Akamai® Media Services Live

Encoder Compatibility Testing and Qualification

May 5, 2015

Akamai Technologies, Inc.

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

Luna Control Center™, 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® Media Services Live
Encoder Compatibility Testing and Qualification

Contents

Preface

Welcome to Akamai® *Media Services Live: Encoder Compatibility Testing and Qualification*.

To facilitate the streaming of high-quality live events in Media Services Live, you must use third-party live encoders that are compatible with Media Services Live for the Adobe® Flash® platform, Apple® iPhone® and iPad®, or Microsoft® Silverlight®.

This document describes the process and resources the company has compiled to facilitate the testing of these encoders on its platform to ensure live events in Media Services Live have the quality and reliability you and your viewers expect. It also describes the qualification process in which selected NetAlliance partners can participate to provide visibility and value for their products by working with Media Services Live on an ongoing basis to ensure both parties' solutions evolve together and continue to provide the highest quality solution for you.

This document comprises the following chapters:

- Chapter 1. Live Encoders Compatibility Testing and Qualification explains benefits and process of compatibility testing and qualification.

  The rest of the chapters provide details for specific platforms:
- Chapter 2. Media Services Live for the RTMP Platform: Live Encoder Guidelines
- Chapter 3. Media Services Live for HLS Ingest: Live Encoder Guidelines
- Chapter 4. Media Services Live for Smooth Ingest: Live Encoder Guidelines
- Chapter 5. Media Services Live for HDS Ingest: Live Encoder Guidelines
- Chapter 6. Media Services Live for DASH: Live Encoder Guidelines

New Material in This Document

This version of the document provides the following:

05/05/2015:

Updated the recommendations for backup URLs following information for qualified encoders and the latest software versions in Section Playback Workflow in Chapter 3. Media Services Live for HLS Ingest: Live Encoder Guidelines .

04/15/2015 (v15.3):

Updated the following information for qualified encoders and the latest software versions in Section Qualified Live Encoders' List in Chapter 1. Live Encoders Compatibility Testing and Qualification :

- Updated the version of Wowza Media Server to v4.1.2 (for RTMP, HDS, HLS)
- Updated the version of Haivision Kulabyte Live encoder to v4.3 (for RTMP)
- Added TV1.EU LT, v1.0 (for RTMP)

03/05/2015 (v15.2):

Added a summary of all qualified encoders and the latest software versions — refer to Section Qualified Live Encoders' List in Chapter 1. Live Encoders Compatibility Testing and Qualification .

01/15/2015 (v15.1):

Clarified that 10-second length is maximum value supported for HLS Ingest in Chapter 3. Media Services Live for HLS Ingest: Live Encoder Guidelines .

12/12/2014:

- Added detailed information on publishing and playback for HLS primary and backup flows. Refer to Primary and Backup Workflows.
- Added a new recommendation for RTMP (HLS output) — Usage of B-Frames.

09/24/2014 (v14.6):

Changed all labels throughout the document to reflect product naming change to Media Services Live.

07/15/2014 (v14.4):

- Added a requirement for Flash — Authentication. Refer to Chapter 2. Media Services Live for the RTMP Platform: Live Encoder Guidelines .
- Fixed primary and backup hostnames (removed **hds**) in Primary and Backup Workflows for HDS.
- Clarified information on Upload Order for HLS.
- Changed all labels to reflect new product naming throughout the document: Media Services Live.

05/08/2014 (v14.2):

- Added requirements for HDS — refer to Chapter 5. Media Services Live for HDS Ingest: Live Encoder Guidelines .
- Added requirements for DASH — refer to Chapter 6. Media Services Live for DASH: Live Encoder Guidelines .

01/15/2014 (v9.7):

- Changed the **onMetaData** requirement for Adobe Flash Platform:

  The onMetaData object information must be sent in the first packet and consistently throughout the life of the live stream.

- Added the **Packet Timestamps** requirement for Adobe Flash Platform:

  Media Services Live: Stream Packaging streaming requires that FLV timestamp data from the encoder to be in increasing order regardless of types (audio/video/metadata/etc.).

10/30/2013 (v9.5):

Updated **Recommendations for Stream Lag / Jump Forward Solution** for Adobe Flash Platform.

08/26 /2013 (v9.3):

- Added new **Recommendations for Stream Lag / Jump Forward Solution** for Adobe Flash Platform.
- Fixed URL examples in iPhone/iPad—Live encoder guidelines. Refer to Section **Upload URL Contents**.

04/04/2013:

- New navigation system—all active links are highlighted in **Blue**.

Chapter 1. Live Encoders Compatibility Testing and Qualification

In This Chapter

Benefits of Compatibility Testing and Qualification

Due to the fact that Media Services Live supports multiple input and output types with a great deal of flexibility and robustness, it is highly beneficial to test any live stream encoder before it is used for a live event. This ensures live events are successful and provides third-party vendors an opportunity to validate their solution in a non-production environment. This testing process has enabled Media Services Live to have a long track record of high-quality live events.

Compatibility Testing

Basic compatibility testing is a very lightweight process that can quickly determine if a stream encoder is capable of powering a live event in Media Services Live. It is not a robust test process and may not cover many of the functionality requirements such as the ability to recover from types of various interruptions. This should be considered the bare minimum level of testing an encoder receives before it is used for a live event. To ensure high-quality, uninterrupted live events, we suggest using encoders that have gone through the more robust testing of our qualification process.

Qualification

Qualification has several benefits in addition to the higher level of testing involved that enables more robust live events. Live encoding vendors that receive a qualified rating must also be part of NetAlliance partner program. This enables both parties to benefit from an ongoing relationship, both business and technical, allowing them to evolve together and address potential issues quickly and efficiently. Ultimately this provides a better experience for mutual customers. As a partner, we will work with the vendor to ensure new features and functionality in Media Services Live and in the vendor's product continue to work together by providing ongoing technical updates and testing capabilities in Media Services Live. This cooperation will allow us to recommend qualified encoders to interested customers and prospects, as they have been shown to provide a reliable solution.

Limitations of Using Non-Qualified Encoders

While too numerous to list here, some of the limitations that might be experienced when using a non-qualified encoder include difficult technical escalations and issue resolution, unsupported security mechanisms, and incompatibilities with various functionality such as 24/7 streams.

Using a qualified encoder in Media Services Live has many benefits that lead to having a less stressful event setup and a more successful live event. This is why it is highly recommended using one of the many encoders that have been through the qualification process. For a list of encoders that are qualified for Media Services Live, refer to the user guide for the streaming protocol you plan on using:

- *Media Services—Live: Stream Packaging User Guide*
- *Media Services—Live: HLS Ingest User Guide*
- *Media Services—Live: Smooth Ingest User Guide*

Process for Compatibility Testing

1. Third party performs technical review of encoder guidelines.
2. Third party returns completed guidelines' checklist per protocol.
3. Test Entry Points and playback URLs are provided.
4. Third party publishes functioning stream on provided Entry Point.
5. We analyze streams and provides approval or feedback.

Third parties interested in having their encoder solution tested in Media Services Live must have their technical personnel review the encoder guidelines for each protocol they are going to test. These guidelines describe the requirements for encoders in Media Services Live, recommendations for encoders on the network, and the benefits or limitations that can be expected for each item discussed. For details, refer to the following Appendices:

- Akamai HD for the Adobe Flash Platform—Live Encoder Guidelines
- Akamai HD for iPhone and iPad—Live Encoder Guidelines
- Akamai HD for Microsoft Silverlight—Live Encoder Guidelines

Upon review of these guidelines, if a solution meets the requirements described in the therein, the third party must complete the associated checklist for each protocol they are testing and forward it to **encoder_qual_requests@akamai.com**. They must be certain to include the following:

- Appropriate business and technical contacts from their organization,
- A list of our customers and prospects interested in using their solution,
- Any personnel they are working with so they can be kept up-to-date,

- The checklist for each protocol they are testing.

Once we have reviewed the information, they will contact the party with instructions describing how to receive access to our system. Access will depend on the situation and the protocol(s) to be tested.

Once the third party has received access to our various networks they must publish a stream to the provided Entry Point that plays back successfully. At this point, we analyze the incoming stream and provides an evaluation. If the encoder is deemed compatible with Media Services Live, then the encoder can be used to stream live events. If the stream is unsuccessful, we are able to provide feedback to assist with resolving the problem.

Process for Qualification

1. Begin by contacting Partner team at **partner_qualifications@akamai.com**
2. Conference call introducing Partner Program and Qualification Process
3. Technical review of encoder guidelines by vendor
4. Completion and returning of checklist, per protocol
5. Testing:
    a. In-Lab Testing: Live encoder shipped to our Lab w/SDI input and instructions
    b. Remote Testing: Test Entry Points provided.
6. We analyzes encoder and streams over time, then provides feedback and/or approval

Third parties interested in becoming a qualified encoder must start by contacting our NetAlliance Partner team at **partner_qualifications@akamai.com** with their business information. A call will be arranged to discuss the NetAlliance program, the protocols supported by their solution and the process that will be followed to thoroughly test their solution in Media Services —Live to provide it a qualified rating. They should begin by reviewing the current guidelines for encoders to be compatible with Media Services Live (see the appendices). Reviewing these guidelines and returning their associated checklists should be completed as early as possible.

Testing is performed by our engineering department's encoder lab. To support testing, the third party should arrange a loaner encoder, and we would make arrangements, space permitting, to set up their solution in a lab. Testing in our lab environment provides the most flexibility in determining compatibility, flexibility, and reliability. Remote testing may be possible in certain circumstances, but it presents additional logistical challenges and cannot adequately test fault recovery in the same way we can in the lab. Arrangements for remote testing will be handled on a case-by-case basis.

Once qualified, both parties will ensure the following items are in place over the life of the partnership:

- Formal business and technical contacts
- Business and technical escalation paths
- Technical documentation and resources for our CCare group
- Mutual updates provided for new versions of hardware and software
- Cooperation to ensure that new versions will continue to be compatible with Media Services Live

Qualified partners will be highlighted to our field teams and suggested to our customers and prospects as recommended solutions to enable their live events in Media Services Live. Partners can also inform their customers and prospects that they have been qualified for use in Media Services Live.

Qualified Live Encoders' List

To ensure a successful and high quality Live streaming delivery, use a 3rd party live encoder qualified with Media Services Live for the Adobe® Flash® platform, Apple® iPhone® and iPad®, and Microsoft® Silverlight®. The 3rd party encoders are thoroughly tested against Media Services Live to ensure the hardware, software, and future updates meet compatibility guidelines.

The following tables list live encoders and the latest software versions that are qualified to use with Media Services Live

For previously qualified live encoders' software versions, contact your Account Representative.

Company	Encoder	RTMP Ingest	HLS Ingest	HDS Ingest	DASH Ingest	Smooth Ingest
Bright Cove	**Zencoder**	1.0	Not supported	Not supported	Not supported	Not supported
Digital Rapids	**StreamLE**	3.7.1	3.7.1	Not supported	Not supported	3.7.1
Elemental	**Live**	2.6.2	2.6.2	2.6.2	2.6.2	2.6.2
Envivio	**Halo**	4.00	4.00	Not supported	Not supported	4.00
Envivio	**Muse Live** (formerly 4Caster)	3.21	3.2.1	Not supported	Not supported	3.21
Haivision	**Kulabyte Live**	4.3	4.1	Not supported	Not supported	Not supported
Harmonic	**ProMedia**	1.4.0.0	1.4.0.0	Not supported	Not supported	1.4.0.0

Company	Encoder	RTMP Ingest	HLS Ingest	HDS Ingest	DASH Ingest	Smooth Ingest
iStreamPlanet	**Aventus**	2.4	2.4	2.3.23800.1	Not supported	Not supported
Media Excel	**Hero**	3.42	3.42	Not supported	Not supported	3.42
Real Network	**Helix**	3.42	3.42	Not supported	Not supported	3.42
Viewcast	**Niagara**	7.0.283.0	7.0.283.0	Not supported	Not supported	Not supported
Wowza	**Media Server**	4.1.2	4.1.2d	4.1.2	Not supported	Not supported

Table 1: Partners' Live Encoders

Company	Encoder	RTMP Ingest	HLS Ingest	HDS Ingest	DASH Ingest	Smooth Ingest
Adobe	**Prime Live Packager**	Not supported	1.3.2	Not supported	Not supported	Not supported
Anevia	**ViaMotion Plus**	Not supported	3.5.4	Not supported	Not supported	3.5.4
Anvato	**Live**	2.1	2.1	Not supported	Not supported	Not supported
AllegroDVT	**AL2000**	spruce.4.23	spruce.4.23	Not supported	Not supported	spruce.4.23
Ateme	**Titan**	3.0	Not supported	Not supported	Not supported	3.0
Cires21	**C21 Live**	2.91	Not supported	Not supported	Not supported	Not supported
Cisco (formerly Inlet Spinnaker)	**AS Series Media Processor**	8.5.1	8.5.1	Not supported	Not supported	8.5.1
G&L	Custom	1.0	Not supported	Not supported	Not supported	Not supported
Igolgi	**BEO**	Not supported	Not supported	Not supported	Not supported	2.1
Orban	**Opticodec-PC**	1010	Not supported	Not supported	Not supported	Not supported
Polycom		1.0	Not supported	Not supported	Not supported	Not supported
Prixmio		0.5	Not supported	Not supported	Not supported	Not supported

Company	Encoder	RTMP Ingest	HLS Ingest	HDS Ingest	DASH Ingest	Smooth Ingest
Telestream	**Wirecast**	5.0.3	Not supported	Not supported	Not supported	Not supported
Telos	**ProSTREAM x.2**	1.03	Not supported	Not supported	Not supported	Not supported
Telos	**ProSTREAM**	2.6.3beta	Not supported	Not supported	Not supported	Not supported
Telos	**Omnia A.XE**	1.73	Not supported	Not supported	Not supported	Not supported
TV1.EU	**LT**	1.0	Not supported	Not supported	Not supported	Not supported
Thomson Video Networks	**ViBE VS7000**	3.01	3.01	Not supported	Not supported	3.01
VBrick	**VBrick9000**	3.2.0a	Not supported	Not supported	Not supported	Not supported

Table 2: Qualified Live Encoders

Chapter 2. Media Services Live for the RTMP Platform: Live Encoder Guidelines

In This Chapter

This chapter explains the required and recommended specifications for live stream encoders that are compatible with Media Services Live for the RTMP Platform. Compatible encoders should be capable of successfully streaming a live event, but meeting these guidelines does not guarantee performance of any encoder in Media Services Live.

Requirements

The following items **are required** for an encoder to function in Media Services Live: Stream Packaging (RTMP Ingest). (It is not recommended to use any encoder that does not meet these requirements):

- Compliant with RTMP, H.264, and VP6
- onMetaData
- Authentication
- Packet Timestamps
- Single Stream per TCP Connection
- Sync Across Renditions
- End of Stream Sent Before Connection Close
- Resolve Hostnames and Reconnect
- Failover to Alternate Host Upon Error/Failure
- Entrypoint and Error Recovery

Additionally, the following items are **recommended** for encoders in Media Services Live: Stream Packaging (RTMP Ingest):

- Stream Lag / Jump Forward Solution

- Usage of B-Frames

Compliant with RTMP, H.264, and VP6

The encoder must be in compliance with the RTMP, H.264, and VP6 specifications.

onMetaData

The onMetaData object information must be sent in the first packet and consistently throughout the life of the live stream.

This object must include the following properties:

- videocodecid
- audiocodecid
- audioonly
- videoonly
- stereo
- audiocodecid
- videocodecid
- width
- height
- videodatarate
- audiodatarate
- audiosamplerate
- audiosamplesize
- audiochannels
- framerate
- avcprofile
- avclevel
- aacaot

Authentication

The authentication is required for RTMP, either Akamai or Adobe authentication.

Note: When using Adobe authentication in a multi-TCP connection, you need to sequence your multi bitrate TCP connection in series and not in parallel.

Packet Timestamps

Media Services Live for Stream Packaging requires that FLV timestamp data from the encoder to be in increasing order regardless of types (audio/video/metadata/etc.).

Note: Keep in mind that Adobe's RTMP specification only requires that timestamps be increasing within a type. Timestamps can be out of order across different types. On contrary, we require timestamps monotonically increasing across types.

Single Stream per TCP Connection

The encoder must be capable of producing each stream on its own TCP connection. This will optimize performance and allow for additional performance and control during congested periods.

If an encoder does not use separate TCP connections for each stream, it will likely experience packet loss, delays, and synchronization issues during an event.

Sync Across Renditions

The encoder must provide some mechanism to sync across renditions to support MBR (Multiple Bitrate) streaming. Encoding solutions that use multiple encoders for MBR sets must provide a solution to sync renditions across each encoder.

End of Stream Sent Before Connection Close

The encoder must ensure the appropriate end-of-stream signal is sent before a TCP connection is closed.

Encoders that fail to properly send the end of stream will orphan the stream, leaving it in an unusable state. Future attempts to use the same stream name and ID will be unsuccessful.

Resolve Hostnames and Reconnect

The encoder must be capable of performing DNS lookups to resolve hostnames and properly reconnect to an Entry Point in the event that the connection is interrupted. While reconnecting the encoder should re-resolve host names to ensure up-to-date DNS information is used when reconnecting.

If an encoder is unable to perform a DNS lookup, it will be unable to start any encoding sessions. If it is unable to properly reconnect, it will be unable recover from an outage.

Failover to Alternate Host Upon Error/Failure

The encoder must be capable of using an alternative, or backup, host should an Entry Point become unavailable and re-resolving the hostnames provides a new IP address. This will enable a stream to continue in event of an error and the hostname is updated to a new Entry Point. Under normal conditions the encoder should not need to re-resolve the hostname until the TTL (Time To Live) has expired for that hostname.

In the event of an error, if an encoder is unable to connect to an alternate host, the stream will likely be unable to recover.

Entrypoint and Error Recovery

In the event that an Entry Point is restarted, an encoder must be capable of recovering. This requires the encoder to restart the stream, sending the appropriate onMetaData packets as if it was a completely new stream.

If an encoder is unable to recover from this type of event, the stream will become orphaned and unplayable.

Recommendations

Stream Lag / Jump Forward Solution

Encoder should detect that current RTMP packets are lagging behind or jumping forward in the timestamp. This condition might happen due to many reasons, such as:

- Network connectivity issue between an encoder and our Entrypoint/Ingest;
- Exhaustion of CPU and memory resources on the encoder;
- Encoder bugs.

If the stream lag/jump forward is greater than configurable number of GOP sizes (default should be 2 GOP sizes), an encoder should take the following actions if it detects such condition:

- TCP disconnect and reconnect all renditions/bitrates in that adaptive bitrate session;
- Drop all data from the time the timestamp lag/jump forward has been detected till the current time when connection has been re-established;
- Restart stream at GOP boundary;
- Resend all stream initialization information on a reconnect like onMetadata, if video is H.264 encoded then video sequence headers, if audio is AAC-encoded then audio sequence headers.

If this feature is not available on the encoder there is a setting per stream on the our Ingest system that can be enabled to do the above from the Ingest system, meaning, the Ingest will detect the lag and disconnect all renditions assuming the encoder will reconnect all of them. The lag amount to disconnect the stream is configurable on the Ingest.

This has been seen to work well for transient network loss/latency issues. If the degradation is ongoing for a longer period then this would result in frequent disconnect and reconnect.

Usage of B-Frames

It is not recommended to use B-frame. Introduction of B-frame will cause diminishing quality of HLS playback. For more details, contact your Account Representative.

Encoder Qualification Compatibility Checklist

The purpose of this checklist is to understand which live stream encoders meet the requirements and recommendations of Media Services Live for the Adobe Flash Platform (Stream Packaging). After completing this checklist please return it to **encoder_qual_requests@akamai.com** to assist testing.

```
-- General Information --

Contact Name:
Contact Company:
Contact Email:
Contact Phone:
Encoder Manufacter:
Encoder Model:
Encoder Software Version:
```

```
-- Requirements --

Compliance with RTMP, H.264, and VP6
  - Is the encoder designed in compliance with the RTMP, H.264 and VP6
      specifications?

onMetaData
  - Does the encoder send the appropriate onMetaData information in the
      first packet of every stream start?

Single Stream per TCP Connection
  - Does the encoder use a single TCP connection per stream?

Sync Across Renditions
  - Does the encoder provide some mechanism to sync renditions across a
      multiple bitrate set?

End of Stream Sent Before Connection Close
  - Does the encoder send an "end of stream" before closing connections?

Resolve Hostnames and Reconnect
  - Can the encoder resolve hostnames upon the initial connection and
      reconnects?

Failover to Alternate Host Upon Error/Failure
  - Can the encoder use an alternative host should an Entry Point become
      unavailable?

Entry Point & Error Recovery
  - Can the encoder properly recover when an entrypoint restarts?


-- Support Functionality Questions --

Support for Multiple Streams on the Same Encoder
  - Does your encoder support the ability to generate multiple time
      synced streams, aka MBR, from a single instance of your encoder?

Support for Multiple Streams Across Multiple Encoders
  - Does your encoder support the ability to generate multiple time
      synced streams, aka MBR, using multiple instances of your encoder?

Support for Audio-Only Streams
  - Does your encoder support the ability to produce streams that only
      contain an audio track?

Support for Video-Only Streams
  - Does your encoder support the ability to produce streams that only
      contain an video track?
```

Chapter 3. Media Services Live for HLS Ingest: Live Encoder Guidelines

In This Chapter

This chapter explains the recommended and required specifications for a live stream encoder that is compatible with Media Services Live for HLS Ingest. Compatible encoders should be capable of successfully streaming a live event, but meeting these guidelines does not guarantee performance of any encoder on our network.

Recommended items will provide end-users a better experience. The limitations of not having those items are explained where appropriate.

Note: Media Services Live for HLS Ingest does not honor live encoders’ ***HTTP DELETE*** *requests. While our servers will return a 200 HTTP status code, the content is not actually deleted as requested.*

Requirements

The following items **are required** for an encoder to function in Media Services Live (It is not recommended you use any encoder that does not meet these requirements):

- Compliance with Apple® HTTP Live Streaming
- URL Configurability
- PUT/POST Request
- Segment Numbering
- Segment Directory Rollover
- Session ID
- PUT/POST Request
- Upload Retries and Timeouts
- Upload Order
- DNS lookups Upon Error/Failure Conditions
- TCP Upload Optimization
- Codecs

- Properly Marking Playlists Complete
- Key Frame Alignment

Additionally, the following items are **recommended** for encoders in Media Services Live and described in the these sections:

- Primary and Backup Workflows
- DNS Lookups, Ongoing
- Unique Session ID
- Video Encoding
- Audio Encoding
- Audio Only
- Entire Event Playlist/Index Upload
- Alternate Hostname Uploads
- Segment Configurability
- Segment Numbering After Restart
- Logging and Reporting
- Parallel Uploads
- Chunk Encoding on Uploads
- Security
- Encoder Authentication

Compliance with Apple® HTTP Live Streaming

Encoders must meet the recommendations set by Apple Inc. in the HTTP Live Streaming overview (**http://developer.apple.com/library/ios/#documentation/networkinginternet/conceptual/streamingmediaguide/Introduction/Introduction.html**).

URL Configurability

The encoder must allow specification of a POST server address also known as a POST URL, upload URL, or ingress URL.

The encoder must allow specification of a playback address also known as a playback URL or retrieval link.

PUT/POST Request

The encoder must be capable of uploading generated content, video segments, and playlist/manifest files using multiple PUT/POST requests. Each segment should be uploaded in its own request. It is preferable to use a persistent connection for these requests.

Segment Numbering

- Segments generated by the encoder must be created with monotonically increasing segment numbers such as 1, 2, 3, etc.
- That segment number must be extractable from the upload URL using regex.

Segment Directory Rollover

To support events longer than an hour, an encoder must be capable of using multiple directories to upload segments. It must be configured to rollover to a new directory at a set interval to ensure that approximately one (1) hour' worth of segments are uploaded per directory. For example, if the encoder is configured to use the maximum supported 10-second segments, then your directory rollover will consist of 2000 segments. This will cause the rollover to occur 5.5 hours.

The directory names can be anything you choose; one example may be to append a number after the directory name that increases with each rollover. It is preferred that you restart the numbering of segments within a directory at zero (0), but this is optional.

Session ID

The encoder must not introduce any other changing parameters (For example, frequently changing time stamps) within the URL path. For example, if the URL path format were:

<eventname>/<sessionID>/<bitrateID>/<directory_rollover#>/<timestamp>_<sequence#>.ts

introducing the **<timestamp>** parameter creates another set of directories for each **.ts** file, a condition that would cause excess directories and possible timeouts over long periods of time.

Ideally, the path would appear as follows:

<eventname>/<sessionID>/<bitrateID>/<directory_rollover#>/<somename>_<sequence#>.ts

In this case, only the **<directory_rollover#>** and **<sequence#>** parameters change, which is the desired behavior.

URL Formatting

The encoder must be capable of using a POST URL in the following form:

- **http://example-i.akamaihd.net/**[*stream_id*]/[*event_name*]/[*file-name_.m3u8_or_.ts*]

The playback URL takes the form:

- **http://example-i.akamaihd.net/hls/live/**[*stream_id*]/[*event_name*]/[*file-name_.m3u8_or_.ts*]

Upload Retries and Timeouts

The encoder must be capable of retrying all upload requests that encounter recoverable server or network errors using the same POST URL. On an initial failure, the encoder should attempt at least one retry. If new segments are prepared for upload, the encoder may then attempt to upload those, but there will be a jump in playback for any missing segments.

Non-recoverable server or network errors, such as authorization failures (403) or path errors (404), should not be retried.

For connection timeouts, the encoder should determine when to close the connection based on the length of segments being uploaded.

Upload Order

In accordance with Apple's recommendation, the encoder must upload all required files for a specific bitrate and segment before proceeding to the next segment. For example, for a bitrate that has segments, a playlist that updates every segment, and key files, the encoder should upload the segment file followed by a key file (optional) and the playlist file in serial fashion. The encoder must not upload segment (N+1) until segment N has been uploaded. The order of operation should be:

1. Upload .ts
2. Upload .m3u8

If there is a problem with either Steps 1 or 2, retry them. Do not proceed to Step 2 until Step 1 succeeds.

DNS lookups Upon Error/Failure Conditions

The encoder must perform DNS lookups any time an error occurs. Any time there is an upload error, server abort, connection timeout, or HTTP timeout, the encoder should perform a DNS lookup to ensure it continues to be connected with the appropriate server.

TCP Upload Optimization

To maximize upload performance, a TCP window size of 64k is recommended. The encoder must be able to specify an alternate size according to network conditions.

Codecs

The only supported codecs for playback on iOS devices are H.264 for video and AAC for audio. The encoder must be capable of encoding using these formats.

For video encoding, the encoder must be capable of encoding using the H.264 Baseline profile to accommodate all iOS devices.

Properly Marking Playlists Complete

At the end of a list stream the encoder must mark the stream as complete by inserting the appropriate tag into the .m3u8 playlist file. This will ensure the device behaves appropriately when it comes to the end of the stream. This tag, **EXT-X-ENDLIST**, will prevent unexpected playback issues when the device comes to the end of the stream. While this is not required for playback, it is highly recommended to ensure viewers have a consistent and high-quality playback experience.

Key Frame Alignment

The stream encoder must produce key frame alignment across bitrates per the targeted segment duration. For example, if the encoder is generating maximum supported 10-second segments, then every 10 seconds there must be timestamp-aligned key frames across streams. The streams' frame rate could otherwise deviate and the become out of sync.

Recommendations

The following items are recommended for encoders in Media Services Live. Encoders that cannot provide these will have limited functionality. These limitations are described in the following sections:

- Primary and Backup Workflows
- DNS Lookups, Ongoing
- Unique Session ID
- Video Encoding
- Audio Encoding
- Audio Only
- Entire Event Playlist/Index Upload
- Alternate Hostname Uploads
- Segment Configurability
- Segment Numbering After Restart
- Logging and Reporting
- Parallel Uploads

- Chunk Encoding on Uploads
- Security
- Encoder Authentication

Primary and Backup Workflows

To avoid interruptions in the distributed system, the encoder should be capable of publishing the same content to primary and backup locations.

Encoder should be able to input primary and backup publishing and playback host-names and primary and backup paths for a given stream. The same content should be simultaneously published to primary and backup paths.

Publishing Workflow

The encoder will publish primary and backup streams with following URLs:

- *Primary Publishing URL*:

  http://post.example-i.akamaihd.net/[*stream_id*]**/**[*event_name*]

  Example:

  http://post.testconfig-i.akamaihd.net/500002/test79

- *Backup Publishing URL*:

  http://postb.example-i.akamaihd.net/[*stream_id*]**-b/**[*event_name*]

  Example:

  http://postb.testconfig-i.akamaihd.net/500002-b/test79

While optional, using the **post** element in your ingress hostname is recommended, as it will ensure your encoder's broadcast is given priority over other traffic types, thus, reducing upload errors and retry attempts. Stream availability becomes more reliable as well.

Note: The stream ID must be identical between your primary and backup streams. The stream ID can be obtained from the Luna Control Center's ***Manage HD Live Streams*** *and* ***Stream Details*** *pages.*

Playback Workflow

The encoder should generate the primary and backup URLs as following:

- *Primary Playback URL:*

  http://example-i.akamaihd.net/hls/live/[*stream_id*]**/**[*event_name*]**/** [*filename*]**.*m3u8***

  Example:

  http://testconfig-i.akamaihd.net/hls/live/500002/test79/test79.m3u8

- *Backup Playback URL:*

  http://example-i.akamaihd.net/hls/live/[*stream_id*]-b**/**[*event_name*]**/** [*filename*]**.*m3u8***

  Example:

  http://testconfig-i.akamaihd.net/hls/live/500002-b/test79/test79.m3u8

  The above playback information will be embedded in the variant m3u8 file. Backup URLs are highlighted in Green in the following example:

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=648224,RESOLUTION=640x360,CODECS="avc1.4d401e,mp4a.40.2"
http://example-i.akamaihd.net/hls/live/10002/test/01.m3u8
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=648224,RESOLUTION=640x360,CODECS="avc1.4d401e,mp4a.40.2"
http://example-i.akamaihd.net/hls/live/10002-b/test/01.m3u8
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=443680,RESOLUTION=400x224,CODECS="avc1.42e00d,mp4a.40.2"
http://example-i.akamaihd.net/hls/live/10002/test/02.m3u8
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=443680,RESOLUTION=400x224,CODECS="avc1.42e00d,mp4a.40.2"
http://example-i.akamaihd.net/hls/live/10002-b/test/02.m3u8
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=956544,RESOLUTION=640x720,CODECS="avc1.4d401f,mp4a.40.2"
http://example-i.akamaihd.nett/hls/live/10002/test/03.m3u8
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=956544,RESOLUTION=640x720,CODECS="avc1.4d401f,mp4a.40.2"
http://example-i.akamaihd.net/hls/live/10002-b/test/03.m3u8
```

DNS Lookups, Ongoing

The encoder ideally should perform ongoing DNS lookups while streaming.

Be aware that encoders are periodically asked to re-resolve DNS. For long running streams, old Entry Points may get overloaded or go down for maintenance or bad connectivity. In this case, when the IP address changes, we recommend the encoder complete its current POST/PUT request, close the connection, and then reopen a connection to the new IP Address to move encoders to the best available Entry Point for them at that time. To ensure the most optimum throughput and to guarantee high stream availability, we also recommend the introduction of periodic DNS resolution be short (Our DNS TTLs for these records are generally 20 seconds).

Note: In the near future, encoding vendors will be required to support this recommendation in order to receive and maintain a qualified rating. We are working with our partners to implement this functionality, as it improves stream availability and ensures better throughput. We suggest you make every effort to add this recommendation to your solution at your earliest convenience.

Unique Session ID

Encoders that do not provide a unique session ID in their POST URLs may experience cases in which their archived data becomes corrupted when the encoder restarts and attempts to reuse the same sequence numbers and paths.

Note: In the near future, encoding vendors will be required to support this recommendation in order to receive and maintain a qualified rating. We are working with our partners to implement this functionality, as it improves stream availability and ensures better throughput. We suggest that you make every effort to add this recommendation to your solution at your earliest convenience.

Video Encoding

Stream encoders are required to be able to encode using the H.264 Baseline profile to ensure proper playback on iOS devices. It is recommended that encoders be capable of encoding using the H.264 Main profile to provide higher quality on high bitrate streams for compatible devices such as the iPhone 4 and iPad.

Encoders that cannot encode using the main profile of H.264 will produce lower quality renditions for high-end renditions on the iPhone 4 and iPad even when using the exact same bitrate and frame size. This should not cause any technical problems beyond the difference in visual quality between profiles.

Audio Encoding

Encoders are required to be capable of encoding a 64 kbps audio-only stream to meet the requirements Apple mandates for video content delivered in iOS applications.

Customers wishing to provide a seamless audio experience should use encoders capable of encoding the audio track of all renditions to 64 kbps. This will ensure audible clips or other distortions are not experienced when the device switches from one rendition to another.

Customers wishing to provide higher quality audio should use encoders capable of encoding audio tracks at higher bitrates than the 64 kbps required for the lowest rendition. By encoding at higher bitrates, the user should experience higher fidelity audio but may experience audible clips or other distortions when the device switches from one rendition to another. Customers should ensure that any quality issues experienced when switching between the required 64 kbps audio-only stream and their higher quality streams are acceptable before initiating a live event.

Audio Only

For audio encoding, the encoder must be capable of encoding a 64 kbps audio-only stream to meet Apple's requirements for applications submitted to the Apple App Store™ online store.

Entire Event Playlist/Index Upload

When possible the encoder should be able to upload a single .m3u8 playlist file that includes a list of all .ts segments generated after the event completes. This is helpful for video on demand playback after the live event.

Events broadcast using encoders that are not capable of uploading a complete playlist file upon event completion may only have a limited playback window available after playback based on the contents of the final playlist uploaded during the event.

Alternate Hostname Uploads

The encoder should be capable of using multiple hostnames when publishing multiple bitrates to better handle failure situations.

Encoders that cannot use multiple hostnames will have a single point of failure. In the event that the single upload host being used is lost, the event will be interrupted until the host returns or a new host is provisioned.

Segment Configurability

The encoder should be capable of configuring the number of segments per playlist, the number of segments per folder, and the duration of each segment. While not required, this level of flexibility is helpful for situations that require changing these attributes.

Encoders that cannot configure these attributes will not be flexible enough to satisfy specific requirements of certain situations. One common reason to change the duration is to lower the amount of hand-wave latency or lower the number of segment files being created for caching or hardware performance reasons.

Segment Numbering After Restart

Encoders should not reuse segment numbers. In the event that the encoder restarts or recovers from an outage, it should create segments starting at the next available sequence for monotonically increasing sequences (001.ts, 002.ts, 003.ts, etc.) or at a timestamp that has not yet been used. Alternatively, encoders can use a unique session ID that changes upon restart so reusing the original segment numbers would not produce overlap.

Reuse of file names is undesirable for live streams. If an encoder is incapable of ensuring a segment number is not reused, end users may be served old content during an event.

Logging and Reporting

To enable better troubleshooting of issues, it is recommended that the encoder logs errors and reports them. In the event that an encoder experiences an issue, it is far more likely to be resolved and avoided in the future if logs are available for review.

Parallel Uploads

To enable better throughput it is recommended that the encoder upload multiple bitrates in parallel using multiple sockets.

Encoders that cannot use multiple sockets to upload in parallel will likely experience poor performance as a result of bandwidth contention.

Chunk Encoding on Uploads

It is recommended that the encoder use HTTP chunked encoding to upload files more efficiently.

Encoders that cannot use chunked encoding may experience additional hand waving latency.

Security

It is recommended that the encoder encrypt segments using the AES-128 encryption algorithm in compliance with the Apple HTTP live streaming protocol.

For encrypted content with rotating keys, the key name should have a monotonically increasing integer component that can be reset if the event or directory component changes.

Encoders that are unable to provide the security described above should not be used for content that must be protected. Unsecured content delivered on the Apple HTTP live streaming protocol can easily recompiled and distributed.

Encoder Authentication

Stream encoders should be capable of maintaining their state through cookies, headers, or query string parameters and allow for the ability to identify different uploads. The encoder should maintain its state between restarts and resets or should have the ability to determine sequence numbers that have been used already. A simple way to do this is to attempt to retrieve the .m3u8 file for each bitrate in the stream when the encoder starts. If none exist (i.e., a 404 is served), choose the sequence number however you desire. If one does exist, then determine the last sequence number used, and start at the next available one.

Encoders that are not capable of maintaining state as described above may provide end users incorrect content as segments are delivered in the wrong order or mistakenly redelivered at inappropriate points in an event.

Encoder Qualification Compatibility Checklist

The purpose of this checklist is to understand which live stream encoders meet the requirements and recommendations of Media Services Live for HLS Ingest. After completing this checklist please return it to **encoder_qual_requests@akamai.com** to assist testing.

```
-- General Information --

Contact Name:
Contact Company:
Contact Email:
Contact Phone:
Encoder Manufacter:
Exact Encoder Model:
Exact Encoder Software Version:
-- Requirements --

URL Configurability
    - Does the encoder allow the user to specify POST server address and
      playback address?

URL Formatting
    - Can the encoder use POST and playback URLs in the following
      formats?
        POST - http://post.example-f.akamaihd.net/[stream_id]/
      [event_name]/[filename_.m3u8_or_.ts]
        Playback - http://example-f.akamaihd.net/hls/live/[stream_id]/
      [event_name]/[filename_.m3u8_or_.ts]

PUT/POST Request
    - Can the encoder use multiple PUT/POST requests to upload content
      on a persistence connection?

Segment Numbering
    - Can the encoder generate segments with monotonically increasing
      segment numbers, such as 1, 2, 3, etc.?
    - Can that number be extracted from the upload URL using regular
      expressions?

Segment Directory Rollover
    - Can the encoder use multiple directories to ensure no single
      directory has more then 2000 files in it?
    - At what number of segments will the encoder rollover to a new
      directory? If configurable, what is the default?

Upload Retries and Timeouts
    - Can the encoder retry all upload requests that fail from
      recoverable server or network errors?
    - Will it prevent nonrecoverable server or network errors from being
      retried?
    - How is retry logic implemented? How many retries are attempted
      before giving up and moving to the next event?
```

Upload Order

- Can the encoder upload files in the following order: .ts segment, optional key files, .m3u8 file, delete unused .ts files?Re-resolve Hostnames
- Can the encoder re-resolve hostnames upon errors and failure conditions?

TCP Upload Optimization

- Can the encoder use a TCP window size of 64k and adjust according to network conditions?

Codecs

- Does the encoder use the H.264 Baseline codec for video and AAC for audio?

Audio-Only

- Can the encoder generate an audio-only rendition at 64kbps to meet "cellular fallback" requirements of Apple HLS protocol?

Properly mark playlists complete

- Does encoder add the "EXT-X-ENDLIST" tag to playlists upon event completion to signify the event has ended and prevent inconsistent playback experiences?
- Does the encoder add the "EXT-X-ENDLIST" tag to playlists anytime the encoder is graceully turned off?

-- Recommendations --

Video Encoding

- Can the encoder use H.264 Main profile on the higher bitrate streams?

Audio Encoding

- Can the encoder use 64kbps audio across all bitrates to ensure seamless bitrate switching?

Backup Streams

- Can the encoder use primary and backup streams with different names?

Entire Event Playlist Upload

- Can the encoder upload an entire playlist containing all segments for a specific event?

Alternate Hostname Uploads

- Can the encoder use multiple hostnames for different bitrates?

Segment Configurability

- Does the encoder enable configuration or the number of segments per playlist, the number of segments per folder, and the duration of each segment?

Segment Numbering After Restart

- Does the encoder have the ability to ensure segment numbers are not reused upon encoder restart? This is often accomplished by using a unique session ID in addition to the segment ID number.

```
Logging and Reporting
    - Does the encoder log and report errors for
      troubleshooting?Parallel Uploads
    - Can the encoder upload multiple bitrates in parallel using
      multiple sockets?

Chucked Encoding on Uploads
    - Can the encoder use HTTP chunked encoding?

Security
    - Does the device support AES-128 encryption according to Apple HLS
      protocol?
    - Can the encoder use rotating keys with a monotonically increasing
      integer component?

Encoding Authentication
    - Can the encoder maintain its state using cookies, headers, and
      query string parameters?

-- Support Functionality Questions --

Support for Multiple Streams on the Same Encoder
    - Does your encoder support the ability to generate multiple time
      synced streams, aka MBR, from a single instance of your encoder?

Support for Multiple Streams Across Multiple Encoders
    - Does your encoder support the ability to generate multiple time
      synced streams, aka MBR, using multiple instances of your encoder?

Support for Video Only Streams
    - Does your encoder support the ability to produce streams that only
      contain a video track?
```

Chapter 4. Media Services Live for Smooth Ingest: Live Encoder Guidelines

In This Chapter

This chapter explains the recommended and required specifications for a live stream encoder that is compatible with Media Services Live: Smooth Ingest. Compatible encoders should be capable of successfully streaming a live event, but meeting these guidelines does not guarantee performance of any encoder in Media Services Live.

Requirements

The following items **are required** for an encoder to function in Media Services Live (It is not recommended to use any encoder that does not meet these requirements):

- Compliant with Microsoft Live Smooth Streaming
- Single Stream per TCP Connection
- Sync Across Renditions
- End of Stream Sent Before Connection Close
- Resolve Hostnames and Reconnect
- Failover to Alternate Host Upon Error/Failure
- Recover When IISEP Restarts
- HTTP Digest Authentication

Compliant with Microsoft Live Smooth Streaming

The encoder must be in compliance with the Microsoft Live Smooth Streaming encoder specification (http://www.iis.net/community/files/media/smoothspecs/%5BMS-SMTH%5D.pdf).

Single Stream per TCP Connection

The encoder must be capable of producing each stream on its own TCP connection. This will optimize performance and allow for additional performance and control during congested periods.

If a stream encoder does not use separate TCP connections for each stream, it will likely experience packet loss, delays, and synchronization issues during an event.

Sync Across Renditions

The encoder must provide some mechanism to sync across renditions to support MBR (Multiple Bitrate) Streaming. Encoding solutions that use multiple encoders for MBR sets must provide a solution to sync renditions across each encoder.

End of Stream Sent Before Connection Close

The encoder must ensure the appropriate end-of-stream signal is sent before a TCP connection is closed.

Encoders that fail to properly send the end of stream will orphan the stream, leaving it in an unusable state. Future attempts to use the same stream name and ID will be unsuccessful.

Resolve Hostnames and Reconnect

The encoder must be capable of performing DNS lookups to resolve hostnames and properly reconnect to an Entry Point in the event that the connection is interrupted. While reconnecting the encoder should re-resolve hostnames to ensure up-to-date DNS information is used when reconnecting.

If a stream encoder is unable to perform a DNS lookup, it will be unable to start any encoding sessions. If it is unable to properly reconnect, it will be unable recover from an outage.

Failover to Alternate Host Upon Error/Failure

The encoder must be capable of using an alternative, or backup, host should an Entry Point become unavailable and re-resolving the hostnames provides a new IP address. This will enable a stream to continue in event of an error and the hostname is updated to a new Entry Point. Under normal conditions the encoder should not need to re-resolve the hostname until the time to live has expired for that hostname.

In the event of an error, if an encoder is unable to connect to an alternate host, the stream will likely be unable to recover.

Recover When IISEP Restarts

The encoder must be capable of detecting and recovering when an Entry Point restarts. This requires the encoder to appropriately restart the stream ensuring to send the moof box or moov boxes with the correct content.

If an encoder is unable to recover from this type of event, the stream will become orphaned and unplayable.

HTTP Digest Authentication

The encoder must be capable of using Digest Access authentication. Digest authentication uses encryption to send the password over the Internet, which is safer than basic access authentication that sends it in plain text.

Encoder Qualification Compatibility Checklist

The purpose of this checklist is to understand which live stream encoders meet the requirements and recommendations of Media Services Live for Smooth Ingest. After completing this checklist please return it to **encoder_qual_requests@akamai.com** to assist testing.

```
-- General Information --

Contact Name:
Contact Company:
Contact Email:
Contact Phone:
Encoder Manufacter:
Encoder Model:
Encoder Software Version:


-- Requirements --

Compliance with Microsoft Live Smooth Streaming
    - Is the encoder designed in compliance with the Microsoft Live
      Smooth Streaming specifications?
```

```
Single Stream per TCP Connection
    - Does the encoder use a single TCP connection per stream?

Sync Across Renditions
    - Does the encoder provide some mechanism to sync renditions across
      a multiple bitrate set?

End of Stream Sent Before Connection Close
    - Does the encoder send an "end of stream" before closing
      connections?

Resolve Hostnames and Reconnect
    - Can the encoder resolve hostnames upon the initial connection and
      reconnects?

Failover to Alternate Host upon Error/Failure
    - Can the encoder use an alternative host should an Entry Point
      become unavailable?

Recover When IISEP Restarts
    - Can the encoder properly recover when an Entry Point restarts?

HTTP Digest Authentication
    - Does the encoder support HTTP Digest authentication?


-- Support Functionality Questions --

Support for Multiple Streams on the Same Encoder
    - Does your encoder support the ability to generate multiple time
      synced streams, aka MBR, from a single instance of your encoder?

Support for Multiple Streams Across Multiple Encoders
    - Does your encoder support the ability to generate multiple time
      synced streams, aka MBR, using multiple instances of your encoder?

Support for Audio-Only Streams
    - Does your encoder support the ability to produce streams that only
      contain an audio track?

Support for Video-Only Streams
    - Does your encoder support the ability to produce streams that only
      contain an video track?
```

Chapter 5. Media Services Live for HDS Ingest: Live Encoder Guidelines

In This Chapter

This chapter explains the requirements and recommendations for a live stream encoder that is compatible with Media Services Live: HDS Ingest. Compatible encoders should be capable of successfully streaming a live event, but meeting these guidelines does not guarantee performance of any encoder on Media Services Live.

Recommended items will provide end-users a better experience. The limitations of not having those items are explained where appropriate.

Note: Media Services Live: HDS Ingest does not honor live encoders' ***HTTP DELETE*** *requests. While Media Services' servers will return a 200 HTTP status code, the content is not honored.*

Requirements

The following items **are required** for an encoder to function on Media Services Live (It is not recommended you use any encoder that does not meet these requirements):

- Compliance with Adobe® HTTP Dynamic Streaming (HDS)
- URL Configurability
- PUT/POST Request
- Segment and Fragment Numbering
- PUT/POST Request
- Upload Retries and Timeouts
- Upload Order
- TCP Upload Optimization
- Key Frame Alignment
- Extensions and Mime Types
- Encoder Identification

Additionally, the following items are **recommended** for encoders on Media Services Live and described in the these sections:

- Primary and Backup Workflows
- DNS Lookups, Ongoing
- Alternate Hostname Uploads
- Segment Configurability
- Segment Numbering After Restart
- Logging and Reporting
- Parallel Uploads
- Chunk Encoding on Uploads

Compliance with Adobe® HTTP Dynamic Streaming (HDS)

Encoders should meet recommendations set by Adobe HTTP Dynamic Streaming Specification. The F4M follows recommendations mentioned in http://osmf.org/dev/osmf/specpdfs/FlashMediaManifestFileFormatSpecification.pdf

URL Configurability

The encoder must allow specification of a POST server address also known as a POST URL, upload URL, or ingress URL.

The encoder must allow specification of a playback address also known as a playback URL or retrieval link.

PUT/POST Request

The encoder must be capable of uploading media fragments, bootstrap, drm metadata, and manifest file using POST/PUT requests. Each media fragments, bootstrap, drm metadata, and manifest file should be uploaded in its own HTTP request.

Segment and Fragment Numbering

- Fragments generated by encoder must be indexed with monotonically increasing fragment numbers within a segment.

  For example:

Seg1-Frag1	Seg1-Frag4
Seg1-Frag2	Seg1-Frag5
Seg1-Frag3	Seg1-Frag6

- All fragments must be part of one segment. The segment number should not change during a streaming session. This can be achieved by setting segment duration to a very large value, for example, 1 year in seconds. Fragment duration of 6 seconds is only supported for the current version of the product.
- Fragment numbers must not repeat when an encoder restarts. This can be achieved when the fragment numbers are derived from current epoch time. This results in always increasing fragment numbers and prevents overwriting data in the storage system.

URL Formatting

The encoder must be capable of posting URL in the formats described in the following sections:

- Set-Level Manifest
- Drm Metadata
- Upload Retries and Timeouts
- Bootstrap File
- Stream-Level Manifest

Set-Level Manifest

The Set-Level Manifest file must uploaded only once for each stream/event. The Set-Level Manifest name must end with key words **setlevelmanifest** or **master**. For example, a set level manifest name could be **testsetlevelmanifest.f4m** or **examplemaster.f4m**, or simply **master.f4m** or **setlevelmanifest.f4m**.

The URL format for the Set-Level Manifest is:

http://<hostname>/<format>/<streamID>/<eventname>/<anyname>{setlevelmanifest or master}.f4m

Examples:

http://post.50002.testconfig.r.akamaientrypoint.net/hds/50002/test67/test67setlevelmanifest.f4m

http://post.50002.testconfig.r.akamaientrypoint.net/hds/50002/test67/test67master.f4m

http://post.50002.testconfig.r.akamaientrypoint.net/hds/50002/test67/master.f4m

http://post.50002.testconfig.r.akamaientrypoint.net/hds/50002/test67/setlevelmanifest.f4m

Drm Metadata

The drm metadata must be uploaded once for each representation/bitrate and stored in a permanent storage location.

The URL format for the drm metadata is as follows:

http://<hostname>/<format>/<streamID>/<eventname>/<representationID>.drm

Example:

http://post.50002.testconfig.r.akamaientrypoint.net/hds/50002/test79/5000kbps.drm

Media Fragment File

The Media Fragment file must be uploaded every 6 seconds for fragment duration of 6 seconds for each bitrate/representation.

The URL format of the Media Fragment file is as follows:

http://<hostname>/<format>/<streamID>/<eventname>/<representationID>Seg1-Frag<number>

Example:

http://post.50002.testconfig.r.entrypoint.net/hds/50002/test79/5000kbpsSeg1-Frag34

Bootstrap File

The bootstrap file must be uploaded every 6 seconds for fragment duration of 6 seconds for each bitrate/representation. This file must be an external file, not embedded one.

The URL format for the bootstrap file is as follows:

http://<hostname>/<format>/<streamID>/<eventname>/<representationID>.bootstrap

Example:

http://post.50002.testconfig.r.akamaientrypoint.net/hds/50002/test79/5000kbps.bootstrap

Stream-Level Manifest

The Stream-Level Manifest must be uploaded every 6 seconds (the same as fragment duration of 6 seconds) for each bitrate/representation.

The URL format for the Stream-Level Manifest is:

http://<hostname>/<format>/<streamID>/<eventname>/<representationID>.f4m

Example:

http://post.50002.testconfig.r.akamaientrypoint.net/hds/50002/test79/5000kbps.f4m

Upload Retries and Timeouts

The following table summarizes how the failure conditions are handled.

Error Condition	Object Type	Handling
-TCP connection attempt timeout -Abort midstream -TCP send or receive timeout	Manifest	Re-resolve DNS on each retry and retry indefinitely every second.
-TCP connection attempt timeout -Abort midstream -Response timeout -TCP send or receive timeout	Data (media, timed text)	Re-resolve DNS on each retry and attempt 3 retries every 500ms. After 3 retry attempts, drop current data and continue with next segment data.
HTTP 403, 400	Any	Do not retry - Display Error Reason
HTTP 500	Data (media, timed text)	Re-resolve DNS on each retry and attempt 3 retries every 500 millisecond. After 3-retry attempts, drop current data and continue with next segment data.
HTTP 500	Manifest	Re-resolve DNS on each retry and retry indefinitely every 1 second.

Upload Order

The upload order for HDS must be as follows:

1. Set-Level Manifest — uploaded once at the beginning of each stream/event.
2. drm metadata file if using PHDS or Adobe Access — uploaded once for each representation/bitrate. For details, refer to

   http://www.adobe.com/devnet/adobe-media-server/articles/content-protection-using-phds-phls.html
3. Media Fragment File — every 6 seconds for fragment duration of 6 seconds.
4. Bootstrap file — uploaded every 6 seconds for fragment duration of 6 seconds.
5. Stream-Level Manifest — uploaded every 6s (the same as fragment duration) or more frequently if captioning or other media is involved.

TCP Upload Optimization

To maximize upload performance, a TCP window size of 64k is recommended. The encoder must be able to specify an alternate size according to network conditions.

Key Frame Alignment

- Encoder must align timestamps across representations/bit-rates.
- Similarly GOP size/key-frame interval across representations/bit-rates must be same.
- GOP size and in turn HDS fragment size must be 6s.
- The recommended GOP size is the same as fragment size for Adobe HDS content.

Extensions and Mime Types

The following table provides mime type requirements for different support extensions.

Extension	Mime Type
.f4m	application/f4m
Content (fragment) files and .bootstrap file	video/f4f

Encoder Identification

Encoder must be capable to send a User-Agent header that provides information about the encoder brand name, version number, and build number in a readable format.

Recommendations

The following items are recommended for encoders on Media Services Live. Encoders that cannot provide these will have limited functionality. These limitations are described in the following sections:

- Primary and Backup Workflows
- DNS Lookups, Ongoing
- Alternate Hostname Uploads
- Segment Configurability
- Segment Numbering After Restart
- Logging and Reporting
- Parallel Uploads
- Chunk Encoding on Uploads

Primary and Backup Workflows

To avoid interruptions in the distributed system, the encoder should be capable of publishing the same content to primary and backup locations.

Encoder should be able to input primary and backup publishing and playback host-names and primary and backup paths for a given stream. The same content (set-level manifest, stream-level manifest, drm meta files, fragments, and bootstrap) should be simultaneously published to primary and backup paths.

Publishing Workflow

The encoder will publish primary and backup streams with following URLs:

- *Primary Publishing hostname*:

  http://post.{streamID}.{customer}.r.akamaientrypoint.net/

- *Backup Publishing hostname*:

  http://postb.{streamID}.{customer}.r.akamaientrypoint.net/

- *Primary Publishing path*:

  hds/{streamID}/{eventname}/{manifest}.{object}

  Examples:

  http://post.500002.testconfig.r.akamaientrypoint.net/hds/500002/test79/test79.f4m

  http://post.500002.testconfig.r.akamaientrypoint.net/hds/500002/test79/5000kbpsSeg100Frag10

- *Backup Publishing path*:

  hds/{streamID}-b/{eventname}/{manifest}.{object}

  Examples:

  http://postb.500002.testconfig.r.akamaientrypoint.net/hds/500002-b/test79/test79.f4m

  http://postb.500002.testconfig.r.akamaientrypoint.net/hds/500002-b/test79/5000kbpsSeg100Frag10

Note the following:

- The encoder will publish the stream to both locations — primary and backup.
- The segments and .f4m will be duplicated across primary and backup and should be identical.

Playback Workflow

The encoder should allow primary and backup paths to be inputed in following way:

- *Base Playback URL:*

  http://{customer}.-i.akamaihd.net/

- *Primary Playback path:*

  hds/live/{streamID}/{eventname}/{object}

  Examples:

  http://testconfig-i.akamaihd.net/hds/live/500002/test79/test79.f4m

  http://testconfig-i.akamaihd.net/hds/live/500002/test79/5000kbpsSeg100Frag10

- *Backup Playback path:*

  hds/live/{streamID}-b/{eventname}/{object}

  Examples:

  http://testconfig-i.akamaihd.net/hds/live/500002-b/test79/test79.f4m

  http://testconfig-i.akamaihd.net/hds/live/500002-b/test79/5000kbpsSeg100Frag10

The above playback information input from a user will be taken and embedded in the set-level manifest as 2 separate **<adaptiveSet>**s. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns="http://ns.adobe.com/f4m/1.0" version="3.0">
<id>myVideo</id>
<streamType>live</streamType>
<baseURL>http://testconfig-i.akamaihd.net/hds/live/</baseURL>
<adaptiveSet>
<media href="500002/test79/5000kbps.f4m" bitrate="5000"/>
</adaptiveSet>
<adaptiveSet>
<media href="500002-b/test79/5000kbps.f4m" bitrate="5000"/>
</adaptiveSet>
</manifest>
```

The HDS player will request the stream-level manifest using above URLs. The player will then parse information in the stream-level manifest and request object names by appending to the base URLs in stream-level manifests. The player will use HTTP error responses from the server to use the backup stream content:

- **http://testconfig-i.akamaihd.net/hds/live/500002-b/test79/5000kbpsSeg24Frag41**

  or

- **http://testconfig-i.akamaihd.net/hds/live/500002/test79/5000kbps.drmmeta**

DNS Lookups, Ongoing

The encoder ideally should perform ongoing DNS lookups while streaming.

Be aware that encoders are periodically asked to re-resolve DNS. For long running streams, old Entry Points might get overloaded or go down for maintenance or bad connectivity. In this case, when the IP address changes, it is recommended that the encoder complete its current POST/PUT request, close the connection, and then reopen a connection to the new IP Address to move encoders to the best available Entry Point for them at that time. To ensure the most optimum throughput and to guarantee high stream availability, is it also recommended that the introduction of periodic DNS resolution be short (DNS TTLs for these records are generally 20 seconds).

Note: In the near future, encoding vendors will be required to support this recommendation in order to receive and maintain a qualified rating. We is working with our partners to implement this functionality, as it improves stream availability and ensures better throughput. Make every effort to add this recommendation to your solution at your earliest convenience.

Alternate Hostname Uploads

The encoder should be capable of using multiple hostnames when publishing multiple bitrates to better handle failure situations.

Encoders that cannot use multiple hostnames will have a single point of failure. In the event when the single upload host being used is lost, the event will be interrupted until the host returns, or a new host is provisioned.

Segment Configurability

Encoders should be able to configure the following:

- Number of fragments per segment,
- Fragment and segment duration,
- How many fragments/segment should be cached for DVR in Media Services.

Segment Numbering After Restart

Encoders should not use segment numbers on a restart or when it recovers from an outage. Encoders should create segment numbers at the next available monotonically increasing sequence. Alternatively, encoders can use unique session ID that changes upon restart so reusing the original segment numbers would not result in an overlap.

Reuse of file names is undesirable and will result in old content being served.

Logging and Reporting

To enable better troubleshooting of issues, it is recommended that the encoder logs errors and reports them. In the event that an encoder experiences an issue, it is far more likely to be resolved and avoided in the future if logs are available for review.

Parallel Uploads

To enable better throughput it is recommended that the encoder upload multiple bitrates in parallel using multiple sockets.

Encoders that cannot use multiple sockets to upload in parallel will likely experience poor performance as a result of bandwidth contention.

Chunk Encoding on Uploads

It is recommended that the encoder use HTTP chunked encoding to upload files more efficiently.

Encoders that cannot use chunked encoding may experience additional hand waving latency.

Chapter 6. Media Services Live for DASH: Live Encoder Guidelines

In This Chapter

This chapter explains the recommended and required specifications for a live stream encoder that is compatible with DASH Industry Forum specifications. Compatible encoders should be capable of successfully streaming a live event, but meeting these guidelines does not guarantee performance of any encoder in Media Services Live.

Recommended items will provide end-users a better experience. The limitations of not having those items are explained where appropriate.

Note: Media Services Live: DASH does not honor live encoders' ***HTTP DELETE*** *requests. While Media Services' servers will return a 200 HTTP status code, the content is not actually deleted as requested.*

Requirements

The following items are **required** for live encoders to function in Media Services Live and described in the following sections. It is not recommended to use any encoder that does not meet these requirements.

- Compliance with DASH-IF
- URL Configurability
- PUT/POST Request
- Segment Numbering
- URL Formatting
- Upload Retries and Timeouts
- Upload Order
- TCP Upload Optimization
- Key Frame Alignment
- Mime Types
- Encoder Identification

Additionally, the following items are **recommended** for encoders in Media Services Live and described in these sections:

- Primary and Backup Workflows
- DNS Lookups, Ongoing
- Alternate Hostname Uploads
- Segment Configurability
- Segment Numbering After Restart
- Logging and Reporting
- Parallel Uploads
- Chunk Encoding on Uploads

Compliance with DASH-IF

Encoders must meet the recommendations set by DASH Industry Forum. For details, refer to http://dashif.org/w/2013/08/DASH-AVC-264-v2.00-hd-mca.pdf.

URL Configurability

The encoder must allow specification of a POST server address also known as a POST URL, upload URL, or ingress URL.

The encoder must allow specification of a playback address also known as a playback URL or retrieval link.

PUT/POST Request

The encoder must be capable of uploading segments and **mpd** files using multiple **PUT/POST** requests. Each mpd, initialization segment, and media data segment should be uploaded in its own request.

Segment Numbering

- Segment numbers must not repeat when a live encoder restarts. This can be achieved when the segment numbers are derived from current epoch time. This results in always increasing segment numbers and prevents overwriting data in the storage system.
- A media segment must not have any reference to other fragment or segment.

URL Formatting

The encoder must be capable of posting URL in the formats described in the following sections:

- mpd
- Initialization Segment
- Media Data Segment

mpd

The encoder must upload the **.mpd** file at least every 30 seconds. The content of the file may or may not change on each upload.

The URL format of **.mpd** is as follows:

http://{hostname}/{format}/{streamID}/{eventname}/{anyname}.mpd

For example:

http://post.500002.testconfig.r.akamaientrypoint.net/dash/500002/test79/dash.mpd

Initialization Segment

The initialization segment must be uploaded only once for each bitrate/representation.

The URL format of an initialization segment is as follows:

http://{hostname}/{format}/{streamID}/{eventname}/{representationID.ext with init string}.{any extension other than mpd}

*Note: Make sure that the initialization header has **init** keyword in it. The file with **init** extension is also acceptable.*

Examples:

http://post.500002.testconfig.r.akamaientrypoint.net/dash/500002/test79/5000kbps.**init**

http://post.500002.testconfig.r.akamaientrypoint.net/dash/500002/test79/500kbps-**init**-test79.header

http://post.500002.testconfig.r.akamaientrypoint.net/dash/500002/test79/5000kbps-**init**ialization.mp4

Media Data Segment

The media data segment (**m4s, mp4, mp4v, ismv, m4v, m4a, mp4a, isma**) must be uploaded every 6 seconds for a segment duration of 6 seconds.

The URL format of media data segments are as follows:

http://{hostname}/{format}/{streamID}/{eventname}/{representationID}-{segment number}.{m4s,mp4,mp4v,ismv,m4v,m4a,mp4a,isma}

Note: It is a requirement to use the hyphen (-) before segment number.

Example:

http://post.500002.testconfig.r.akamaientrypoint.net/dash/500002/test79/500kbps-100.mp4

Upload Retries and Timeouts

The following table summarizes how the failure conditions are handled.

Error Condition	Object Type	Handling
-TCP connection attempt timeout -Abort midstream -TCP send or receive timeout	Manifest	Re-resolve DNS on each retry and retry indefinitely every second.
-TCP connection attempt timeout -Abort midstream -Response timeout -TCP send or receive timeout	Data (media, timed text)	Re-resolve DNS on each retry and attempt 3 retries every 500ms. After 3 retry attempts, drop current data and continue with next segment data.
HTTP 403, 400	Any	Do not retry - Display Error Reason
HTTP 500	Data (media, timed text)	Re-resolve DNS on each retry and attempt 3 retries every 500 millisecond. After 3-retry attempts, drop current data and continue with next segment data.
HTTP 500	Manifest	Re-resolve DNS on each retry and retry indefinitely every 1 second.

Upload Order

The upload order for MPEG DASH must be as follows:

1. MPD file — uploaded at least every 30 seconds.
2. Initialization Segment — uploaded only once for each bitrate/representation.
3. Media Segment — uploaded every 6 seconds for a segment duration of 6 seconds.

TCP Upload Optimization

To maximize upload performance, a TCP window size of 64k is recommended. The encoder must be able to specify an alternate size according to network conditions.

Key Frame Alignment

- Encoder must align timestamps across representations/bitrates.
- Similarly GOP size/key-frame interval across representations/bitrates must be same.
- MPEG DASH segment size must be 6s. Shorter GOP sizes like 2 seconds are recommended and provide better seeking into the media.

Mime Types

The following table provides mime type requirements for different support extensions.

Extension	Mime Type
.mpd	application/dash+xml
.mp4, .m4v, .mp4v, .ismv	video/mp4
.m4s	video/iso.segment
.m4a, .mp4a	audio/mp4

Encoder Identification

Encoder must be capable to send User-Agent header that provides information about the encoder brand name, version number, and build number in a readable format.

Recommendations

The following items are recommended for encoders in Media Services Live. Encoders that cannot provide these will have limited functionality. These limitations are described in the following sections:

- Primary and Backup Workflows
- DNS Lookups, Ongoing
- Alternate Hostname Uploads
- Segment Configurability
- Segment Numbering After Restart
- Logging and Reporting
- Parallel Uploads
- Chunk Encoding on Uploads

Primary and Backup Workflows

To avoid interruptions in the distributed system, the encoder should be capable of publishing the same content to primary and backup locations.

Encoder should be able to input primary and backup publishing and playback host-names and primary and backup paths for a given stream. The same content (mpd, initialization segments, and media segments) should be simultaneously published to primary and backup paths.

Publishing Workflow

The encoder will publish primary and backup streams with following URLs:

- *Primary Publishing hostname*:

  http://post.{streamID}.dash.{customer}.r.akamaientrypoint.net/

- *Backup Publishing hostname*:

  http://postb.{streamID}.dash.{customer}.r.akamaientrypoint.net/

- *Primary Publishing path*:

  dash/{streamID}/{eventname}/{manifest}.{object}

  Examples:

  http://post.500002.testconfig.r.akamaientrypoint.net/dash/500002/test79/dash.mpd

  or

  http://post.500002.testconfig.r.akamaientrypoint.net/dash/500002/test79/5000kbps-10.m4s

- *Backup Publishing path*:

  dash/{streamID}-b/{eventname}/{manifest}.{object}

  Examples:

  http://postb.500002.testconfig.r.akamaientrypoint.net/dash/500002-b/test79/dash.mpd

  http://postb.500002.testconfig.r.akamaientrypoint.net/dash/500002-b/test79/10.m4s

Note the following:

- The encoder will publish stream to both locations — primary and backup.
- The segments and mpd will be duplicated across primary and backup and should be identical.

Playback Workflow

The encoder should allow primary and backup paths to be inputed in following way:

- *Base Playback URL:*

  http://{customer}-i.akamaihd.net/

- *Primary Playback path:*

  dash/live/{streamID}/{eventname}/{object}

  Examples:

  http://testconfig-i.akamaihd.net/dash/live/500002/test79/dash.mpd

  http://testconfig-i.akamaihd.net/dash/live/500002/test79/dash/5000kbps-10.m4s

- *Backup Playback path:*

  dash/live/{streamID}-b/{eventname}/{object}

  Examples:

  http://testconfig-i.akamaihd.net/dash/live/500002-b/test79/dash.mpd

  http://testconfig-i.akamaihd.net/dash/live/500002-b/test79/dash/5000kbps-10.m4s

The above playback information will be embedded in the manifest **mpd** file as two <**BaseURL**>s, as highlighted in Green in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="urn:mpeg:dash:schema:mpd:2011"
      xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://
      standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
      DASH_schema_files/DASH-MPD.xsd" type="dynamic"
      minimumUpdatePeriod="PT30S" availabilityStartTime="2014-02-
      05T22:29:56" minBufferTime="PT12S" timeShiftBufferDepth="PT1M0S"
      profiles="urn:mpeg:dash:profile:isoff-live:2011">
  <BaseURL>http://test-i.akamaihd.net/dash/live/10001/linearprogram1/
  </BaseURL>
  <BaseURL>http://test-i.akamaihd.net/dash/live/10001-b/linearprogram/
  <BaseURL>
  <Period start="PT0S" duration="PT1M0.6S" id="1">
      <AdaptationSe01t mimeType="video/mp4"
      codecs="avc1.42CE,mp4a.40.5" frameRate="15000/1001"
      segmentAlignment="true" subsegmentAlignment="true"
      startWithSAP="1" subsegmentstartWithSAP="1"
      bitstreamSwitching="true">
          <ContentComponent contentType="video" id="1"/>
          <SegmentTemplate timescale="90000" duration="540000"
             startNumber="74247"/>
          <Representation id="1" width="640" height="360"
              bandwidth="600000">
              <SubRepresentation contentComponent="1" bandwidth="600000"
                  codecs="avc1.42C01E"/>
              <SegmentTemplate duration="540000" startNumber="74247"
                  media="dash_video600-$Number$.mp4
                  initialization="dash_video600-.init"/>
          </Representation>
          <Representation id="2" width="320" height="180"
              bandwidth="200000">
              <SubRepresentation contentComponent="1" bandwidth="200000"
                  codecs="avc1.42C01E"/>
              <SegmentTemplate duration="540000" startNumber="74248"
                  media="dash_video200-$Number$.mp4"
                  initialization="dash_video200-.init"/>
          </Representation>
      </AdaptationSet>
  </Period>
</MPD>
```

If the player detects a 404/error on a segment request, it switches to an alternate stream and vice versa.

DNS Lookups, Ongoing

The encoder ideally should perform ongoing DNS lookups while streaming.

Be aware that encoders are periodically asked to re-resolve DNS. For long running streams, old Entry Points may get overloaded or go down for maintenance or bad connectivity. In this case, when the IP address changes, it is recommended that the encoder complete its current POST/PUT request, close the connection, and then reopen a connection to the new IP Address to move encoders to the best available Entry Point for them at that time. To ensure the most optimum throughput and to guarantee high stream availability, it is also recommended that the introduction of periodic DNS resolution be short (DNS TTLs for these records are generally 20 seconds).

Note: In the near future, encoding vendors will be required to support this recommendation in order to receive and maintain a qualified rating. we are working with our partners to implement this functionality, as it improves stream availability and ensures better throughput. It is suggested that you make every effort to add this recommendation to your solution at your earliest convenience.

Alternate Hostname Uploads

The encoder should be capable of using multiple hostnames when publishing multiple bitrates to better handle failure situations.

Encoders that cannot use multiple hostnames will have a single point of failure. In the event that the single upload host being used is lost, the event will be interrupted until the host returns or a new host is provisioned.

Segment Configurability

Encoder should be capable of configuring:

- Segment duration;
- How many segment should be cached for DVR in Media Services Live.

Segment Numbering After Restart

Encoders should not reuse segment numbers on a restart or when it recovers from an outage. It should create segment numbers at the next available monotonically increasing sequence. Alternatively, encoders can use unique session ID that changes upon restart so reusing the original segment numbers would not result in an overlap.

Reuse of file names is undesirable and would result in old content being served.

Logging and Reporting

To enable better troubleshooting of issues, it is recommended that the encoder logs errors and reports them. In the event that an encoder experiences an issue, it is far more likely to be resolved and avoided in the future if logs are available for review.

Parallel Uploads

To enable better throughput it is recommended that the encoder upload multiple bitrates in parallel using multiple sockets.

Encoders that cannot use multiple sockets to upload in parallel will likely experience poor performance as a result of bandwidth contention.

Chunk Encoding on Uploads

It is recommended that the encoder use HTTP chunked encoding to upload files more efficiently.

Encoders that cannot use chunked encoding may experience additional hand waving latency.