer.html**.

      Note: *OSMF Player requires Flash Player 10.1.*

c Click the Eject button and enter **http://localhost/vod/sample1_1500kbps.f4m**.

d Click Play.

e To test multi-bitrate streaming, click the Q button to turn on manual switching. Click Q- and Q+ to shift down and up.

Understanding the on-demand tutorial

To play a video in OSMF Player, request an F4M file, also called a *manifest file*. The request syntax for on-demand streaming is:

```
http://<server-name>/<location-tag-alias>/<manifest-file-name>.f4m
```

Note: *The syntax is the same for all on-demand use cases.*

The following table explains the variables in the request syntax:

Variable	Description
<server-name>	The domain name or IP address of the computer hosting Flash Media Server. In this tutorial, OSMF Player and Flash Media Server are on the same computer, so the value is `localhost`.
<location-tag-alias>	This value is configured in a Location directive in the Apache /conf/httpd.conf file. The default value for on-demand http streaming in the Apache HTTP Server that installs with Flash Media Server is `vod`.
<manifest-file-name>	The name of the manifest (.f4m) file created by the File Packager.

More Help topics

"Configure on-demand streaming" on page 9

"Requesting on-demand files from a media player" on page 9

Chapter 2: Installing and configuring the HTTP Origin Module

The HTTP Origin Module is an extension to Apache HTTP Server that serves content packaged for HTTP Dynamic Streaming. The HTTP Origin Module is available from adobe.com/products/httpdynamicstreamingand is installed with Flash Media Server 4.0 and later.

About the HTTP Origin Module

HTTP Origin Module requirements

The HTTP Origin Module is an extension to Apache HTTP Server version 2.2. The HTTP Origin Module processes F4F, F4M, F4X, .bootstrap and .drmmeta files. For more information about these file types, see "Input file formats and output file formats" on page 11.

The HTTP Origin Module reference implementation available on adobe.com supports on-demand streaming only. You can install this HTTP Origin Module on any installation of Apache HTTP Server 2.2. This document contains information about on-demand streaming only.

Flash Media Server 4.0 includes Apache HTTP Server and the HTTP Origin Module and is configured for live and on-demand HTTP Dynamic Streaming. For more information, see Apache configurations for HTTP Dynamic Streaming.

Data flow through the HTTP Origin Module

To play a file in the OSMF Player, you provide only the URL to an F4M file. The OSMF Player automatically handles the transactions in steps 4-8.

1. The client sends an HTTP request for media to Apache, for example:

   `http://www.example.com/media/http_dynamic_Streaming.f4m`

2. Apache passes the request to the HTTP Origin Module.
3. The HTTP Origin Module returns the F4M file to the client.
4. The client receives the F4M file and uses the bootstrap information to translate time code into a segment#/fragment# pair.
5. The client sends a second HTTP request to Apache and asks for a specific fragment from the F4F file, for example:

   `http://www.example.com/media/http_dynamic_StreamingSeg1-Frag1`

6. Apache receives the request and passes the request to the HTTP Origin Module.
7. The HTTP Origin Module uses the index file (F4X) to translate the request into a byte offset in the F4F file. It sends the contents of this file that correspond to Segment1 and Fragment. This partial content is an F4F fragment, for example:

   `http://www.example.com/media/http_dynamic_StreamingSeg1.f4f`

8. The client receives the fragment and starts playback based on the bootstrap information provided with the fragment and the information from the F4M file.

Configure Apache and Flash Media Server to run on the same computer

The Flash Media Server installer includes Apache HTTP Server. By default, Flash Media Server receives all traffic on port 80 and proxies it to Apache on port 8134. Proxying the traffic can cause performance issues with HTTP Dynamic Streaming because of the high number of HTTP requests at both edge and origin network locations.

To fix this issue, configure Flash Media Server not to use port 80 and configure Apache to use port 80.

Configure Flash Media Server not to use port 80:

1 Open the *rootinstall*/conf/fms.ini file in a text editor.

2 Remove port 80 from the ADAPTOR.HOSTPORT parameter, as follows:

```
ADAPTOR.HOSTPORT = :1935
```

3 Remove the value from the HTTPPROXY.HOST parameter as follows:

```
HTTPPROXY.HOST =
```

4 Save and close the fms.ini file.

5 Restart the server. See Starting and stopping the server.

Configure Apache to use port 80:

1 Open the *rootinstall*/Apache2.2/conf/httpd.conf file in a text editor.

2 Locate the line `Listen 8134` and change it to the following:

```
Listen 80
```

3 Save and close the httpd.conf file.

4 Restart the server. See Starting and stopping the server.

Note: *Alternately, you can configure Apache to use a port other than port 80. To use a port other than 80, request URLs must contain the port number. By default, Apache on Flash Media Server is configured to use port 8134. This URL requests an on-demand HTTP Dynamic Streaming manifest file: http://www.example.com:8134/vod/mymedia.f4m.*

Installing and configuring the HTTP Origin Module

Prerequisites

1 Install Apache HTTP Server version 2.2 or later.

2 On Windows, download and install Microsoft Visual C++ 2005 Redistributable Package and Microsoft Visual C++ 2008 Redistributable Package.

On Linux, download and install the following libraries: OpenSSL, Expat, and Netscape Portable Runtime (NSPR).

Install the HTTP Origin Module

❖ Copy the following files to the Apache \modules folder:

- (Windows) mod_f4fhttp.so, adbe_F4V.dll, libexpat.dll
- (Linux) mod_f4fhttp.so, libF4V.so, libexpat.so

Configure on-demand streaming

1 Open the Apache \conf\httpd.conf file in a text editor.

2 Locate the LoadModule section and add the following line:

```
LoadModule f4fhttp_module modules/mod_f4fhttp.so
```

3 Configure the HTTP Origin Module to process HTTP requests that are targeted to a sandbox location. The module is not allowed to access files outside the sandbox.

Add the following:

```
<Location /vod>
    HttpStreamingEnabled true
    HttpStreamingContentPath "C:\Program Files\Apache Software
Foundation\Apache2.2\htdocs\vod"
</Location>
```

Substitute your own values for the values in italics.

Directive	Description
`<Location>`	The section of the request URL that maps to the physical location you specify in the `HttpStreamingContentPath` directive. Do not include an http prefix (such as "http://servername") in the `<Location>` path.
`HttpStreamingEnabled`	Indicates whether HTTP streaming is enabled (`true`) or not (`false`).
`HttpStreamingContentPath`	The physical location of the content. Use an absolute path that includes the DocumentRoot. In this example, the HTTP Origin Module can process files in the C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\vod folder or in any of its subfolders.

4 Save and close the file.

5 Restart Apache HTTP Server.

Note: *An httpd.conf file can contain multiple* `Location` *directives. You can configure a* `Location` *directive for on-demand media and another for live media.*

Requesting on-demand files from a media player

In the httpd.conf file `HttpStreamingContentPath` directive, configure a location for physical files on the server. To request a file, a video player uses the `Location` path + the additional path to the file.

Suppose the following is the `Location` configuration:

```
<Location /vod>
    HttpStreamingEnabled true
    HttpStreamingContentPath "C:\Program Files\Apache Software
Foundation\Apache2.2\htdocs\vod"
</Location>
```

This configuration processes any request that contains the path /vod. The HTTP Origin Module maps the request to the `HttpStreamingContentPath`, for example:

Request URL	Physical file served to client
http://example.com/vod/sample.f4m	C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\vod\sample.f4m
http://example.com/vod/free_content/demo.f4m	C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\vod\free_content\demo.f4m
http://example.com/index.html	Not processed.

More Help topics

"Tutorial: On-demand HTTP Dynamic Streaming" on page 4

Logging

Access log

All HTTP access requests are logged in the Apache access_log file in the \logs directory. The HTTP Origin Module doesn't add any logging directives. All access requests are logged through the standard Apache HTTP Server log module.

Error log

All errors are logged in the Apache error_log file in the \logs directory. For information about the response codes that the HTTP Origin Module returns, see Logging for HTTP Dynamic Streaming in the *Flash Media Server Configuration and Administration Guide*.

Chapter 3: Packaging on-demand media

The File Packager is a command-line tool that packages on-demand media files into a format that can stream over HTTP. The File Packager can also encrypt files for protection with Flash Access.

Packaging on-demand media

File Packager location

The File Packager is available from the following locations:

- adobe.com/products/httpdynamicstreaming
- Flash Media Server *rootinstall*/tools/f4fpackager folder

The File Packager requires several libraries to run. These libraries are located in the f4fpackager folder. To move the File Packager, move the f4fpackager folder and all its contents.

Input file formats and output file formats

To stream on-demand media over HTTP, process the file with the File Packager. The File Packager parses the content, translates it into segments and fragments, and writes the segments and fragments to the F4F format. You can also choose to have the File Packager encrypt the content for use with Flash Access.

The File Packager supports the following file formats:

- F4V/MP4 compatible files
- FLV

Important: *The file formats can contain any codec the format supports, except Speex audio. You can package a file that includes Speex audio, the File Packager does not throw an error, but the audio will not play.*

The File Packager outputs the following files:

File type	Description
.f4f	A segment. The tool outputs one or more F4F files. Each file contains a segment of the source file. Each segment contains one or more fragments of content. A player can use a URL to address each fragment.
.f4m	Flash Media Manifest file. Contains information about codec, resolution, and the availability of multi-bitrate files.
.f4x	Index file. Contains the location of specific fragments within a stream.
.bootstrap	Bootstrap file. Contains the bootstrap information for each segment of the file. The tool creates an external .bootstrap file only if you specify the `--external-bootstrap` option. If you don't specify that option, the bootstrap information is included in the manifest file.
.drmmeta	DRM Additional Header file. Contains additional header information about an encrypted file. The tool creates this file only if you specify the `--external-bootstrap` option and encrypt the file. Otherwise, the information is included in the manifest file.

Segments And Fragments

An F4f file contains one segment of a media file. Each segment can contain multiple fragments. Use the `--segment-duration` and `--fragment-duration` options to set the length of each segment and fragment, in seconds. A good size fragment is small enough to send over the internet and long enough to prevent multiple calls to Apache HTTP Server.

Run The File Packager

To package a file, pass the File Packager the name of the input file.

There are two ways to specify the name:

- Enter the `--input-file` option at the command line.
- Create a configuration file that specifies the input file and enter the `--conf-file` option at the command line.

More Help topics

"Command-line options to fragment files" on page 15

"Command-line options to encrypt files" on page 18

"File Packager configuration file" on page 19

Package content on Windows

1 Open a Command Window.

2 Change directories to the directory containing f4fpackager.exe.

3 Enter the name of the tool and any options:

```
f4fpackager --input-file=sample.f4v --output-path=c:\sampleoutput
```

The following files are output: sampleSeg1.f4f, sample.f4x, and sample.f4m.

More Help topics

"Command-line options to fragment files" on page 15

"Command-line options to encrypt files" on page 18

"File Packager configuration file" on page 19

Package content on Linux

1 Open a terminal window.

2 On a clean installation of Linux, set the `LD_LIBRARY_PATH` to the directory containing the File Packager libraries.

3 At the shell prompt, enter the name of the tool and any options:

```
f4fpackager --input-file=sample.f4v --output-path=/sampleoutput
```

The following files are output: sampleSeg1.f4f, sample.f4x, and sample.f4m.

More Help topics

"Command-line options to fragment files" on page 15

"Command-line options to encrypt files" on page 18

"File Packager configuration file" on page 19

Packaging multi-bitrate content

To stream multi-bitrate content, encode a piece of media at multiple bitrates, creating multiple files. Use the File Packager to process each media file individually. The media files share a manifest file that lists information about each media file. The OSMF Player uses this information for playback.

More Help topics

"Command-line options to fragment files" on page 15

"Command-line options to encrypt files" on page 18

"File Packager configuration file" on page 19

About packaging multi-bitrate files

Each time you run the File Packager, it creates a manifest file with the filename of the input file. When you pass the `--manifest-file` option, the File Packager adds the information from the manifest file you specify to the manifest file it creates. For example, suppose you run the following:

```
--input-file=sample1.f4v
```

The tool creates the files: sample1Seg1.f4f, sample1.f4x, and sample1.f4m.

Suppose you then run the following:

```
--input-file=sample2.f4v --manifest-file=sample1.f4m
```

The tool creates the files sample2Seg1.f4f, sample2.f4x, and sample2.f4m. The manifest file sample2.f4m contains information about sample2.f4v and sample1.f4v.

To create a manifest file for multi-bitrate content, pass the `--manifest-file` option and specify the manifest file of the previously processed file. Package the files from lowest bitrate to highest bitrate. Do not pass the `--manifest-file` option for the first file you process.

Package multi-bitrate files

Note: *This example uses Windows. For Linux commands, see "Package content on Linux" on page 12.*

1. Encode a media file at several bitrates, for example, 150 kbps, 700 kbps, and 1500 kbps.

   This example uses three files: sample1_150kbps.f4v, sample1_700kbps.f4v, and sample1_1500kbps.f4v. On Flash Media Server, these files are installed to *rootinstall*\applications\vod\media.

2. Copy the files to the same folder as the File Packager.

   On Flash Media Server 4, the File Packager is located in the *rootinstall*\tools\f4fpackager folder.

3. Open a Command Window.

4. Change directories to the directory containing f4fpackager.exe.

5. Enter the following:

```
f4fpackager --input-file=sample1_150kbps.f4v --bitrate=150
```

The following files are output: sample1_150kbpsSeg1.f4f, sample1_150kbps.f4x, and sample1_150kbps.f4m.

6 Enter the following to process the second file:

```
f4fpackager --input-file=sample1_700kbps.f4v --manifest-file=sample1_150kbps.f4m --
bitrate=700
```

The following files are output: sample1_700kbpsSeg1.f4f, sample1_700kbps.f4x, and sample1_700kbps.f4m. The manifest file sample1_700kbps.f4m includes information from the input file and information from the sample1_150kbps.f4m manifest file.

7 Enter the following to process the third file:

```
f4fpackager --input-file=sample1_1500kbps.f4v --manifest-file=sample1_700kbps.f4m --
bitrate=1500
```

The following files are output: sample1_1500kbpsSeg1.f4f, sample1_1500kbps.f4x, and sample1_1500kbps.f4m. The manifest file sample1_1500kbps.f4m includes information from the input file and information from the sample1_700kbps.f4m manifest file. The sample1_700kbps.f4m file contains information about the sample1_150kbps.f4v file and the sample1_700kbps.f4v file.

8 To play the file, pass the URL of the sample1_1500kbps.f4m file to OSMF Player. Because this manifest file was created last, it contains information about all three files.

Packaging and encrypting content

Use the File Packager to both package and encrypt content for use with Flash Access.

Flash Access includes a license server implementation based on the Flash Access SDK called Adobe® Flash® Access™ Server for Protected Streaming. Flash Access Server for Protected Streaming is designed for use with HTTP Dynamic Streaming. For more information about this server, see Chapter 8, "Using Flash Access Server for Protected Streaming" in the document *Adobe Flash Access Protecting Content*.

Generate a common key

To use the File Packager to encrypt content, you must generate a common key file. The common key file must contain 128 bytes of randomly generated data. The File Packager uses this key with the content ID to generate the content encryption key.

Note: *Although the base key is 128 bytes, only the first 128 bits (16 bytes) are used to generate the key.*

You can use the open source tool Open SSL to generate random data and output a common key file. Download Open SSL from www.openssl.org.

Generate random data and output a comomon key file:

```
openssl rand -out commonKey.bin 128
```

Pass commonKey.bin as the `--common-key` command line option or in the `<common-key>` element of a configuration file.

Use a configuration file to package and encrypt content

Many command-line options are required to encrypt a file. It's easier to use a configuration file to set options than it is to set options at the command line.

1 Open the file f4fpackager_config.xml in a text editor.

By default, the tool looks for f4fpackager_config.xml in the same directory as the File Packager. However, you can move it to any directory and give it any name.

2 Enter values for the following required options (sample values are included here):

```
<offline>
    <input-file>someFile.f4v</input-file>
    <content-id>contentId</content-id>
    <common-key>commonKey.bin</common-key>
    <license-server-url>http://server1.com:9999</license-server-url>
    <license-server-cert>licenseServer.der</license-server-cert>
    <transport-cert>transportCert.der</transport-cert>
    <packager-credential>packagerCredential.pfx</packager-credential>
    <credential-pwd>mYpwd</credential-pwd>
    <policy-file>policyFile.pol</policy-file>
</offline>
```

You can add additional elements to change their default values.

3 Save the configuration file.

4 Open a Command Window.

5 Change directories to the directory containing f4fpackager.exe.

6 Enter the following:

```
f4fpackager --conf-file=f4fpackager_config.xml
```

Use the command line to package and encrypt content

1 Open a Command Window.

2 Change directories to the directory containing the File Packager.

3 Enter the following:

```
f4fpackager --input-file=someFile.f4v --license-server-url=http://server1.com:9999 --
transport-cert=transportCert.der --license-sever-cert=licenseServer.der --packager-
credential=packagerCredential.pfx --credential-pwd=mYpwd --policy-file=policyFile.pol --
common-key=commonKey.bin --content-id=contentId
```

File Packager reference

Command-line options to fragment files

You can use the command line to set all the options for the File Packager. Options set at the command line override any options set in the configuration file. The only required option is `--input-file`.

Option	Description	Required	Default value
--input-file	The path to the source container file. The path can be absolute or relative to the directory containing the File Packager. If you specify the `--conf-file` option and the configuration file contains an <input-file> value, you do not need to use the `--input-file` option.	Yes	None
--output-path	The path to the directory to which the packaged files are output. If the directory doesn't exist, the tool creates it. The path can be absolute or relative to the directory containing the File Packager.	No	Current directory
--manifest-file	Updates an existing manifest file. Use this option to package a file encoded at multiple bitrates for dynamic streaming. For example, you could package three files (sample_500kpbs.f4v, sample_1000kbps.f4v, and sample_1500kbps.f4v) and add them all to the same manifest file. If you omit this option, the tool creates a Flash Media Manifest file (F4M).	No	None
--output-manifest-file	Creates a manifest file with the name provided.	No	None
--conf-file	The configuration file that contains settings for the packaging process. You can specify an absolute path or a path relative to the directory containing the tool. You can use the filename f4fpackager_config.xml or any other legal filename. Suppose the tool is in the c:\media directory and the configuration file is in the c:\media\test directory. Run the following from the command line: `f4fpackager --conf-file=test\f4fpackager_config.xml`.	No	f4fpackager_config.xml If the f4fpackager_config.xml file is not in the current directory, the default value is "none".
--bitrate	Specifies the bitrate of the content to fragment and adds it to the manifest file. Use this option to serve files encoded at multiple bitrates. Use this option or manually edit the manifest file to add the bitrate.	No	0
--segment-duration	The length, in seconds, of a segment. The default value is 0 which creates 1 segment.	No	0
--fragment-duration	The target length of each fragment, in seconds. The actual length of each fragment depends on properties of the input file, such as the keyframe interval.	No	4
--duration-precision	How precise the fragment durations are, in seconds. When using a time-based fragment duration, duration precision controls how much rounding is applied to the fragment durations in the fragment run tables. This prevents small variations in the fragment durations from causing the run tables to grow too large.	No	0.1

Option	Description	Required	Default value
--frame-rate	The frame rate of the original content, in frames per second (fps). The value is floating point; for NTSC, use the value 29.97.	No	
--frame-precision	This option is the frame-based equivalent to the --duration-precision option. This value is the number of frames to round to in the fragment run tables.	No	1
--frames-per-keyframe-interval	The number of frames between each keyframe. For example, 30 fps video with a keyframe every 2 seconds contains 60 frames per keyframe interval.	No	
--keyframe-intervals-per-fragment	The number of keyframe intervals per fragment. The default value is 1 which sets the fragment size to the same size as the keyframe interval.	No	1
--allowed-drift	As the tool rounds off durations, it drifts from the real time. This option specifies how far the tool can vary from the real time, in seconds.	No	1
--external-bootstrap	Outputs the bootstrap box to a .bootstrap file. If you omit this option, the information in the bootstrap box is placed in the manifest file (F4M). Outputs DRM additional header information to a .drmmeta file when a file is encrypted. If you omit this option, the additional header information is included in the manifest file.	No	None
--edit-list	If you specify this option, the File Packager follows the edit list in the input file. By default, the packager does not follow the edit list.	No	None
--config-dump	Dumps the configuration settings. This option does not fragment the file.	No	None
--sample-dump	Dumps the samples and the fragments they're set to. This option does not fragment the file.	No	None
--progress	Prints progress updates.	No	None
--help	Lists and describes every File Packager command.	No	None
--inspect-fragments	Inspects the F4F fragments in an F4F file.	No	None
--inspect-bootstrap	Inspects the bootstrap file. If a .bootstrap file is provided as an input, it inspects the file. If not, it inspects the bootstrap box in an F4F file.	No	None
--inspect-index	Inspects the AFRA index file if an AFRA file is provided.	No	None

Setting fragment duration based on frames or time

The File Packager has options that determine the size of fragments. There is a frame-based option, `--keyframe-interval-per-fragment`, and a time-based option, `--fragment-duration`. Each option has additional options that refine how the tool creates fragments.

Use the frame-based option when the source media contains video encoded at a constant frame rate. The frame-based option lets you carefully match the fragment size to the video's keyframe interval. Use the time-based option for media that contains audio or data but not video.

Frame-based options: `--keyframe-interval-per-fragment`, `--frames-per-keyframe-interval`, `--frame-rate`, `--frame-precision`.Time-based options: `--fragment-duration`, `--duration-precision`. The frame-based option overrides the time-based option. If you don't provide a `--keyframe-interval-per-fragment` option, the File Packager uses the `--fragment-duration` option. If you don't provide values for `--keyframe-interval-per-fragment` or `--fragment-duration`, the File Packager uses the default fragment duration, which is 4 seconds.

Command-line options to encrypt files

Use the File Packager to encrypt files for use with Flash Access 2.0. Some of the following options require digital certificates. To obtain digital certificates, see the Flash Access Certificate Enrollment Site.

Option	Description	Required	Default
--transport-cert	The DER encoded transport certificate file.	Yes	None
--license-server-url	The URL of the license server that handles license acquisition for this content.	Yes	None
--license-server-cert	The DER encoded license server certificate file used for content protection.	Yes	None
--packager-credential	The PFX file containing the packager's protection credentials.	Yes	None
--credential-pwd	The password string used to secure the packager credentials.	Yes	None
--common-key	A file containing the common key. The common key is used with the content ID to generate the content encryption key. The common key must be randomly generated and 128 bytes. Although the common key is 128 bytes, only the first 128 bits (16 bytes) are used to generate the key. For information about generating a common key file, see "Generate a common key" on page 14. For dynamic streaming, use the same common key and content ID for an entire set of content. Using the same key and id allows a single license to decrypt a set of content.	Yes	None
--content-id	The content ID used with the common key to generate the content encryption key.	Yes	None
--policy-file	The file containing the policy for this content. Currently only a single policy can be applied to content packaged with this tool.	Yes	None
--encrypt-audio	Specifies whether to encrypt the audio in the file.	No	true
--encrypt-video	Specifies whether to encrypt the video in the file.	No	true

Option	Description	Required	Default
--encrypt-data	Specifies whether to encrypt the data in the file. This option supports both FLV and F4V/MP4 formats.	No	true
--ms-unencrypted	The number of milliseconds at the beginning of the content that remains unencrypted. Use this option to improve stream start times because it allows the client to acquire the license asynchronously as playback begins.	No	0
--video-encrypt-level	The level of encryption for H.264 content. Possible values are 0 (low), 1(medium), and 2 (high). The default value is 2. **Note:** *Using this option on non-H.264 content won't generate an error, but it won't change the encryption level*. The values "0" and "1" mean "partial encryption"; only important samples like video keframes are encrypted. This setting creates fewer frames to decrypt. Use this setting to improve playback performance on the client. The value "2", encrypts all video samples (after --ms-unencrypted, if specified)..	No	2

File Packager configuration file

The File Packager uses an XML configuration file to read values for command-line options. The default configuration file, f4fpackager_config.xml, is located in the same directory as the File Packager. It is often easier to specify options in a text file that you can reuse than it is to enter options at the command line.

Options you set at the command line override options set in the configuration file. This feature lets you create a configuration file for a set of content but edit a few settings for individual files.

Note: *You do not need to use the configuration file to run the tool.*

Enter the `--conf-file` option at the command line to pass the name and location of a configuration file. If you don't enter the `--conf-file` option, the tool checks for a file called f4fpackager_config.xml file in the same directory as the tool.

You can enter all the options for packaging and encrypting, including the input file, in the configuration file.

The following is the format of the configuration file:

```
<offline>
    <input-file>/media/input-file.f4v</input-file>
    <output-path>/media/http</output-path>
    <fragment-duration></fragment-duration>
   ... You can set as many options as you want ...
</offline>
```

For options that take a path, you can specify absolute paths or paths relative to the directory containing the tool.

Errors

The File Packager outputs the following errors:

Error code	Error message	Description
1	File not found	The input file, the manifest file or the configuration file could not be found.
2	Input File not given	No input file has been specified. To run the File Packager, you must specify a path to a supported file type. See "Input file formats and output file formats" on page 11.
3	Invalid input file	The input file, manifest file, or configuration file is an invalid input file.
4	Can't access file	The input file, manifest file, or configuration file cannot be accessed.
5	Unsupported codec	The input file contains unsupported codecs.
6	No supported tracks	The input file doesn't contain any supported tracks.
7	Write error	A write error occurred. Verify that the input file is structured correctly and that there is enough memory to run the File Packager.
8	XML library error	An error occur ed when parsing the configuration file. Verify that the XML is structured correctly.
9	Program Options error	The program options library failed. Verify that the command line arguments are structured correctly.
10	Offline Encryption info error	The File Packager doesn't have enough information to encrypt the file. Verify that there aren't any missing inputs.
11	Encryption library error	The encryption library threw an error while encrypting the content.
12	Standard error	A Standard Exception has occur ed. Verify that there is enough memory to run the File Packager.
13	Invalid Bootstrap error	The bootstrap file is corrupt and cannot be parsed.
14	FLV Error	The FLV file cannot be read.
999	Internal error	There has been an internal error in the File Packager.
1000	Invalid Input file warning	Invalid input file warning.
2000	Invalid Error code	The error code is invalid.