Adobe Primetime Dynamic Adaptive Streaming over HTTP (DASH) Interoperability Guidelines: January 2016 FINAL

Adobe® Primetime Dynamic Adaptive Streaming over HTTP (DASH) Interoperability Guidelines: January 2016 FINAL

Published January 2016.

1. Scope

This document outlines the requirements for creating DASH content that is compatible with Adobe Primetime.

This document assumes a strong familiarity with [DASH], [DASH-IF], [CENC], and [AAXS-OVERVIEW]. Readers are strongly encouraged to familiarize themselves with the above documents prior to reading this document.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

The following typographical conventions are used in this document:

- `@examplePrefix:exampleAttribute` is an XML attribute with name "exampleAttribute" that belongs to the XML namespace with prefix "examplePrefix". "examplePrefix:" may be omitted to indicate an attribute that belongs to the default XML namespace.
- `<examplePrefix:exampleElement>` is an XML element with name "exampleAttribute" that belongs to the XML namespace with

prefix "examplePrefix". "examplePrefix:" may be omitted to indicate an element that belongs to the default XML namespace.

- `<exampleElement>@exampleAttribute` is the XML attribute with name "exampleAttribute" that is contained by an XML element with name "exampleElement".

3. Primetime DASH Interoperability Point

This section defines the *Primetime DASH interoperability point* (IOP). See [DASH] for a definition of DASH, DASH interoperability point ([DASH] Section 8.1), and additional terminology and concepts related to DASH.

Primetime DASH MPDs, *Primetime DASH clients*, and *Primetime DASH servers* refer to MPDs, clients, and servers that conform to the Primetime DASH interoperability point.

3.1. Relationship to DASH-AVC/264

This section is non-normative.

This section describes the relationship of the Primetime DASH IOP to the DASH-AVC/264 Main Profile IOP. See [DASH-IF] Section 6.1 for a definition of the DASH-AVC/264 Main Profile IOP.

At a high level, Primetime DASH is a variant of the DASH-AVC/264 Main Profile.

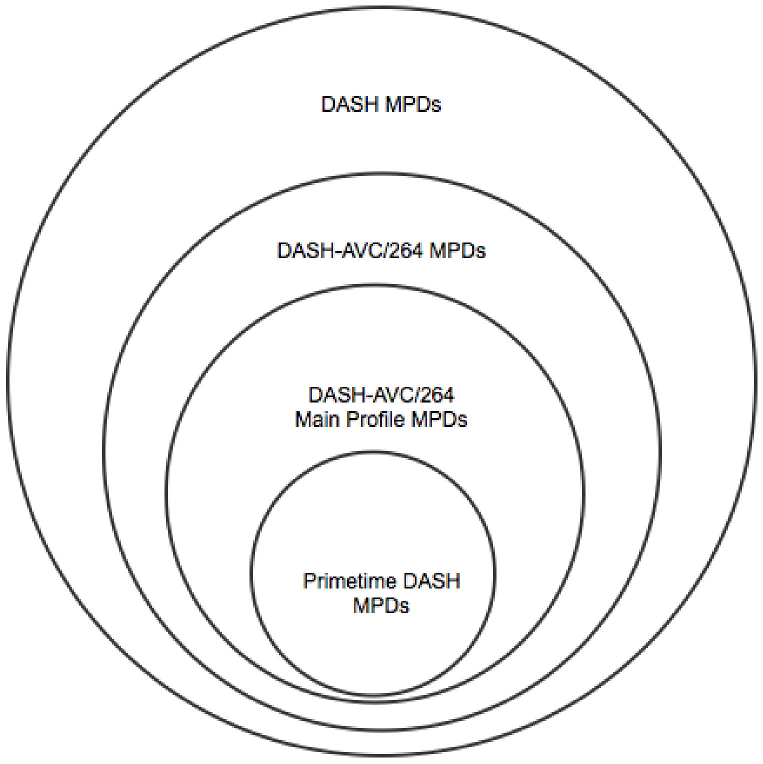

Figure 1. Illustration of the relationship between DASH, DASH-AVC/264, and Primetime DASH MPDs.

The requirements imposed on Primetime DASH MPDs are stronger than those imposed on DASH-AVC/264 Main Profile MPDs. **A Primetime DASH MPD is always a DASH-AVC/264 Main Profile MPD.** A DASH-AVC/264 Main Profile MPD is not necessarily a Primetime DASH MPD.

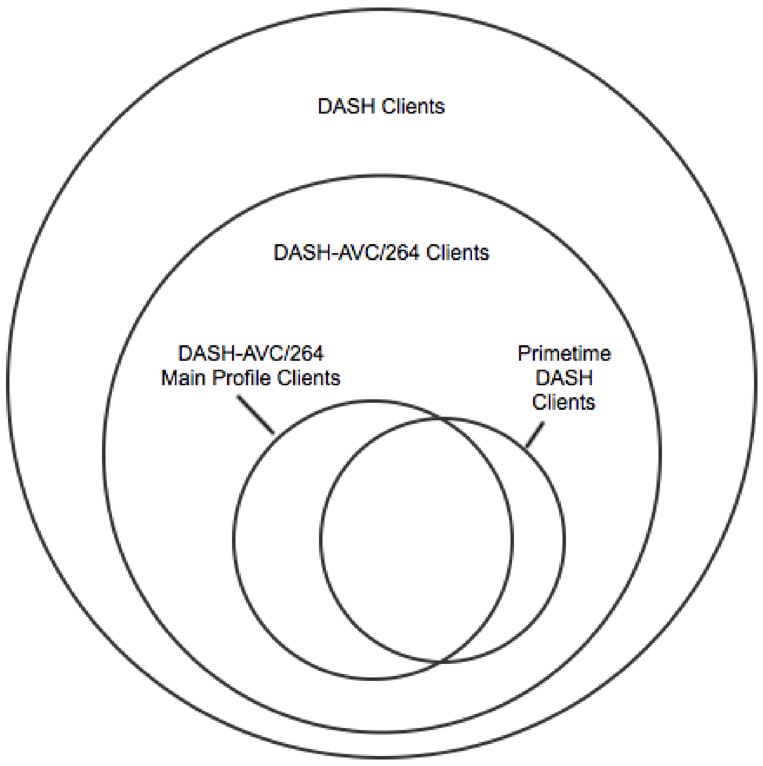

Figure 2. Illustration of the relationship between DASH, DASH-AVC/264, and Primetime DASH Clients.

Most Primetime DASH client requirements are also DASH-AVC/264 Main Profile client requirements.

Some DASH-AVC/264 Main Profile client requirements are not Primetime DASH client requirements. For example, Primetime DASH clients are not required to support the ISO BMFF On Demand Profile, but such support is required of DASH-AVC/264 Main Profile clients. This document highlights capabilities that are requirements for DASH-AVC/264 Main Profile clients, but are not requirements for Primetime DASH clients by marking such capabilities as "OPTIONAL".

Some Primetime DASH client requirements are not DASH-AVC/264 Main Profile client requirements. For example, Primetime DASH clients are required to implement WebVTT support, but this is not required of DASH-AVC/264 Main Profile clients

The requirements of Primetime DASH clients do not conflict with those of DASH-AVC/264 Main Profile clients. Therefore, it is possible for a client to conform to both the Primetime DASH and DASH-AVC/264 Main Profile interoperability points.

3.2. Basics

Primetime DASH MPDs, clients, and servers SHALL conform to [DASH-IF] Sections 3-5. Notable topics covered by these sections include (but are not limited to):

- DASH features including MPD constraints, segment format constraints, trick mode support and other topics
- Client implementation requirements and guidelines
- Video and audio codecs
- Closed captioning via CEA-608/708
- Content protection

Any constraints outlined in the current document take precedence over those outlined in [DASH-IF].

Primetime DASH client support for the ISO BMFF Live Profile is REQUIRED, but support for the ISO BMFF On Demand Profile is OPTIONAL. See [DASH] Section 8 for definitions of the ISO BMFF Live and On Demand Profiles.

A Primetime DASH MPD SHALL conform to the ISO BMFF Live Profile. `<MPD>@profiles` SHALL include "`urn:mpeg:dash:profile:isoff-live:2011`". On Demand content can be expressed by specifying an `<MPD>@type` of "`static`" with the ISO BMFF Live Profile .

3.3. Period

A Primetime DASH client SHALL support playback of MPDs that contain one or more periods. See [DASH] Section 5 for more information on periods.

3.4. Segments

A Primetime DASH media segment SHALL contain exactly one media subsegment; A media segment SHALL contain exactly one `` `moof` `` box. See [DASH] Section 6.2 for a definition of media segments, subsegments, and initialization segments.

In a Primetime DASH MPD:

- Each representation SHALL have exactly one initialization segment assigned.
- `<SegmentTemplate>@media` SHALL be present.
- `<Representation>@dependencyId` SHALL NOT be present.
- Each `<Representation>` SHALL contain or inherit `@codecs`.
- Representations within an adaptation set SHALL specify the same or compatible values for `@codecs`. For example, representations within the same adaptation set shall not specify both the H.265/HEVC and H.264/AVC codecs. However, representations within the same adaptation set may specify different, compatible profiles of the same codec.

A Primetime DASH client SHALL support:

- `<SegmentTimeline>` and its attributes when present at the `<AdaptationSet>` and/or `<Representation>` level.
- The following attributes and elements of `<SegmentTemplate>` :
    - `@timescale`
    - `@presentationTimeOffset`
    - `@initialization`
    - `@media`
    - `@duration`
    - `@startNumber`
    - `<Initialization>`

Primetime DASH client support for the following features is OPTIONAL:

- multiple media subsegments
- `<URLType>` `@range`
- `<URLType>` `@sourceURL`
- `<BaseURL>` `@byteRange` and [DASH-IF] Section 3.4.3.

3.5. Video and Audio Codecs

A Primetime DASH client SHALL be able to decode H.264/MPEG AVC Progressive High Profile up to level 3.1.

A Primetime DASH client SHALL be able to decode MPEG-4 HE-AAC v2 Level 2 Audio. Primetime DASH client support for Dynamic Range Control is OPTIONAL.

A Primetime DASH MPD SHOULD contain video encoded at H.264/MPEG AVC Progressive High Profile up to level 3.1.

A Primetime DASH MPD SHOULD contain audio encoded as MPEG-4 HE-AAC v2 Level 2 Audio.

3.6. Content Protection

A Primetime DASH client SHOULD support at least one of the following content protection schemes:

- Adobe Access
- Common Encryption
- Microsoft PlayReady

Content protected by Adobe Access SHALL conform to Section 4.

Content protected by Common Encryption SHALL conform to [DASH-IF] Section 5 .

Content protected by Microsoft PlayReady SHALL conform to [DASH-MSPR].

3.7. Closed Captions

A Primetime DASH client SHALL support at least one of the following caption formats:

- WebVTT
- CEA-608 (cc_type 00 or 01)

Primetime DASH client support for the following caption formats is OPTIONAL:

- SMPTE-TT
- CEA-708 (cc_type of 10 or 11) captions

See [WEBVTT], [CEA608], [CEA708] , and [FCC].

3.7.1. CEA-608

A Primetime DASH MPD advertises the presence of CEA-608 captions by specifying a supplemental property descriptor with a `@schemeIdURI` of "`urn:dashif:org:cc:cea608:sei:2014`". See [DASH] Section 5.8.4.9 for more information on supplemental property descriptors.

Content that advertises the supplemental property descriptor SHALL conform to the CEA-608 requirements set forth in [DASH-IF] Section 4.4.3 .

The CEA-608 supplemental property descriptor SHOULD be provided at the adaptation set level. If any representation in a Primetime DASH adaptation set provides CEA-608 captions, all representations in the adaptation set SHOULD provide CEA-608 captions. A Primetime DASH client that supports CEA-608 captions SHALL support captions signaled at the adaptation set level , but support for captions signaled at the representation level is OPTIONAL.

3.7.2. WebVTT

A Primetime DASH MPD signals the presence of WebVTT content by including an `<AdaptionSet>` with a `@mimeType` of "`text/vtt`". The `<AdaptationSet>` SHALL specify `@lang`. The adaptation set SHALL contain a single representation.

Under Primetime DASH, WebVTT captions are delivered as a single segment or multiple segments. WebVTT captions for live content SHOULD be delivered in multiple segments. WebVTT captions for on-demand content MAY be delivered in multiple segments, or they MAY be delivered as a single segment encompassing the entire duration of the content.

The `<Representation>` and/or `<AdaptationSet>` SHALL specify segments using `<SegmentTemplate>`.

A Primetime DASH MPD can signal the presence of multiple WebVTT resources in different languages by including an `<AdaptationSet>` and corresponsponding `@lang` for each language.

Each WebVTT segment SHALL be a resource in the WebVTT format as described in [WEBVTT]. Each WebVTT segment SHALL be self-contained. That is, a WebVTT segment SHALL not require the usage of other WebVTT segments to render captions properly.

A WebVTT segment SHALL contain all cues that are entirely contained within the segment boundaries specified in the MPD. The segments SHALL also contain all cues that intersect the MPD segment boundaries. When maximum rendering fidelity is desired, cues contained within a segment SHOULD be entirely contained within the MPD segment boundaries, but it is acknowledged that this is not always feasible. To maximize consistency between captions and video in live content, the segment boundaries for WebVTT segments for live content SHOULD equal the segment boundaries of the corresponding video segments.

The requirements for Primetime DASH WebVTT segments are equivalent to the requirements for HTTP Live Streaming WebVTT segments, as described in [HLS] Section 3.0. **A WebVTT segment that is properly formatted for use with HTTP Live Streaming is a valid Primetime DASH WebVTT segment.**

3.8. Trick Modes

A Primetime client SHALL support playback of trick mode representations and segments.

To advertise the availability of trick mode representations, a Primetime MPD SHALL conform with [DASH-IF] Section 3.2.9, which instructs the content author to:

> - *add an Adaptation Set that that only contains trick modes Representations*
> - *annotate the Adaptation Set with an EssentialProperty descriptor or SupplementalProperty descriptor with URI " http://dashif.org/guide lines/trickmode" and the `@value` the value of `@id` attribute of the Adaptation Set to which these trick mode Representations belong. The trick mode Representations must be time-aligned with the Representations in the main Adaptation Set. The value may also be a white-space separated list of `@id` values. In this case the trick mode Adaptation Set is associated to all Adaptation Sets with the values of the `@id`.*
> - *signal the playout capabilities with the attribute `@maxPlayoutRate` for each Representation in order to indicate the accelerated playout that is enabled by the signaled codec profile and level.*

In addition, the following constraints apply to trick mode representations and segments:

- Each trick mode media segment SHOULD contain exactly one independent frame.
- A trick mode representation SHALL include `@frameRate`. If a trick mode segment is 1 second long and contains a single independent frame, the frame rate is 1 frames-per-second.

- A trick mode representation SHALL include `@maxPlayoutRate`. The `@maxPlayoutRate` value reflects the maximum rate at which the content frames may be presented to the decoder while conforming to the the constraints of the codec profile and level. This is needed in order to ensure the decoder can decode frames at the required rate.

3.9. Alternative Language Audio

A Primetime DASH client SHALL have the ability to select between multiple audio tracks by `@lang`.

A Primetime DASH MPD follows the same mechanism for signalling the presence of multiple audio tracks as [DASH-IF]. Under [DASH-IF], audio representations that represent the same language are grouped into an adaptation set and audio representations of different languages are in separate adaptation sets. As stated in [DASH-IF] Section 3.2, `@lang` shall be present on `<AdaptationSet>` .

3.10. Ad Insertion

A Primetime DASH client SHALL have the ability to perform ad insertion.

DASH content that will be used for ad insertion by a Primetime DASH client SHALL be Primetime DASH content. Notably, the content shall begin with an SAP of type “1”, as is true for any Primetime DASH and DASH-IF content.

A Primetime DASH MPD SHALL signal cues using the semantics specified in [DPI].

4. Adobe Access (AAXS) and DASH

This section describes topics related to the Adobe Access (AAXS) DRM scheme and its usage with DASH.

The following terms will be used in this section:

AAXS content encryption key is equivalent to the term "content encryption key" as defined in [AAXS-OVERVIEW].

AAXS metadata is equivalent to the term "DRM metadata" as defined in [AAXS-OVERVIEW].

An *AAXS segment* is a (media or initialization) segment that utilizes the Adobe Access DRM scheme.

AAXS information refers to information required by the Adobe Access DRM scheme.

An *AAXS MPD* is an MPD that contains content protected by the Adobe Access DRM scheme.

4.1. Overview

4.1.1. Relationship to Common Encryption

This section is non-normative.

The Adobe Access DASH requirements are compatible and compliant with both the Common Encryption scheme and DASH-AVC/264 as defined by [CENC] and [DASH-IF].

This specification provides definitions for AAXS-specific elements such as `<access:pssh>`. These elements are intended to enable features that are not expressible using [CENC] and [DASH-IF] semantics. If AAXS-specific features are not needed, an MPD may include the [CENC] and [DASH-IF] versions of the elements and still be fully compliant with this specification.

4.1.2. AAXS Information Locations

This section is non-normative.

The AAXS `` `pssh` `` box is a container for information required by the Adobe Access DRM scheme. Information included within an AAXS `` `pssh` `` box includes a mapping of key identifiers (KID's) to AAXS metadata and encrypted rotation keys. AAXS metadata is an opaque binary object created by the Adobe Access packaging toolchain. See Section 4.2.1 for a normative description of the AAXS `` `pssh` `` box.
See [AAXS-OVERVIEW] or [AAXS-REF-IMPL] for additional information on the Adobe Access toolchain.

AAXS `` `pssh` `` boxes can be included in multiple locations:

- They can be embedded within the initialization segment.
- They can be embedded within media segments.

- They can be embedded within an MPD using `<access:pssh>`. `<access:pssh>` is a child element of an AAXS `<ContentProtection>`, which in turn is a child of `<AdaptationSet>` or `<Representation>`. See Section 4.3.4.
- They can be referenced as an external resource by an MPD using `@access:URI` on `<access:pssh>` .
- They can be embedded within an MPD using `<cenc:pssh>`. `<access:pssh>` is often preferable to `<cenc:pssh>` due to `<access:pssh>`'s additional capabilities. If such capabilities is not desired, `<cenc:pssh>` may be desirable for interoperability with generic DASH clients.

Typically, AAXS `pssh` boxes are provided (or referenced) by the MPD when segments do not contain embedded AAXS `pssh` boxes or when the embedded AAXS information needs to be overridden.

4.1.3. AAXS Information Precedence

This section is non-normative.

As described in Section 4.1.2, AAXS information can be included in multiple locations for a given presentation. This section describes the relation precedence of AAXS information when it is present in multiple locations. This section is required to address a few key capabilities of AAXS MPDs:

1. Unlike `<cenc:pssh>`, `<access:pssh>` can be used to override information embedded within a media segment. DASH-IF does not describe a means to override media segment DRM information.
2. `To enable compatibility with Common Encryption and DASH-IF, an AAXS MPD can contain <access:pssh> and/or <cenc:pssh>.`

The following describes how to determine the AAXS information for a given key identifier:

> If `<access:pssh>` or `<cenc:pssh>` is present at the representation level,
>
> > Search the following locations, in order, for an AAXS `pssh` box that provides AAXS information for the desired key identifier:
> >
> > 1. Representation `<access:pssh>` elements that specify `@access:startTimestamp`
> > 2. Media segment
> > 3. Representation `<access:pssh>` element that does not specify `@access:startTimestamp`
> > 4. Representation `<cenc:pssh>`
> >
> > Use the AAXS information for the first such AAXS `pssh` box. Notice that the initialization segment AAXS `pssh` box is not inspected.
>
> Otherwise if `<access:pssh>` or `<cenc:pssh>` is present at the adaptation set level,
>
> > Search the following locations, in order, for an AAXS `pssh` box that provides AAXS information for the desired key identifier:
> >
> > 1. Adaptation set `<access:pssh>` elements that specify `@access:startTimestamp`
> > 2. Media segment
> > 3. Adaptation set `<access:pssh>` element that does not specify `@access:startTimestamp`
> > 4. Adaptation set `<cenc:pssh>`
> >
> > Use the AAXS information for the first such AAXS `pssh` box. Notice that the initialization segment AAXS `pssh` box is not inspected.
>
> Otherwise (if AAXS information is not provided by the MPD)
>
> > Search the following locations, in order, for an AAXS `pssh` box that provides AAXS information for the desired key identifier:
> >
> > 1. Media segment
> > 2. Initialization segment
> >
> > Use the AAXS information for the first such AAXS `pssh` box. Notice that the initialization segment AAXS `pssh` box is inspected.

See Sections 4.3.5 and 4.3.6 for a normative description of the above process.

4.2. AAXS Information in ISOBMFF

This section defines the semantics for specifying AAXS information within ISO Base Media File Format (BMFF) resources. See [ISOBMFF] for a description of the ISO Base Media File Format.

When applied to Primetime DASH or DASH-AVC/264 content, this section describes the semantics for AAXS information that is embedded within the initialization and media segments.

4.2.1. AAXS Protection System Specific Header (`pssh`) Box

The *AAXS Protection System Specific Header Box* is a container for other boxes that provide AAXS information.

An AAXS `pssh` box is a Protected System Specific Header (`pssh`) box whose SystemID is "0xF239E769EFA348509C16A903C6932EFB". See [CENC] for a definition of the `pssh` box.

The Data of an AAXS `pssh` box SHALL begin with one AAXS Metadata box. If DataSize is greater than the size of the enclosed AAXS metadata box, the meaning of the additional Data bytes is currently undefined.

See Section 4.4.1 for an example AAXS `pssh` box.

4.2.2. AAXS Metadata (`amet`) Box

The *AAXS Metadata Box* supplies information required the AAXS DRM system such as the AAXS metadata or encrypted rotation keys.

Using the conventions of [ISOBMFF], the syntax of the `amet` box is:

```
aligned(8) class AdobeAccessMetadataBox(bit(24) flags)
    extends FullBox(‘amet’, version = 0, flags) {
    unsigned int(32) KID_count;
    int i;
    for (i=0; i < KID_count; i++) {
        unsigned int(8)[16] KID;
        if (flags & 1) {
            unsigned int(8)[16] encrypted_key;
        }
    }
    if (flags & 2) {
        unsigned int(32) metadata_size;
        unsigned int(8)[metadata_size] metadata;
    }
}
```

KID_count specifies the number of KID entries.

KID specifies a key identifier.

The flags field is used to indicate whether the `amet` box contains AAXS metadata, encrypted rotation keys, or both:

1s' bit of flags signals that the `amet` box contains encrypted rotation keys.

2s' bit of flags signals that the `amet` box contains AAXS metadata.

When the `amet` box contains encrypted rotation keys:

encrypted_key is the encrypted rotation key associated with the preceding KID. See Section 4.3.6.

The `amet` box does not associate encrypted rotation keys with the KID's specified in the parent `pssh` box unless those KID's are also included in the `amet` box.

When the `amet` box contains AAXS metadata:

metadata_size is the size in bytes of metadata. metadata_size SHALL NOT be 0.

metadata is the AAXS metadata. The AAXS metadata is associated with all key identifiers specified in the `amet` box as well as those specified in the enclosing `pssh` box. An AAXS metadata is an opaque binary object created by the Adobe Access packaging toolchain. Se e [AAXS-OVERVIEW] or [AAXS-REF-IMPL] for additional information on the Adobe Access toolchain.

If an `amet` box has the 1's bit or the 2's bit of flags set, the sum of the `amet` box's KID_count and the parent `pssh` box's KID_count SHA LL be greater than zero.

The meaning of all bits of flags except the 1's or 2's bit is undefined. These bits are reserved for future use and their value SHOULD be zero.

4.3. AAXS Information in the DASH MPD

This section defines the semantics for specifying AAXS information within DASH MPD's.

See Section 4.4.3 for an example of AAXS information in the MPD.

4.3.1. AAXS XML Namespace

Elements and attributes that are specific to the AAXS DRM scheme use the AAXS XML namespace. The AAXS XML namespace is "`urn:adobe:aaxs-dash:2015`". In the context of this document, elements and attributes that use the AAXS XML namespace are referred to by the prefix "`access`".

4.3.2. AAXS MPD

An AAXS MPD SHALL conform to [DASH-IF] Section 7.

As described in [DASH-IF] Section 7.5.2, a `<ContentProtection>` descriptor for the mp4 Protection Scheme SHALL be present in the MPD.

4.3.3. AAXS `<ContentProtection>`

An AAXS `<ContentProtection>` contains AAXS information for a given Adaptation Set or Representation.

An AAXS `<ContentProtection>` element is a `<ContentProtection>` element that specifies a `@schemeIdUri` of "`urn:uuid:F239E769-EFA3-4850-9C16-A903C6932EFB`" and a `@value` of "AdobeAccess 5.0". See [DASH] for a definition of `<ContentProtection>` . An AAXS `<ContentProtection>` does not use the AAXS XML namespace.

The AAXS `<ContentProtection>` is OPTIONAL, even for Adaptation Sets or Representations that contain AAXS content.

The following attributes are defined for an AAXS `<ContentProtection>` , in addition to any defined for a `<ContentProtection>` by [DASH] or [DASH-IF]:

`@cenc:default_`KID

See [CENC]. This attribute SHOULD be present on an AAXS `<ContentProtection>` .

`@access:default_`IV_size

This attribute SHOULD be present on an AAXS `<ContentProtection>` . The value is an unsigned integer. If present, the value SHALL be equal to the value specified by the initialization segment's `` `tenc` `` box. See [CENC] for a description of the `` `tenc` `` box.

`@access:default_`IsEncrypted

This attribute SHOULD be present on an AAXS `<ContentProtection>` . The value is an unsigned integer. If present, the value SHALL be equal to the value specified by the initialization segment's `` `tenc` `` box. See [CENC] for a description of the `` `tenc` `` box.

The following child elements are defined for an AAXS `<ContentProtection>` :

`<access:pssh>`

An AAXS `<ContentProtection>` MAY contain one or more child `<access:pssh>` . An AAXS `<ContentProtection>` SHALL contain at most one child `<access:pssh>` that does not specify `@access:startTimeStamp`. An AAXS `<ContentProtection>` MAY contain more than one child `<access:pssh>` that specifies `@access:startTimeStamp`.

`<cenc:pssh>`

An AAXS `< ContentProtection>` MAY contain one child `<cenc:pssh>` . An AAXS `< ContentProtection>` SHALL NOT contain more than one child `<cenc:pssh>` .

If an AAXS `< ContentProtection>` contains one child `<cenc:pssh>` and at least one `<access:pssh>` , the information provided by the `<cenc:pssh>` and the `<access:pssh>` SHALL be consistent; If an AAXS `<ContentProtection>` contains one child `<cenc:pssh>`:

- The AAXS `< ContentProtection>` SHALL contain at most one child `<access:pssh>` .
- The `<access:pssh>` SHALL NOT specify `@access:startTimestamp`.
- The embedded or referenced AAXS `` `pssh` `` box the `<access:pssh>` SHALL be identical to the box embedded within the `<cenc:pssh>` .

AAXS `<ContentProtection>` typically appears at the `<AdaptationSet>` level, but MAY appear at the `<Representation>` level. The behavior of AAXS `<ContentProtection>` at the `<Subrepresentation>` level is currently undefined.

4.3.4. `<access:pssh>`

An `<access:pssh>` provides an AAXS `pssh` box.

When `@access:URI` is not present on this element, the element value is an AAXS `` `pssh` `` box expressed as base64. The value includes the box header.

The following attributes are defined for `<access:pssh>` :

`@access:URI`

> This attribute is used to reference an external AAXS `` `pssh` `` box. The attribute value is a URI from which an AAXS pssh resource can be fetched. If this attribute is present, the element value SHALL be empty.
>
> The attribute value may be absolute or relative. Relative values use `<BaseURL>` for resolution. See [DASH] for a description of `<BaseURL>` .
>
> An AAXS pssh resource is a resource whose contents consist of a single AAXS `` `pssh` `` box.

`@access:startTimeStamp`

> The value of this OPTIONAL attribute is the start time from which the AAXS `` `pssh` `` box will apply. The time is specified as a non-negative integer number of milliseconds. See Sections 4.3.5 and 4.3.6 for a description of how this attribute is used in AAXS metadata and rotation key resolution.
>
> As described in Sections 4.1, 4.3.5, and 4.3.6, an `<access:pssh>` that does not specify `@access:startTimeStamp` overrides AAXS information provided by the initialization segment. An `<access:pssh>` that specifies `@access:startTimeStamp` takes precedence over AAXS in formation provided by media segments.

`@access:prefetchDeadline`

> The value of this OPTIONAL attribute is a time. When the attribute is present, an AAXS client SHOULD fetch the associated AAXS license by the time specified by `@access:prefetchDeadline`. The time is specified as a non-negative integer number of milliseconds .

4.3.5. Determining the AAXS Metadata

This section provides a normative description of how the AAXS metadata is determined for a given key identifier. For an informative description, see Section 4.1. See Section 4.4 for examples.

The `associated-aaxs-metadata` of (`k`,`t`,`r`), the AAXS metadata that is associated with key identifier `k`, presentation time `t` and Representation `r`, is determined by the following process:

```
If r contains an AAXS <ContentProtection>,
    Let current-content-protection-element = r's AAXS <ContentProtection> .
Else if r's <AdaptiveSet> contains an AAXS <ContentProtection>,
    Let current-content-protection-element = r's <AdaptiveSet >'s AAXS <ContentProtection> .
Else,
    Let current-content-protection-element = null.
If current-content-protection-element is not null and it contains one or more <access:pssh> and/or one <cenc:pssh> ,
    Let candidate-timestamped-pssh-elements = the set of child <access:pssh> of c that specify an @access:startTimestamp that is less than or equal to t.
    If candidate-timestamped-pssh-elements is not empty,
        Let matching-timestamped-pssh-element = the element of candidate-timestamped-pssh-elements with the minimum @access:startTimestamp.
        If matching-timestamped-pssh-element's AAXS `pssh` box associates k with AAXS metadata,
            The associated-aaxs-metadata is the AAXS metadata associated with k by matching-timestamped-pssh-element's AAXS `pssh` box. Done.
    If the media segment for time t and Representation r contains an AAXS `pssh` box,
        If the media segment's AAXS `pssh` box associates k with AAXS metadata,
            The associated-aaxs-metadata is the AAXS metadata associated with k by the media segment's AAXS `pssh` box. Done.
    Let untimestamped-pssh-element = the child <access:pssh> of c that does not specify @access:startTimestamp.
    If untimestamped-pssh-element exists and its `pssh` box associates k with AAXS metadata,
        The associated-aaxs-metadata is the AAXS metadata associated with k by untimestamped-pssh-element's `pssh`
        box. Done.

    Let cenc-pssh-element = the child <cenc:pssh> of c.

    If cenc-pssh-element exists and its `pssh` box associates k with AAXS metadata,

        The associated-aaxs-metadata is the AAXS metadata associated with k by cenc-pssh-element's `pssh` box. Done.

    The associated-aaxs-metadata does not exist (if it was not found in the above process). Done.

Else,

    If the media segment for time t and Representation r contains an AAXS `pssh` box,

        If the media segment's AAXS `pssh` box associates k with AAXS metadata,

            The associated-aaxs-metadata is the AAXS metadata associated with k by the media segment's AAXS `pssh`
            box. Done.

    If the initialization segment for Representation r contains an AAXS `pssh` box,

        If the initialization segment's AAXS `pssh` box associates k with AAXS metadata,

            The associated-aaxs-metadata is the AAXS metadata associated with k by the initialization segment's AAXS `pss
            h` box. Done.

    The associated-aaxs-metadata does not exist (if it was not found in the above process).
```

All key identifiers that appear in an AAXS protected segment SHALL have an associated AAXS metadata (that exists).

4.3.6. Rotation Keys

A *rotation key* is the key used to encrypt one or more segments.

In some cases, the rotation key is equal to the AAXS content encryption key (which is obtained from the AAXS metadata). In other cases, the rotation key differs from the AAXS content encryption key. To accommodate the latter, AAXS `pssh` boxes can associate key identifiers with an encrypted form of the rotation key. This encrypted form of the rotation key is known as the *encrypted rotation key*. The encrypted rotation key is the result of encrypting the rotation key with AES-128 ECB using the AAXS content encryption key.

A key identifier may or may not be associated with an encrypted rotation key. The process for determining the encrypted rotation key for a given key identifier and segment is equivalent to the process for determining `associated-aaxs-metadata` described in Section 4.3.5, with the following exceptions

- Replace all instances of the term "AAXS metadata" with "encrypted rotation key".
- Replace all instances of the term `associated-aaxs-metadata` with `associated-encrypted-rotation-key`

For a given key identifier and segment, the rotation key is the result of decrypting the associated encrypted rotation key. Otherwise, the rotation key is the AAXS content encryption key.

4.4. AAXS Examples

This section is non-normative.

4.4.1. Example AAXS `pssh` box

This example demonstrates the usage of the AAXS `pssh` box.

Suppose a `pssh` box provides the following values:

```
ProtectionSystemSpecificHeaderBox(version = 1)
{
    SystemID = 0xF239E769EFA348509C16A903C6932EFB // AAXS SystemID
    KID_count = 1
    KID = 0x00000001
    DataSize = ...
    Data = AdobeAccessMetadataBox(flags = 3)
          {
              KID_count = 2
              KID = 0x00000002
              encrypted_key = 0x22222222
              KID = 0x00000003
              encrypted_key = 0x33333333
              metadata_size = 8
              metadata = 0x12341234
          }
}
```

In this situation, we can deduce the following KID mapping:

KID	AAXS Metadata	Rotation Key
0x00000001	0x12341234	Unspecified. In the absence of other AAXS `pssh` boxes, this defaults to the AAXS metadata CEK.
0x00000002	0x12341234	Value obtained by decrypting 0x22222222 using AAXS metadata CEK.
0x00000003	0x12341234	Value obtained by decrypting 0x33333333 using AAXS metadata CEK.

4.4.2. Example of AAXS information embedded in segments

This example demonstrates how to embed AAXS information in segments.

In this example, assume that:

- `ExamplePresentation` is a presentation.
- `ExampleInitSegment` is the initialization segment for `ExampleRepresentation`.
- `ExampleMediaSegment` is a media segment of `ExampleRepresentation.`

Suppose that `ExampleInitSegment` contains a `pssh` box with the following values:

```
ProtectionSystemSpecificHeaderBox(version = 1)
{
    SystemID = 0xF239E769EFA348509C16A903C6932EFB // AAXS SystemID
    KID_count = 1
    KID = 0x00000001
    DataSize = ...
    Data = AdobeAccessMetadataBox(flags = 2)
          {
              KID_count = 1
              KID = 0x00000002
              metadata_size = ...
              metadata = 0xAAAAAAAA
          }
}
```

Suppose that `ExampleMediaSegment` contains a `pssh` box with the following values:

```
ProtectionSystemSpecificHeaderBox(version = 1)
{
    SystemID = 0xF239E769EFA348509C16A903C6932EFB // AAXS SystemID
    KID_count = 1
    KID = 0x00000003
    DataSize = ...
    Data = AdobeAccessMetadataBox(flags = 3)
          {
              KID_count = 2
              KID = 0x00000004
              encrypted_key = 0xBBBB4444
              KID = 0x00000005
              encrypted_key = 0xBBBB5555
              metadata_size = ...
              metadata = 0xBBBBBBBB
          }
}
```

In this situation, we can deduce the following KID mapping:

KID	AAXS Metadata	Rotation Key
0x00000001	0xAAAAAAAA **Reason:** 1. Skip `ExampleMediaSegment` since its `` `pssh` `` and `` `amet` `` don't include the KID. 2. Use `ExampleInitSegment` since its `` `pssh` `` includes the KID.	AAXS metadata CEK **Reason:** 1. Skip `ExampleMediaSegment` since its `` `amet` `` doesn't include the KID. 2. Skip `ExampleInitSegment` since its `` `amet` `` flags indicate that it does not include `encrypted_key`'s. 3. Default to AAXS metadata CEK .
0x00000002	0xAAAAAAAA **Reason:** 1. Skip `ExampleMediaSegment` since its `` `pssh` `` and `` `amet` `` don't include the KID. 2. Use `ExampleInitSegment` since its `` `amet` `` includes the KID.	AAXS metadata CEK **Reason:** Same as 0x00000001.
0x00000003	0xBBBBBBBB* **Reason:** Use `ExampleMediaSegment` since its `` `pssh` `` includes the KID.	Value obtained by decrypting 0xBBBB3333 using AAXS metadata CEK. **Reason:** Use `ExampleMediaSegment` since its `` `amet` `` includes the KID.
0x00000004	0xBBBBBBBB* **Reason:** Use `ExampleMediaSegment` since its `` `amet` `` includes the KID.	Value obtained by decrypting 0xBBBB4444 using AAXS metadata CEK. **Reason:** Same as 0x00000003.
0x00000005	0xBBBBBBBB **Reason:** Same as 0x00000004.	Value obtained by decrypting 0xBBBB5555 using AAXS metadata CEK. **Reason:** Same as 0x00000004.

In this example, the initialization and media segments both associate AAXS metadata with KID 0x0000003 and 0x00000004. This is atypical, but is included in this example to illustrate the relative precedence of AAXS information. Typically, the initialization and media segment will not associate different AAXS metadata with the same key.

4.4.3 Example of AAXS information in MPD

This example demonstrates how to embed AAXS information in the MPD.

In this example, assume that:

- `ExamplePresentation` is a presentation.
- `ExampleRepresentation` is a representation of `ExamplePresentation` that contains AAXS protected content.
- `ExampleMediaSegment` is a media segment of `ExampleRepresentation`. It does not contain an AAXS `pssh` box.
- `ExampleInitSegment` is the initialization segment for `ExampleRepresentation`. It does not contain an AAXS `pssh` box.

Suppose that `ExampleRepresentation`'s <`Representation`> contains the following AAXS <`ContentProtection`>:

```
<ContentProtection schemeIdUri="urn:uuid:F239E769-EFA3-4850-9C16-A903C6932EFB"
                   value="AdobeAccess 5.0"
                   cenc:default_KID="1"
                   access:default_IV_size="4"
                   access:default_IsEncrypted="1">
     <!-- the cenc:default_KID, access:default_IV_size, access:default_IsEncrypted
          must match the values specified in the `tenc` -->
     <access:pssh>
        ...
        <!-- the base64 representation of a `pssh` box with the following values:
        ProtectionSystemSpecificHeaderBox(version = 1)
        {
            SystemID = 0xF239E769EFA348509C16A903C6932EFB // AAXS SystemID
            KID_count = 1
            KID = 0x00000001
            DataSize = ...
            Data = AdobeAccessMetadataBox(flags = 2)
            {
               KID_count = 1
               KID = 0x00000002
               metadata_size = ...
               metadata = 0xCCCCCCCC
            }
        } -->
    </access:pssh>
</ContentProtection>
```

In this situation, we can deduce the following KID mapping:

KID	AAXS Metadata	Rotation Key
0x00000001	0xCCCCCCCC	AAXS metadata CEK **Reason:** 1. Skip <`access:pssh`> since its ` `amet` ` flags indicate that it does not include `encrypted_key` 's. 2. Default to AAXS metadata CEK .
0x00000002	0xCCCCCCCC	AAXS metadata CEK **Reason:** Same as 0x00000001.

Note: *This example will function identically if the MPD is modified to include <`cenc:pssh`>. If both <`access:pssh`> and <`cenc:pssh`> are present, the value of <`cenc:pssh`> must be identical to the body of <`access:pssh`>.*

4.4.4 Example of AAXS information in both the MPD and segments

This example demonstrates how AAXS information in the MPD interacts with AAXS information embedded within segments.

In this example, assume that:

- `ExamplePresentation` is a presentation.
- `ExampleRepresentation` is a representation of `ExamplePresentation` that contains AAXS protected content.
- `ExampleMediaSegment` is a media segment of `ExampleRepresentation`.

- `ExampleInitSegment` is the initialization segment for `ExampleRepresentation`.

Suppose that `ExampleInitSegment` contains a `` `pssh` `` box as described in Section 4.4.2.

Suppose that `ExampleMediaSegment` contains a `` `pssh` `` box as described in Section 4.4.2.

Suppose that `ExampleRepresentation`'s <`Representation`> contains an AAXS <`ContentProtection`> as described in Section 4.4.3.

In this situation, we can deduce the following KID mapping:

KID	AAXS Metadata	Rotation Key
0x00000001	0xCCCCCCCC **Reason:** 1. Skip `ExampleMediaSegment` since its `` `pssh` `` and `` `amet` `` don't include the KID. 2. Use <`access:pssh`> since its `` `pssh` `` includes the KID.	AAXS metadata CEK **Reason:** 1. Skip `ExampleMediaSegment` since its `` `amet` `` doesn't include the KID. 2. Skip <`access:pssh`> since its `` `amet` `` flags indicate that it does not include `encrypted_key`'s. 3. Always skip `ExampleInitSegment` (since <`access:pssh`> is present) 4. Default to AAXS metadata CEK .
0x00000002	0xCCCCCCCC **Reason:** 1. Skip `ExampleMediaSegment` since its `` `pssh` `` and `` `amet` `` don't include the KID. 2. Use <`access:pssh`> since its `` `amet` `` includes the KID.	AAXS metadata CEK **Reason:** Same as 0x00000001.
0x00000003, 0x00000004, 0x00000005	Same as Section 4.4.2 **Reason:** The new <`access:pssh`> do not include KID 0x00000003, 0x00000004, or 0x00000005.	Same as Section 4.4.2 **Reason:** The new <`access:pssh`> do not include KID 0x00000003, 0x00000004, or 0x00000005.

4.4.5 Example of AAXS information in both the MPD and segments with media segment override

This example demonstrates how AAXS information can be included in an <`access:pssh`> to take precedence over information provided by the media segments.

In this example, assume that:

- `ExamplePresentation` is a presentation.
- `ExampleRepresentation` is a representation of `ExamplePresentation` that contains AAXS protected content.
- `ExampleMediaSegment` is a media segment of `ExampleRepresentation`.
- `ExampleInitSegment` is the initialization segment for `ExampleRepresentation`.

Suppose that `ExampleInitSegment` contains a `` `pssh` `` box as described in Section 4.4.2.

Suppose that `ExampleMediaSegment` contains a `` `pssh` `` box as described in Section 4.4.2.

Suppose that `ExampleRepresentation`'s <`Representation`> includes the AAXS <`ContentProtection`> described in Section 4.4.3.

Suppose that `ExampleRepresentation`'s <`Representation`> also includes the <`access:pssh`> described below:

```
<access:pssh access:startTimestamp="4000">
    ...
    <!-- the base64 representation of a `pssh` box with the following values:
    ProtectionSystemSpecificHeaderBox(version = 1)
    {
        SystemID = 0xF239E769EFA348509C16A903C6932EFB // AAXS SystemID
        KID_count = 1
        KID = 0x00000003
        DataSize = ...
        Data = AdobeAccessMetadataBox(flags = 3)
        {
            KID_count = 1
            KID = 0x00000004
            encrypted_key = 0xDDDD4444
            metadata_size = ...
            metadata = 0xDDDDDDDD
        }
    } -->
</access:pssh>
<access:pssh access:startTimestamp="8000">
    ...
    <!-- the base64 representation of a `pssh` box with the following values:
    ProtectionSystemSpecificHeaderBox(version = 1)
    {
        SystemID = 0xF239E769EFA348509C16A903C6932EFB // AAXS SystemID
        KID_count = 1
        KID = 0x00000003
        DataSize = ...
        Data = AdobeAccessMetadataBox(flags = 3)
        {
            KID_count = 1
            KID = 0x00000004
            encrypted_key = 0xEEEE4444
            metadata_size = ...
            metadata = 0xEEEEEEEE
        }
    } -->
</access:pssh>
```

The start and end times of `ExampleMediaSegment` are 4000ms and 8000ms respectively.

In this situation, we can deduce the following KID mapping:

KID	AAXS Metadata	Rotation Key
0x00000001, 0x00000002, 0x00000005*	Same as Section 4.4.3 **Reason:** The new <`access:pssh`> do not include KID 0x00000001, 0x00000002, or 0x00000005.	Same as Section 4.4.3 **Reason:** The new <`access:pssh`> do not include KID 0x00000001, 0x00000002, or 0x00000005.
0x00000003	0xDDDDDDDD **Reason:** Use the <`access:pssh`> with `@access:startTimeStamp` of 4000 since its `` `pssh` `` includes the KID and it has an appropriate `@access:startTimeStamp`.	Value obtained by decrypting 0xBBBB3333 using AAXS metadata CEK. **Reason:** 1. Skip all <`access:pssh`> with `@access:startTimeStamp` since their `` `amet` `` does not includes the KID. 2. Use `ExampleMediaSegment` since its `` `amet` `` includes the KID.

0x00000004	0xDDDDDDDD **Reason:** Use the <`access:pssh`> with `@access:startTimeStamp` of 4000 since its `` `amet` `` includes the KID and it has an appropriate `@access:startTimeStamp`.	Value obtained by decrypting 0xDDDD4444 using AAXS metadata CEK. **Reason:** Use the <`access:pssh`> with `@access:startTimeStamp` of 4000 since its `` `amet` `` includes the KID and it has an appropriate `@access:startTimeStamp`.

***In this example, the media segment associates AAXS metadata with KID 0x0000005, while the MPD's <`access:pssh`> does not. This is atypical, but is included in this example to illustrate the relative precedence of AAXS information.*

4.4.6. Example AAXS `pssh` box serialization

This example demonstrates an example serialization of an Adobe `` `pssh` `` box.

ExamplePsshSerialization.pssh (from the example files) is the binary serialization of a single `` `pssh` `` box. Example files are available at the Adobe Primetime Technology Center. The following table describes the contents of ExamplePsshSerialization.pssh.

Offset	Length	Data	Description	References
0			`ProtectionSystemSpecificHeaderBox {`	Section 4.2.1, [CENC] 8.1
0	4	`00001324`	`size = 4900`	[ISOBMFF] 4
4	4	`70737368`	`type = 'pssh'`	[ISOBMFF] 4
8	2	`0000`	`version = 0`	[ISOBMFF] 4
10	2	`0000`	`flags = 0`	[ISOBMFF] 4
12	16	`F239E769 EFA34850 9C16A903 C6932EFB`	`SystemID`	Section 4.2.1, [CENC] 8.1
28	4	`00001304`	`DataSize = 4868`	[CENC] 8.1
32			`Data = {`	Section 4.2.1, [CENC] 8.1
32			`AdobeAccessMetadataBox {`	Section 4.2.2
32	4	`00001304`	`size = 4868`	[ISOBMFF] 4
36	4	`616D6574`	`type = 'amet'`	[ISOBMFF] 4
40	2	`0000`	`version = 0`	[ISOBMFF] 4
42	2	`0002`	`flags = 2`	Section 4.2.2, [ISOBMFF] 4
44	4	`00000001`	`KID_count = 1`	Section 4.2.2
48	16	`FC7A1ECD 0C9258AE FAD21C13 A69C8D65`	`KID[0]`	Section 4.2.2
64	4	`000012E0`	`metadata_size = 4832`	Section 4.2.2
68	4832	`308212DC … E62167E2`	`metadata`	Section 4.2.2
4900			`}`	
4900			`}`	

4900			}	

5. Document History

January 2016 FINAL

Editorial changes:

- 4.4.6: Added.
- 4.1.2, 6: Added reference to [AAXS-REF-IMPL].
- 4.2.2: Repeat informative information about AAXS metadata for clarity.
- Throughout: Replace instances of the term "AAXS DRM metadata" with "AAXS metadata" for consistency.

October 2015 FINAL

Normative changes:

- 4.2.2. New requirements for other flags bits: "The meaning of all bits of flags except the 1's or 2's bit is undefined. These bits are reserved for future use and their value SHOULD be zero."

Editorial changes:

- 3.7.2: Clarified relationship of Primetime DASH WebVTT segments and HTTP Live Streaming WebVTT segments
- 5: Added Document History
- 4.4.1: Addition text to clarify relationship of CENC/DASH-IF to Primetime DASH.
- 4.4.2: Clarify purpose of precedence process.
- Throughout: Add cross-references to examples
- 4.4: Expand and separate examples
- Formatting.
- 4.1.2: Clarify AAXS metadata.
- 4.2.2: Some rewording for clarity.

June 2015 FC DRAFT

First version

6. References

Normative References

[DPI]

Adobe Systems Incorporated, "Primetime Digital Program Insertion Signaling (DPI) Version 1.2", 2015.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, March 1997, https://www.ietf.org/rfc/rfc2119.txt.

[DASH]

"Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and delivery formats", ISO/IEC 23009-1:2014, May 2014, http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip.

[CENC]

"Information technology – MPEG systems technologies – Part 7: Common encryption in ISO base media file format files", ISO/IEC CD 23001-7 3rd Edition, 2014.

[DASH-IF]

DASH Industry Forum, "Guidelines for Implementation: DASH-IF Interoperability Points Version 3.0", April 2015, http://dashif.org/wp-conten t/uploads/2015/04/DASH-IF-IOP-v3.0.pdf .

[DASH-MSPR]

Microsoft, "DASH Content Protection using Microsoft PlayReady Version 1.2", October 2014, http://download.microsoft.com/download/7/7/6/7762455C-D254-4C84-BE17-16B0C60E31FD/MPEG%20DASH%20PlayReady%201.2%20-%202014-10-08.pdf.

[ISOBMFF]

"Information technology – Coding of audio-visual objects – Part 12: ISO base media file format", ISO/IEC 14496-12:2012, July 2015.

[WEBVTT]

"WebVTT: The Web Video Text Tracks Format", April 2015, http://dev.w3.org/html5/webvtt/.

[FCC]

Federal Communications Commission, "Electronic Code of Federal Regulations," May 2015, http://www.ecfr.gov/cgi-bin/retrieveECFR?gp=&SID=8b6798f33bd3185de2583e513a9b6ba4&r=PART&n=pt47.4.79#se47.4.79_1101.

[CEA608]

Consumer Electronics Association, "Line 21 Data Services," ANSI/CEA-608-ER-2014, April 2008.

Informative References

[AAXS-OVERVIEW]

Adobe Systems Incorporated, "Adobe Access Version 4.0: Overview", 2013, http://www.adobe.com/support/adobeaccess/pdfs/server/AdobeAccess_4_Overview.pdf.

[AAXS-REF-IMPL]

Adobe Systems Incorporated, "Using the Adobe Access Reference Implementations", 2014, http://www.adobe.com/support/adobeaccess/pdfs/server/AdobeAccess_4_RefImpl.pdf.

[CEA708]

Consumer Electronics Association, "Digital Television (DTV) Closed Captioning," ANSI/CEA-708-E, August 2013.

[HLS]

Pantos, R., "HTTP Live Streaming", draft-pantos-http-live-streaming-16, April 2015.