MPEG DASH

MPEG DASH Adaptive Streaming

Specifications

MPEG DASH is MPEG's standardized Adaptive Streaming over HTTP. It is specified in the following international standards:

- ISO/IEC 23009-1 specifies the overall DASH architecture and the XML syntax for the MPD (Media Presentation Document)
- ISO/IEC 23001-7 specifies the Common Encryption for MP4 fragments
- ISO/IEC 14496-12/AMD 3 specifies the extensions to 14496-12 that are necessary to support DASH with fragmented MP4 media

Quick introduction to MPEG DASH

At a very high level, an MPEG DASH presentation consists of an initial XML manfifest, called the **Media Presentation Document** (MPD for short), which describes media segments that form a complete presentation. Along with a number of attributes, the MPD allows the MPEG DASH player to compute the URL of each segment, and to download it and render it.
The `mp4dash` tool takes care of creating the DASH MPD document, as well as packaging the media segment in their proper form and output location, including pre- and post- processing if necessary (like encryption for DRM support).

Using `mp4dash`

`mp4dash` is the name of the Bento4 tool that can convert one or more input media files into a complete MPEG DASH presentation. Running this tool requires that you have python (2.6 or above) installed on your system (python 3.x is not currently supported). Please visit www.python.org if you need to install python.
If you are using the Bento4 precompiled SDK distribution, you will find it convenient to use the `mp4dash` command, located in the `bin/` directory, which will automatically run the `mp4-dash.py` python script for you.
If you are running from a source distribution, you may need to invoke the python script (located under `Source/Python/utils`) directly, like this:
`python <path-to-bento4-python-utils>/mp4-dash.py <arguments>`
In the examples below, we assume that you are running from a standard precompiled distribution, thus we will simply use the `mp4dash` command.

The `mp4dash` tool needs to invoke the Bento4 command line binaries `mp4dump`, `mp4encrypt`, `mp4info`, and `mp4split`. If you are invoking the tool as distributed in the pre-compiled SDK, the binaries will automatically be found in the SDK package. If you are running the tool from the Source SDK, with binaries that you compiled from the source, so you'll need to use the `--exec-dir` option to specify the directory where your built binaries reside (platform dependent).

If you just run the `mp4dash` tool by itself, it will print out the options that are supported.
For a complete list of command line options for `mp4dash`, visit the mp4dash command line details page.

Preparing the input files

To create DASH MP4 content, you need to start with **fragmented MP4 files**. The input files you will be working with may or may not already be in fragmented MP4 form. Ideally, your encode will already produce MP4 files that way.
The command line tool `mp4info` can tell you if an MP4 file is fragmented or not: here's an example of what you will see in the `Movie:` part of the `mp4info` output for a non-fragmented MP4 file (the line `fragments: no`)

```
Movie:
    duration:   147188 ms
    time scale: 2997
    fragments:  no

Found 2 Tracks
```

If you have non-fragmented MP4 files, you can use the `mp4fragment` tool to fragment them.

Example

```
mp4fragment video.mp4 video-fragmented.mp4
```

Dealing with ISMV and ISMA input files

ISMV and ISMA files are basically fragmented MP4 files, and as such can be used as input to `mp4dash`. However, most legacy tools (encoder and packagers) that produce ISMV and ISMA files generate fragmented MP4 files that lack a `tfdt` timestamp box. While these files appear to be normal fragmented MP4 files, the lack of `tfdt` renders those media files unsuitable for playback with many DASH clients, including most HTML5 based clients.
If you have such files, you can *fix this problem* by refragmenting your ISMV and ISMA files into compliant fragmented MP4 files, using the `mp4fragment` tool.

Example

```
mp4fragment video.ismv video.mp4
```

This should print something like

```
NOTICE: file is already fragmented, it will be re-fragmented
found regular I-frame interval: 1 frames (at 25.000 frames per second)
```

You can then use the re-fragmented files as input to `mp4dash`.

Tip
You can tell whether your fragmented input file has a `tfdt` box or not by using `mp4dump`.
A file with `tfdt` will show entries like this:

```
[moof] size=8+816
  [mfhd] size=12+4
    sequence number = 490
  [traf] size=8+792
    [tfhd] size=12+4, flags=20000
      track ID = 2
    [tfdt] size=12+8, version=1
      base media decode time = 32281600
    [trun] size=12+744, flags=301
      sample count = 92
      data offset = 832
```

with a `tfdt` box as a child of the `traf` container box

Generating a DASH presentation

Once you have fragmented MP4 files to work with as input, you can generate a DASH presentation, including an MPD and media files or media segments. For single-bitrate streaming, a single MP4 file is required. For multi-bitrate streaming, you will need a set of MP4 files that have been encoded with closed GOPs (Group Of Pictures) with equal durations. Also, the audio tracks in all the files should be encoded with the same parameters. Once you have your input files, you can use the `mp4dash` tool to automatically generate the DASH MPD and optionally split the MP4 file into individual file segments.

Simple examples

Single MP4 input file

```
mp4dash video.mp4
```

Multi-bitrate set of MP4 files

```
mp4dash video_1000.mp4 video_2000.mp4 video_3000.mp4
```

Advanced usage

Multi-language and stream selection

By default, `mp4dash` will create an output that contains all the audio tracks found in its input(s). It is possible to limit the selected tracks by using input stream selectors. A stream selector is specified as a prefix before an input file path, enclosed between square brackets. The selector syntax is `[<property>=<value>]`, where `<property>` can be `type`, `track`, or `language`. When the property name is `type`, the value can be `audio` to select only the audio track(s) from the input, or `video` for the video track. When the property name is `track`, the value is the ID of the track to select (could be audio or video). When the property name is `language` the value is an audio language code (2 or 3 letter code as defined by the DASH specification – IETF RFC 5646). It is possible to combine the `language` selector with a `type` selector, by separating the two by a comma (ex: `[type=audio,language=fr]`. If the `language` selector is used by itself without a `type` selector, the video track as well as the matching audio track will be selected.

Examples:
Select the French audio track and the video track from video.mp4:

```
mp4dash [language=fr]video.mp4
```

Select the French audio track from video1.mp4 and the video track from video2.mp4:

```
mp4dash [type=audio,language=fr]video1.mp4 [type=video]video2.mp4
```

Select track ID 3 from video1.mp4 and all tracks from video2.mp4:

```
mp4dash [track=3]video1.mp4 video2.mp4
```

Select the track with 'undefined' language from video1.mp4 and remap the language to 'fr'

```
mp4dash --language-map=und:fr [language=und]video1.mp4
```

Using the `--verbose` option may be useful when debugging language selection options

Encryption and DRM

MPEG DASH streams can be encrypted, and played on clients that have a DRM-enabled DASH player.
Visit the MPEG DASH Encryption & DRM page for details.

Serving DASH Streams

By default, `mp4dash` produces an output that can be served by any regular HTTP server. In order to make this possible, each one of the media segments and the init segments are stored in separate files. The MPD then refers to relative URLs for those files. Care should be taken to configure the HTTP server to return the correct `Content-Type` for the MPD document and segment files. The MPD document (named `stream.mpd` by default by the `mp4dash` tool) should be served with `Content-Type: application/dash+xml` and the init and media segments should be served with a `Content-Type: video/mp4`

Serving DASH without splitting segments

With special support from the server, it is possible to serve DASH streams without splitting the segments into individual files. This is achieved by having a server-side module that can virtualize URLs, so as to be able to expose each segment as an separate URL, mapping it back to a byte range in a source file when requests are made. MPEG does not specify a common standard for such URL virtualization. Each server that supports this model is free do do it any way it chooses. For example, the Microsoft IIS server can be used with a IIS Media Services extension for Smooth Streaming to deliver MPEG DASH streams. More details can be found on the Serving MPEG DASH with Microsoft IIS page. Another useful way to do the same thing is to use the open-source Hippo Media Server package, which knows how to virtualize URLs for MPEG DASH and Smooth Streaming.
The `--hippo` option of `mp4dash` is used to generate the server-side manifest needed by the Hippo Media Server.

2 COMMENTS ON “MPEG DASH”

Pingback: How to encode Multi-bitrate videos in MPEG-DASH for MSE based media players (2/2) Streamroot Blog

Pingback: How to encode multi-bitrate videos in MPEG-DASH for MSE based media players | TDNGAN

Comments are closed.

© 2016 Axiomatic Systems LLC