DASH-IF Implementation Guidelines: Token-based Access Control for DASH (TAC)

February 19th, 2019

DASH Industry Forum

Draft Version 1.0 (Final)

Scope

The scope of this document is to define a token-based access control mechanism and to enable the signaling of Authentication and Authorization (AA) protocols for DASH-based streaming. An Access Token is a proof that a DASH client or user of the client have been successfully authenticated and authorized in some pre-determined AA Systems to access a particular DASH resource, e.g. DASH segments or MPDs.

This document defines an Access Token format for accessing DASH resources and its transport between a DASH client and a server, hence ensuring interoperability between content providers and content delivery networks. The document focuses on the signaling and exchange mechanisms to facilitate Access Token-protected requests for the delivery of MPDs, licenses, keys and segments. This document can be used in addition to the general DASH-IF Interoperability Points.

Disclaimer

This is a document made available by DASH-IF. The technology embodied in this document may involve the use of intellectual property rights, including patents and patent applications owned or controlled by any of the authors or developers of this document. No patent license, either implied or express, is granted to you by this document. DASH-IF has made no search or investigation for such rights and DASH-IF disclaims any duty to do so. The rights and obligations which apply to DASH-IF documents, as such rights and obligations are set forth and defined in the DASH-IF Bylaws and IPR Policy including, but not limited to, patent and other intellectual property license rights and obligations. A copy of the DASH-IF Bylaws and IPR Policy can be obtained at http://dashif.org/.

The material contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, this material is provided AS IS, and the authors and developers of this material and DASH-IF hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of respons-es, of workmanlike effort, and of lack of negligence.

In addition, this document may include references to documents and/or technologies controlled by third parties. Those third party documents and technologies may be subject to third party rules and licensing terms. No intellectual property license, either implied or express, to any third party material is granted to you by this document or DASH-IF. DASH-IF makes no any warranty whatsoever for such third party material.

Note that technologies included in this document and for which no test and conformance material is provided, are only published as a candidate technologies, and may be removed if no test material is provided before releasing a new version of this guidelines document. For the availability of test material, please check http://www.dashif.org.

If you have comments on the document, please submit comments at the following URL:

- at the github repository https://github.com/Dash-Industry-Forum/TAC/issues
- at the public repository https://gitreports.com/issue/Dash-Industry-Forum/TAC

.

Contents

1 Introduction

1.1 General

Common DASH use cases may require authentication of the end user or their player/device, followed by authorization to access the content described in an MPD. The authentication and authorization operations are commonly performed by Authentication and Authorization (AA) systems. Because authorization depends on authentication, the two functions are usually performed sequentially starting with authentication and then authorization.

It is also important to distinguish AA systems from DRM systems. DRM systems provide technical means to securely deliver keys, wrapped in licenses, necessary for the decryption of encrypted content while AA systems protect the access to a resource. The question whether the resource is itself encrypted is irrelevant for an AA system. Therefore, Authentication and Authorization systems and DRM systems are orthogonal and can be used in conjunction. As a matter of fact, the delivery of DRM licenses and keys are protected by an Authentication and Authorization system. Furthermore, an AA system does not require a DRM system and an AA system may simply protect the access to an unencrypted content.

Example scenarios where authorization is useful, possibly with additional mechanisms, are:

- Streaming is restricted to a geographic region where the service provider has distribution rights.
- Streaming is restricted to DASH clients that present ads in the video, MPD, or signaled for insertion; and accurately report playback.
- Streaming is restricted to end users who are subscribers or have purchased rental/ownership for streaming/download of SD/HD/UHD quality, for a particular date range or number of views, for particular devices and protection systems, for a maximum number of devices or simultaneous streams, etc.
- Streaming is enabled using federated identity systems such as TV Everywhere, UltraViolet, OpenID, and various SSO systems (Single Sign On), and federated rights systems such as DECE, KeyChest, TV Everywhere, etc.

This document defines the signaling, the exchange mechanism of Access Tokens and the Access Token format for granting the DASH clients access to DASH resources, e.g. DASH segments, MPDs, etc. Typically, AA information takes the form of AA Tokens as proofs that the clients or the users of the clients have been authenticated and authorized according to some pre-determined AA systems (or schemes). Essentially, the goal of this document is to enable interoperability of the resource access control between DASH clients and servers delivering DASH content while allowing the choice of the enforcement rules by the content provider.

The signaling and the exchange mechanisms defined in this document leverages on the AA signaling and information exchanging mechanisms defined in [DASH-AMD3].

1.2 References

1.2.1 Normative References

[DASH] ISO/IEC 23009-2:2014 Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats.

[DASH-AMD3] ISO/IEC 23009-1:2014/Amd 3:2016, Authentication, MPD linking, Callback Event, Period Continuity and other Extensions, October 2016.

[URISigning] URI Signing for CDN Interconnection (CDNI), R. van Brandenburg, K. Leung, P. Sorber, October 23, 2018, https://tools.ietf.org/html/draft-ietf-cdni-uri-signing-16.

[RFC-7519] JSON Web Token (JWT), M. Jones, J. Bradley, N. Sakimura, May 2015.

1.2.2 Informative References

[RFC-7230] Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, R. Fielding, J. Reschke, June 2014.

[DASH-IOP] Guidelines for Implementation: DASH-IF Interoperability Points, December 2016.

[OMAP] Online Multimedia Authorization Protocol. Open Authentication Technology Committee. Version 1.0, August 22, 2012.

[OAuth] RFC 6749, The OAuth 2.0 Authorization Framework, D. Hardt, Ed., October 2012.

[CPA] ETSI TS 103 407: Cross Platform Authentication for limited input hybrid consumer equipment, https://tech.ebu.ch/cpa.

1.3 Terms & Definitions

AA Token

authentication token or authorization token

Access Token

token granting, if valid, the access to a resource identified by an URL

Authentication

process of determining whether a user or client is who or what it is claims to be or not. This may rely on some type(s) of user and/or client identification and credentials

Authentication Token

token as a proof of being authenticated

Authorization

process of determining whether a user or client has permissions to access content or not. This may rely on authenticating the user or client

Authorization Token

token as a proof of being authorized

Content

One or more audio-visual elementary streams and the associated MDP if in DASH format

Token

form of credentials or proof, used for some purpose, encoded digitally as a string, possibly with opaque parts

Policy Enforcement Element

parameters implementing rules limiting the validity of a Token

Signature

a data structure cryptographically securing that the Policy Enforcement Elements are authentic, complete and not tampered

2 Authorization and Authentication Use Cases for DASH Resource Access Control

2.1 Introduction

Authentication and Authorization information needs to be exchanged between DASH system entities, typically servers and clients, in order to allow DASH clients to access protected content. Typically, the AA information takes the form of AA Tokens. Upon reception of valid AA Tokens, the client obtains an Access Token. Consequently, when requesting segments or MPDs, the client provides the Access Token along with the requests. When the Access Token is valid, the server grants access to the content and deliver the resource to the client. Note that the AA Systems and workflows for generating these Access Tokens may vary and depends on several factors, such as available client identification and credentials, server requirements and security measures as well as the type and quality of content that can be On Demand or Live, and SD, HD or UHD.

This section presents different use cases of Access Token generations and exchanges granting access to DASH resources. For the generic use case of HTTP resource protected by a token system see Annex A.

The Access Tokens scenarios can be characterized by the following aspects: the issuer, the receiver, the validity period, the scope, the transport. The following use cases reflect different variations of the Access Token scenarios.

2.2 Mandatory Pre-roll

A service provider offers MPDs that contain two Periods, i.e. one for a pre-roll advertisement and one for the main content. The service provider requires that the DASH clients first play the pre-roll advertisement before being able to retrieve the main content. The end users must first contact the ad server before viewing the main content. To this end, an Access Token protects the delivery of the main content. Only the segment requests providing a valid Access Token are granted. In addition, the service provider configured the HTTP server delivering the advertisement to insert the segment Access Token for the main content in the HTTP response of the advertisement segment. A possible flow is shown in Figure 1.

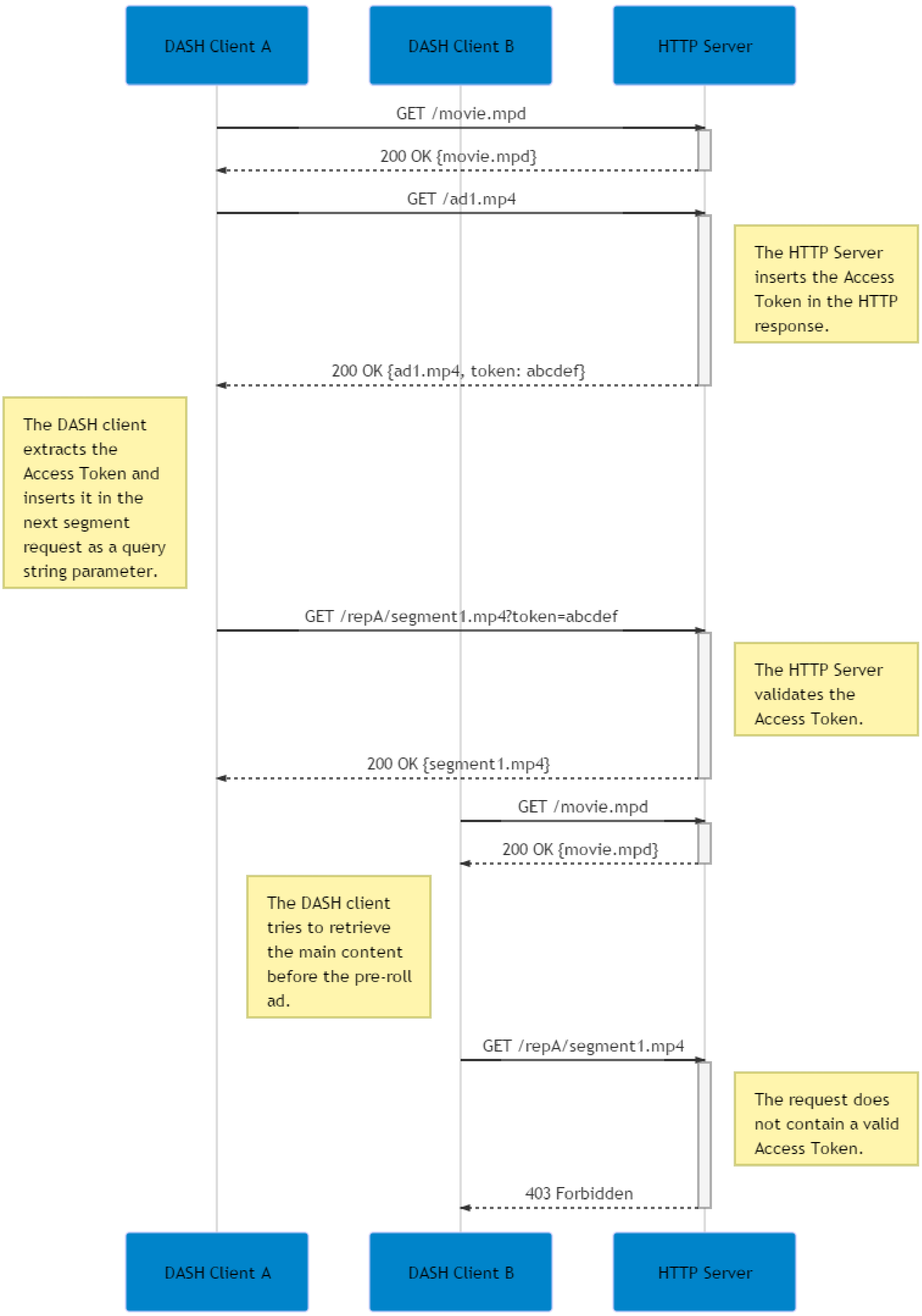
DASH Client A
DASH Client B
HTTP Server
GET /movie.mpd
200 OK {movie.mpd}
GET /ad1.mp4
The HTTP Server inserts the Access Token in the HTTP response.
200 OK {ad1.mp4, token: abcdef}
The DASH client extracts the Access Token and inserts it in the next segment request as a query string parameter.
GET /repA/segment1.mp4?token=abcdef
The HTTP Server validates the Access Token.
200 OK {segment1.mp4}
GET /movie.mpd
200 OK {movie.mpd}
The DASH client tries to retrieve the main content before the pre-roll ad.
GET /repA/segment1.mp4
The request does not contain a valid Access Token.
403 Forbidden
DASH Client A
DASH Client B
HTTP Server

Figure 1 - Ad pre-roll download sequence

2.3 Ad Free Premium Service

In this use case, the same service provider, as in use case 2.2, wants to offer the possibility to pay a monthly fee to be able to skip the pre-roll advertisement. The end user must authenticate himself/herself via the portal and subscribe to the ad free feature. Upon MPD download, the MPD server provides the MPD along with the segment Access Token generated by the Authorization Server shortly before. Note that, in this case, the MPD only contains the main video without the advertisement period as in the previous use case. A possible flow is shown in Figure 2.

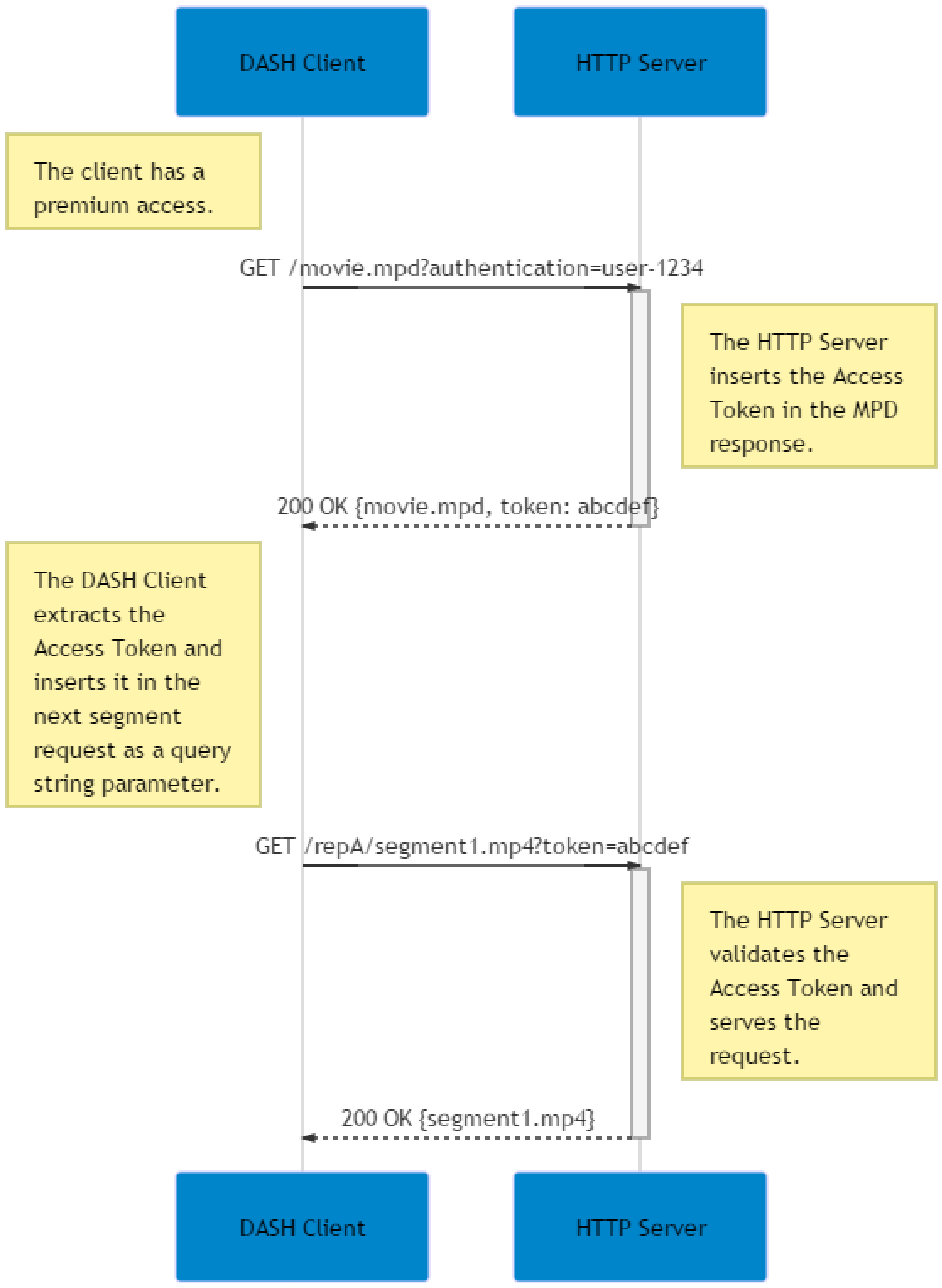
DASH Client
HTTP Server
The client has a premium access.
GET /movie.mpd?authentication=user-1234
The HTTP Server inserts the Access Token in the MPD response.
200 OK {movie.mpd, token: abcdef}
The DASH Client extracts the Access Token and inserts it in the next segment request as a query string parameter.
GET /repA/segment1.mp4?token=abcdef
The HTTP Server validates the Access Token and serves the request.
200 OK {segment1.mp4}
DASH Client
HTTP Server

Figure 2 - Ad free viewing for premium client sequence

2.4 Service Provider Using CDNs

A service provider wants to manage its movie catalog but outsources the delivery of the MPD and the segments to a CDN provider. In this case, the CDN has no business logic to decide whether a DASH client requesting an MPD or a segment is authorized to retrieve the content.

However, a CDN may verify whether a token is valid using signed information, e.g. the expiration time, client IP address, etc. The end user first purchases a movie via the portal of the service provider. Upon success, the portal gives the MPD URL as well as a short-term Access Token to the application. This Access Token is a short-term token, i.e. it expires couples of seconds after its generation. It mitigates the risk of unauthorized clients using the token after its generation. Consequently, the application sends a MPD request to the CDN along with the short-term Access Token. The CDN verifies its validity and delivers a new Access Token with a new validity time inside the MPD response for the DASH client to access the segments. This Access Token may have a longer validity period compared to the one for the MPD. Every time the DASH client requests a segment it uses the latest received Access Token. Upon segment requests, the CDN returns a refreshed Access Token to extend the chain of tokens. More details on this mechanism are provided in paragraph 6.1. A possible flow is shown in Figure 3.

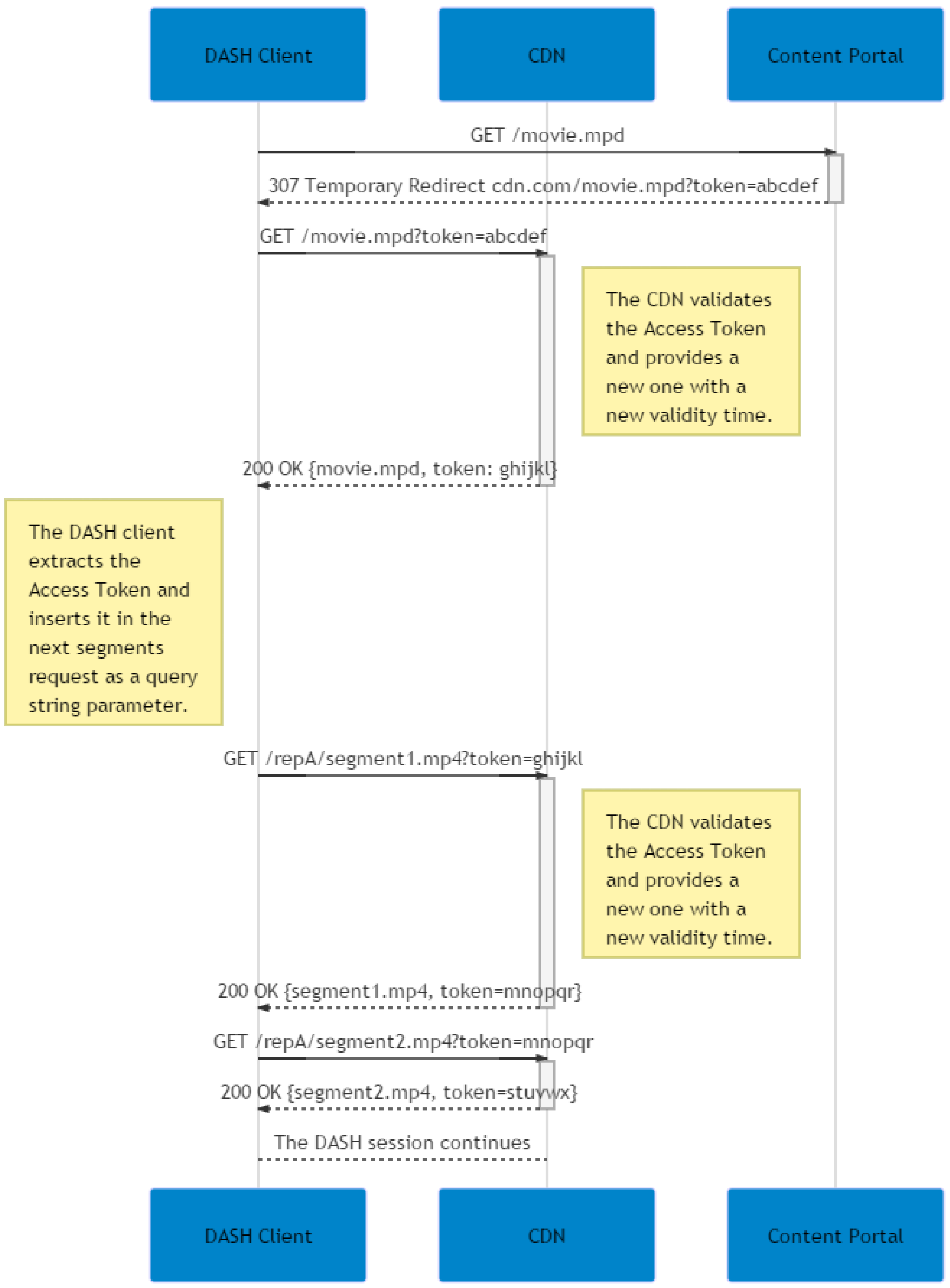
DASH Client
CDN
Content Portal
GET /movie.mpd
307 Temporary Redirect cdn.com/movie.mpd?token=abcdef
GET /movie.mpd?token=abcdef
The CDN validates the Access Token and provides a new one with a new validity time.
200 OK {movie.mpd, token: ghijkl}
The DASH client extracts the Access Token and inserts it in the next segments request as a query string parameter.
GET /repA/segment1.mp4?token=ghijkl
The CDN validates the Access Token and provides a new one with a new validity time.
200 OK {segment1.mp4, token=mnopqr}
GET /repA/segment2.mp4?token=mnopqr
200 OK {segment2.mp4, token=stuvwx}
The DASH session continues
DASH Client
CDN
Content Portal

Figure 3 - Segment access token sequence with CDN integration

2.5 DRM License Retrieval Protection

A DRM system provides an API to download a license embedding content keys for decrypting DASH content. This API can be used only by authorized DASH client. These DASH clients get an Access Token to request a license. To this end, the HTTP server hosting content key is configured to require the presence of a valid Access Token in the request for a license.

Although the retrieval of license key is not signaled within the MPD, a workflow similar to Media Segments and MPD retrieval can be applied to the protection of license key acquisition.

For this workflow, it assumed that the license is identified by the following URL https://drm.com/license.key and that the Authentication Server of the DRM system is exposed at https://drm.com/authenticate. Note though that the protocol to authenticate the user is out-of-scope.

A possible flow is shown in Figure 4.

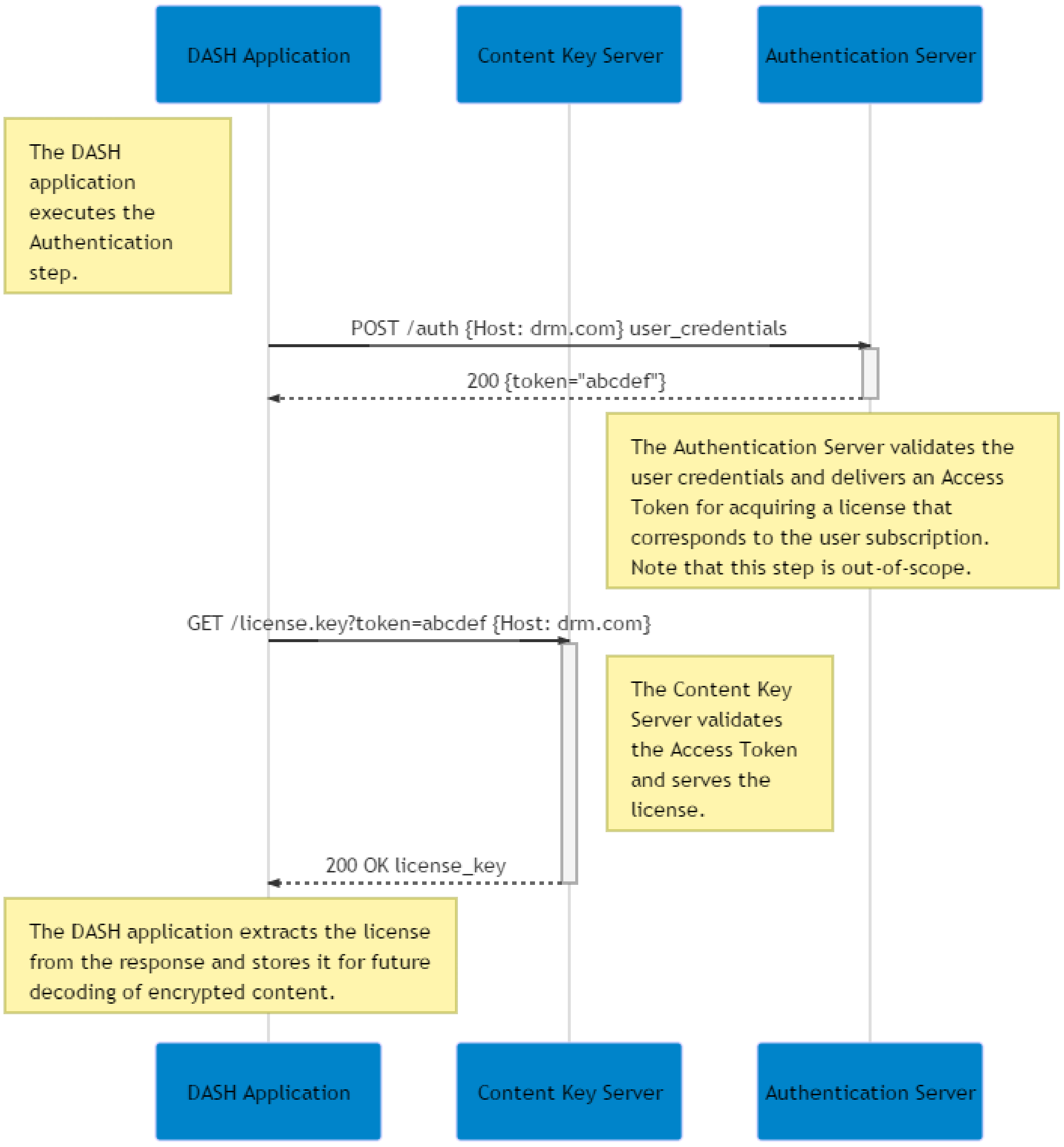
DASH Application
Content Key Server
Authentication Server
The DASH application executes the Authentication step.
POST /auth {Host: drm.com} user_credentials
200 {token="abcdef"}
The Authentication Server validates the user credentials and delivers an Access Token for acquiring a license that corresponds to the user subscription. Note that this step is out-of-scope.
GET /license.key?token=abcdef {Host: drm.com}
The Content Key Server validates the Access Token and serves the license.
200 OK license_key
The DASH application extracts the license from the response and stores it for future decoding of encrypted content.
DASH Application
Content Key Server
Authentication Server

Figure 4 - Segment access token sequence with DRM System

3 Access Token Format

3.1 Introduction

[URISigning] defines a token format for protecting the access to any resource identified by a URL. The token format relies on a signed JSON Web Token (JWT) [RFC-7519] profile. Additionally, [URISigning] specifies new JWT claims to support the use case of token-based protection for segmented content, especially for adaptive streaming scenarios. More information, can be found in Annex B.

This section specifies the format and the transport encoding of the Access Token.

3.2 Format

The Access Token shall be formatted as defined in Section 2 of [URISigning]. In addition, the CDNI Signed Token Transport (`cdnistt`) claim value of the Access Token shall be set the value of 2.

The [URISigning] specification provides the ability to parametrize the transport mechanism via this JWT `cdnistt` claim.

The value "2" is registered as the "DASH-IF Token Transport"

Editor's note: The CDNI working group has not yet created the registry. The actual value might change upon registration.

An overview of the JWT claims defined in [URISigning] is provided in Annex B for information.

3.3 Transport Encoding

The Access Token shall be encoded as a signed JWT as defined in [URISigning].

4 Transport Mechanism for DASH

4.1 Introduction

Using cookies (`cdnistt` value 1 in [URISigning]) to communicate tokens in general can be problematic for the following reasons:

- Some embedded devices do not use the same User Agent to get the MPD and the segment, resulting in cookie not found.
- When content providers use multiple CDNs or deliver MPDs from a domain other than the segments, cookies are not delivered to the CDN since cross-domains cookie is not supported.
- Cookie support across browsers varies to such a degree that even across versions of the same browser it may differ significantly

Therefore, the transport of the Access Tokens specified in this document does not rely on cookies but relies on HTTP header extensions and on query string parameters as specified in this paragraph.

4.2 Access Token Transport over HTTP

The Access Token shall be transported in a HTTP header field when communicated in a HTTP 2xx Successful message.

The following ABNF syntax for the header field shall be used:

```
DASH-header-field = "DASH-IF-IETF-Token" ":" token

token = access-token
```

The field `access-token` contains an Access Token whose format is defined in paragraph 3.2 and encoded as defined in paragraph 3.3.

The Access Token shall be transported in a query string parameter when communicated in a HTTP 3xx Redirection message or in a HTTP request.

The query string parameter name shall be "`dash-if-ietf-token`" and shall contain an Access Token whose format is defined in paragraph 3.2.

NOTE – Although HTTP header names are case insensitive, the typography used above helps the distinction between the HTTP header name and the query string parameter name.

NOTE – Future token format may be added a later point in time by defining new combinations of HTTP custom header name and query string parameter names.

5 Access Token Exchange Protocol and Signalization for DASH

5.1 Introduction

This section specifies the signaling of the protocol to acquire and exchange an Access Token. It also specifies the successive steps for the acquisition and use of an Access Token for some specific protocols.

The specification enables three types of mechanisms for a DASH client to retrieve the initial Access Token. The MPD author may choose one of these three types.

- The first mechanism instantiates the Access Token request from regular DASH operations, e.g. MPD request, segments requests, Xlink resolution etc. In practice, the HTTP server delivers the Access Token along with the requested resource. The MPD author signals in the MPD how and where to extract the Access Token. This is a HTTP-based exchange protocol. More details are provided in paragraph 5.2.
- The second mechanism instantiates the Access Token request from an appropriate XML element in the MPD. In some situations, the MPD may not be delivered via HTTP or the MPD author wishes to embed the initial Access Token within the MPD itself. This is a MPD-based exchange protocol. More details are provided in paragraph 5.3.
- The last mechanism instantiates the Access Token request via an external protocol. In this case, the MPD signals the protocol to be used by the application to retrieve the Access Token. When the application recognizes one of the signaled protocols, it executes the corresponding protocol as specified by the scheme. Since out-of-scope of this specification, it is expected that the application implements the different steps of the protocol that leads to the acquisition of the Access Token. This is an external-based exchange protocol. More details are provided in paragraph 5.4.

NOTE – In both first and second mechanisms, the DASH client is agnostic as to what the nature and content of the Access Token are. The DASH client merely sees the Access Token as an opaque string that needs to passed along with future HTTP requests.

5.2 HTTP-based Access Token Usage

For this type of Access Token acquisition, the Access Token is delivered to the DASH client via regular DASH operations following the HTTP transport mechanisms specified in paragraph 4.

The Access Token may be delivered in the header of a MPD response, a segment response, a xlink response, etc... Consequently, the MPD author needs to instruct the DASH client where to extract the Access Token from. To this end, the **`ExtUrlQueryInfo,`** defined in I.3 of [DASH-AMD3], shall be used and should be configured according to the mechanism expected by the MPD author.

The example MPD below signals the presence of the Access Token in the HTTP responses for MPD requests (`@headerParamSource` attribute) inside the HTTP custom header called "DASH-IF-IETF-Token" and instructs the DASH client to insert this Access Token into segment and MPD requests within the "dash-if-ietf-token" query string parameter.

```
<AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1" maxWidth="1280"
maxHeight="720" maxFrameRate="25" par="16:9">
  <EssentialProperty schemeIdUri="urn:mpeg:dash:urlparam:2016:querystring"
xmlns:up="urn:mpeg:dash:schema:urlparam:2016">
    <up:ExtUrlQueryInfo
        headerParamSource="mpd"
        includeInRequests="segment mpd"
        queryTemplate="dash-if-ietf-token=$header:DASH-IF-IETF-Token$"/>
  </EssentialProperty>
  <SegmentTemplate duration="2" startNumber="1" media="seg$Number$.mp4">
  </SegmentTemplate>
    <Representation id="v0" codecs="avc3.4d401f" width="1280" height="720" frameRate="25"
sar="1:1" bandwidth="3000000"/>
</AdaptationSet>
```

In the example above, the DASH client is expected to perform the following steps according to [DASH-AMD3] when sending HTTP requests for MPD and segments.

Step	Action	URL construction
1	Determine the base URL of the resource	http://cdn.com/movie/seg1.mp4
2	Insert the query string part in the base URL using the `@queryTemplate` attribute	http://cdn.com/movie/seg1.mp4?dash-if-ietf-token=$header:DASH-IF-IETF-Token$

3	Determine the most recent value of the HTTP header 'Access-Token' attribute received in a MPD response.	`HTTP/1.1 200`
		`DASH-IF-IETF-Token: rtziwO2HwPfWw~yYD`
		
		`<MPD>`
		…
		`</MPD>`

4	Substitute the expression in the query template	http://cdn.com/movie/seg1.mp4?dash-if-ietf-token=rtziwO2HwPfWw~yYD

NOTE – The body of the HTTP response can be cached by the CDN edge in this case since the MPD is not client specific. Only the HTTP header value for the Access Token is computed on-the-fly which is customary for many of the HTTP headers in HTTP responses.

5.3 MPD-based Access Token Usage

The Access Token may be inserted in the MPD.

The `@queryString` attribute of either the **`UrlQueryInfo`** or **`ExtUrlQueryInfo`** descriptors shall be used to carry the Access Tokens.

Below is an example using the **`ExtUrlQueryInfo`** descriptor:

```
<EssentialProperty schemeIdUri="urn:mpeg:dash:urlparam:2016:querystring"
xmlns:up="urn:mpeg:dash:schema:urlparam:2016">
    <up:ExtUrlQueryInfo
        includeInRequests="mpd segment"
        queryString="token=nitfHRCrtziwO2HwPfWw~yYD"
        queryTemplate="dash-if-ietf-token=$query:token$"/>
  </EssentialProperty>
</EssentialProperty>
```

NOTE – The query string parameter in the `@queryString` attribute is not normatively defined. The MPD author may choose the query string parameter name it wishes.

When using the **`UrlQueryInfo`** descriptor, the echoing mechanism described in paragraph 5.2 is not possible.

This **`EssentialProperty`** should be located in the appropriate level in the MPD, for instance in a Representation, an Adaptation Set, etc... In the example above, the DASH client is expected to perform the following steps according to [DASH-AMD3] when sending HTTP requests for MPD and segments.

Step	Action	URL construction
1	Determine the base URL of the resource	http://cdn.com/movie/seg1.mp4
2	Insert the query string part in the base URL using the `@queryTemplate` attribute	http://cdn.com/movie/seg1.mp4?dash-if-ietf-token=$query:token$
3	Substitute the expression in the query template, i.e. extracting the value of the parameter access-token in the `@queryString` attribute	http://cdn.com/movie/seg1.mp4?dash-if-ietf-token=nitfHRCrtziwO2HwPfWw~yYD
4	Send the HTTP request with the final URL	GET http://cdn.com/movie/seg1.mp4?dash-if-ietf-token=nitfHRCrtziwO2HwPfWw~yYD$

NOTE – The MPD is in this case personalized for a given DASH client. As a result, the generated MPD cannot be cached for serving different DASH client requests.

5.4 External Protocol Access Token Usage

There are possible AA protocols that can be used as external protocols. However, their integration has not been verified and may require non-interoperable measures to implement them.

These protocols are:

- Multimedia Authorization Protocol [OMAP]
- The OAuth 2.0 Authorization Framework [OAuth]
- EBU's Cross-Platform Authentication [CPA]

The external instantiation of Access Tokens comprises the following additional aspect compared to the protocol specified in 3 Access Token Format :

- Use of the Authentication and Authorization descriptors in [DASH-AMD3],
  - o to signal any AA schemes and possibly parameters needed to obtain Access Tokens, and
  - o to carry any available Access Tokens or to signal any URLs where Access Tokens can be retrieved.

The table below constitutes the only normative aspect introduced by this section.

Table 1 - Parameter identifier for substitution in query string template

$<Identifier>$	**Substitution parameter**
`$AASchemeIdUri$`	The identifier shall be substituted by the scheme identifier of the **`EssentialProperty`** whose attribute `@id` value was `"mpeg:dash:content-authorization:2015"` that was used to obtain the Access Token.
`$AccessToken$`	The identifier shall be substituted by the Access Token obtained via an AA System.

The Client Authentication and Content Authorization descriptors in [DASH-AMD3] provide a mechanism to signal identification information of AA Systems. The following examples illustrate how client authentication and content access authorization information is signaled in the MPD.

```
<EssentialProperty schemeIdUri="urn:org:example:plan-a"
id="mpeg:dash:client-authentication:2015"
value="http://domain.com/authenticationServerA/protocolA?=ServiceSpecificInfoA"/>

<EssentialProperty schemeIdUri="urn:org:example:plan-b"
  id="mpeg:dash:client-authentication:2015"

value="http://domain.com/authenticationServerA/protocolB?=ServiceSpecificInfoB"/>

<EssentialProperty schemeIdUri="urn:org:example:plan-c"
   id="mpeg:dash:content-authorization:2015"

value="http://domain.com/authorizationServerC/protocolC?=ContentSpecificInfoC"/>

<EssentialProperty schemeIdUri="urn:org:example:plan-d"
   id="mpeg:dash:content-authorization:2015"

value="http://domain.com/authorizationServerD/protocolD?=ContentSpecificInfoD"/>
```

The application on top of the DASH client is expected, in the nominal case, to perform the following actions.

Step	**Action**
1	Find the supported Client Authentication descriptors if any
2	Execute one of the supported Client Authentication protocol whose endpoint is signaled in the `@value` attribute
3	Retrieve the Authentication Token as proof of a successful client authentication
4	Find the supported Content Authorization descriptors if any
5	Execute one of the supported Content Authorization protocols whose endpoint is signaled in the `@value` attribute. It may require to provide the Authentication Token from step 3, if present, depending on the actual protocol.
6	Receive the Access Token

The **`UrlQueryInfo`** provides the DASH client with the query string to append in the segment URL:

```
<AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1" maxWidth="1280"
maxHeight="720" maxFrameRate="25" par="16:9">

  <EssentialProperty schemeIdUri="urn:mpeg:dash:urlparam:2014"
xmlns:up="urn:mpeg:dash:schema:urlparam:2014">
    <up:UrlQueryInfo queryTemplate="system=$AASchemeIdUri$&token=$AccessToken$"/>
  </EssentialProperty>

  <SegmentTemplate duration="2" startNumber="1" media="seg$Number$.mp4"/>

  <Representation id="v1" codecs="avc3.4d401f" width="640" height="360" frameRate="25"
sar="1:1" bandwidth="1500000"/>

</AdaptationSet>
```

NOTE – Since this instantiation use external protocols and token formats, there is neither normative HTTP header name nor query string names to use. This is up to the MPD author to choose them.

The DASH client is then expected to perform the following steps:

Step	**Action**	**URL construction**
1	Determine the URL of the segment	http://cdn.com/movie/seg1.mp4
2	Insert the query string part in the requested URL according to the `@queryTemplate` attribute	http://cdn.com/movie/seg1.mp4?system=$AASchemeIdUri$&token=$AccessToken$
3	Substitute `$AASchemeIdUri$` and `$AccessToken$` with the AA scheme used and the Access Token retrieved from it.	http://cdn.com/movie/seg1.mp4?system=urn:org:example:plan-c&token=PfWw~yYD
4	Send the HTTP request with the final URL	GET http://cdn.com/movie/seg1.mp4?system=urn:org:example:plan-c&token=PfWw~yYD

6 Deployment and Security Considerations

6.1 Access Token Refresh

In some cases, the Access Token may expire while the user may want to continue the playback of the content, for instance when the user paused the playback for a certain amount of time.

For enabling the refresh of the Access Token, the descriptor **`ExtUrlQueryInfo`** must be used in combination with the instantiation described in paragraph 5.2. As specified by Table I.4 [DASH-AMD3], the DASH client shall insert in the query string "the latest received value of the header-name HTTP header in the HTTP responses indicated by the `@headerParamSource` attribute".

This way, an Authorization Server located on the HTTP server serving the Media Segments can send a refreshed Access Token upon reception of a segment request containing a valid Access Token. See Section 3 of [URISigning] for more details on generating refreshed Access Tokens.

When MPD update mechanism is used, the DASH client regularly sends an HTTP request at the MPD location. In addition, it is possible to protect the MPD and the Media Segments with the same Access Token using an appropriate path pattern value (see URI Pattern Container (UPC) in Annex B.2). This way, it is also possible to refresh the Access Token every time the DASH client fetches a new MPD version. For this to work, the validity period of the Access Token should be set to a value greater or equal to the period between two MPD are fetched by the DASH clients.

6.2 Query string length

Although not bounded in length by specifications, query strings may have in practice a maximum length in web browser implementations and HTTP server configuration. As a result, one should take care of verifying that the HTTP server handling the Access Token validation and generation are properly configured to handle such long query strings. Nevertheless, it is already very common for many web applications or even streaming services to use long query strings to convey additional information. The transport of the Access Token does constitute a higher requirement in that respect.

6.3 Safe Delivery of the Access Token

As any sensitive data, care should be taken to the delivery of the Access Token. There are mainly two risks when the Access Token travels between an HTTP server and a DASH client. The first risk is that an intermediate entity seating on the delivery path intercepts the HTTP request or response and extracts the Access Token. This is also known as man-in-the-middle attack. When the Access Token is intercepted, the CDN is subject to replay attack, see 6.4 for counter-measures. The second risk is the loss of the Access Token in HTTP headers when HTTP responses are badly proxied on the way to the DASH client. Although rare and seen as bad implementation of the proxy according to [RFC7230] (Section 3.2.1), there is a chance that the Access Token header gets stripped off and never reaches the DASH client. For both preventing the interception and the loss of HTTP header, TLS over HTTP can be used to deliver the DASH resource (Media Segments or MPD) along with the Access Token. TLS protocol prevents attack of this nature by encrypting end-to-end the information exchanged between the HTTP server and the DASH client, including HTTP headers and query string parameters that hold the Access Token. Note that this is the same recommendation as proposed by [DASH-IOP] in paragraph 3.4.4 Transforming Proxies and Other Adaptation Middleboxes when dealing with proxies.

6.4 Measures Against Replay Attack

Assuming the transport of the Access Token prevents the leakage of the Access Token (see paragraph 6.3), there may be rogue DASH clients that share their Access Token with illegitimate DASH clients. To mitigate this risk, these following measures can be applied:

- Validating the incoming request against client IP and information from the header attributes.
- Keep the validity period of the Access Token short by setting the Not Before (`nbf`) and the Expiry Time (`exp`) JWT claims around the estimated time of the client request.
- Enforce a one-time use of the Access Token via the Nonce (`jti)` JWT claim.

These measures are explained in greater details in paragraph 7 of [URISigning].

A Annex A – Overview of generic token-based access control concept (informative)

A.1 Token Concept and Definition

An Access Token is a proof of one or more past actions granting access to a resource. Figure 5 shows the concept of accessing a protected resource with a token.

A client is sending an HTTP request to an HTTP server. If the HTTP request does not contain a valid token or any token at all, the HTTP server does not serve the request. On the contrary, if a valid token is provided in the HTTP request, the HTTP server delivers the requested resource. In the context of DASH, the resource can be for instance MPDs or media segments.

A token can be characterized by two main aspects:

- The required action to obtain the token
- The rules that determine the validity of the token

When the action to obtain a token involves authenticating a client, this token is usually called Authentication Token. Similarly, when the action to obtain a token involves authorizing a client, this token is usually called Authorization Token. Note also that these two steps are commonly chained, i.e. a client needs to provide an Authentication Token before requesting an Authorization Token.

Regarding the validity rules of a token, they typically involve a validity time, a resource identifier, e.g. an URL, an IP address of the allowed clients, etc. Nevertheless, the DASH client is agnostic as to what the token holds as information and merely sees it as an opaque string. Only the entity issuing the token and the one validating it must understand the token format.

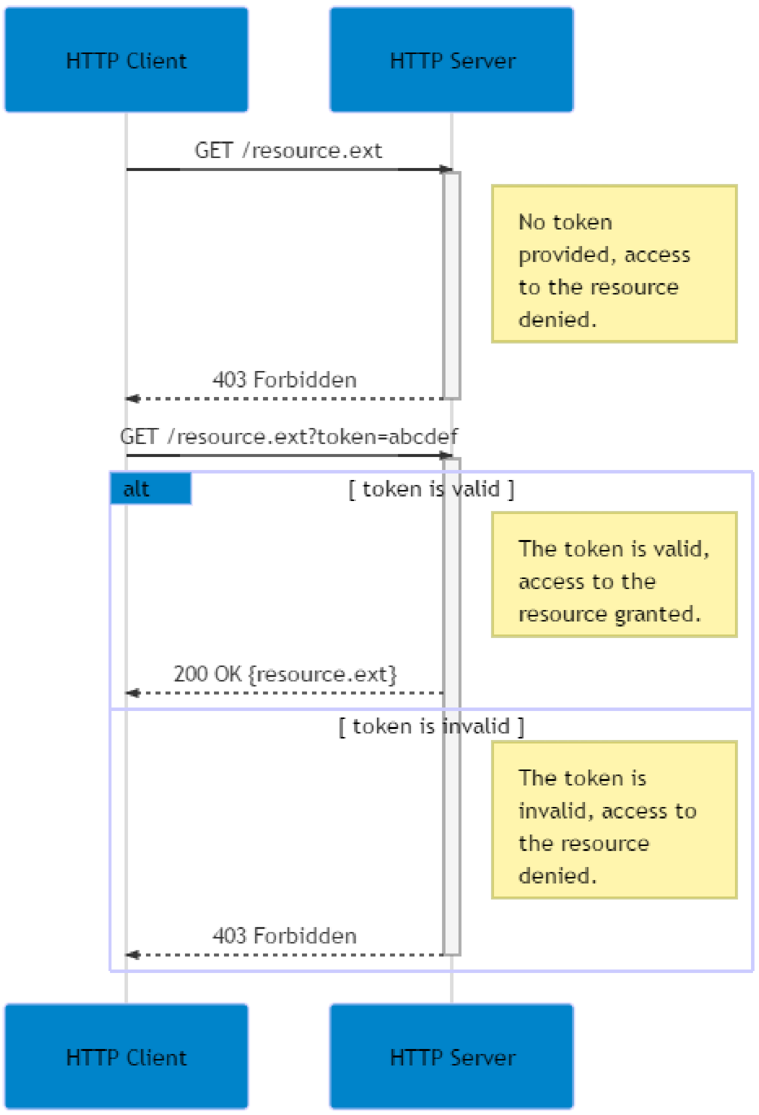
HTTP Client
HTTP Server
GET /resource.ext
No token provided, access to the resource denied.
403 Forbidden
GET /resource.ext?token=abcdef
alt
[token is valid]
The token is valid, access to the resource granted.
200 OK {resource.ext}
[token is invalid]
The token is invalid, access to the resource denied.
403 Forbidden
HTTP Client
HTTP Server

Figure 5 - Token-protected resource retrieval

A.2 Overview of AA System Architecture

This informative section gives an overview of the functions involved in AA System. The DASH client and/or the application around is able to request and retrieve a token that grants him access to the requested resource such as segments, MPD, etc.

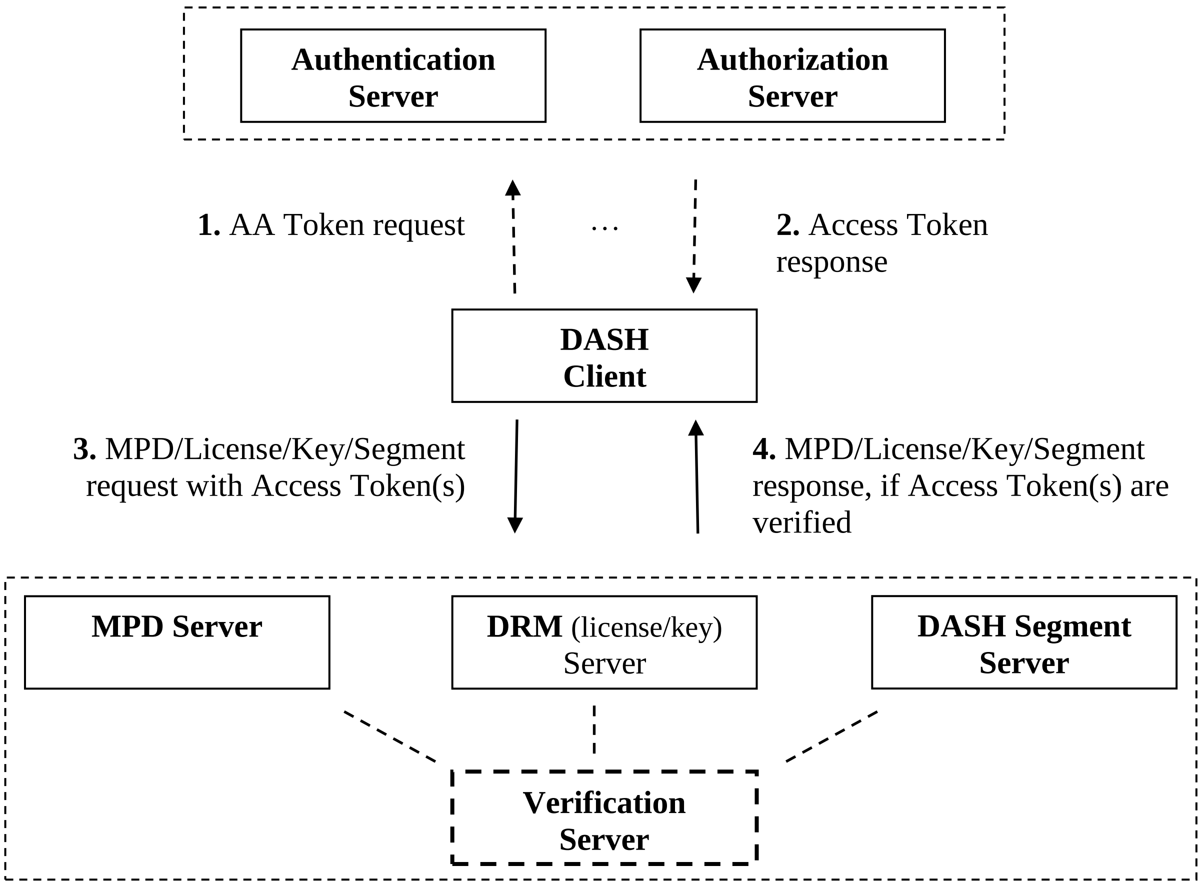

Figure 6 - AA System Architecture

Figure 6 shows logical entities that may request, issue, provide and verify token-based AA information for the purpose of granting access to requested MPD, DRM licenses, crypto keys and content segments. A physical entity may combine multiple logical roles, and a logical role can be played by more than one physical entities (e.g., accessing segments of different types and qualities may be authorized by different Authorization Servers and verified by different Verification Servers). The point of origin for information (e.g., credentials and protocols used for obtaining tokens) and information contained within tokens can differ; so various information flows in requesting and generating tokens are possible.

B Annex B – Overview of the signed JSON Web Token and claims from the URI Signing specification (informative)

B.1 Introduction

The following gives an overview of the way [URISigning] uses signed JSON Web Token (JWT), defined in [RFC-7519], as well as specific claims for the purpose of client authorization. Note that this only gives an short introduction and it is encouraged to read the specification [URISigning] for completeness.

The [URISigning] specification essentially defines a method for an HTTP server to validate an incoming HTTP request sent by a user agent. The validation relies on the presence of a valid signed JWT. Two types of information are specified by [URISigning]: the information on the transport of the signed JWT and the set of claims that can be used to enforce a certain distribution policy by the content provider.

B.2 Enforcement claims

The following claims are used to enforce the distribution policy which determines whether the requested resource may be delivered to the client:

- *Issuer (`iss`) [optional]* – This claim may be used to validate authorization of the issuer of a signed JWT and may be used to confirm that the indicated key was provided by said issuer.
- *URI Container (`cdniuc`) [mandatory]* –This representation can take one of several forms detailed in [URISigning], namely a URI Hash Container or URI Regular Expression Container. If the hash or the regex in the signed JWT does not match the URI of the content request, the CDN rejects the request
- *Client IP (`cdniip`) [optional]* – IP address, or IP prefix, for which the Signed URI is valid. If the CDN validating the signed JWT does not support Client IP validation, or if the Client IP in the signed JWT does not match the source IP address in the content request, the CDN rejects the request.
- *Expiry Time (`exp`) [optional]* – Expiration time on or after which the JWT is no longer accepted for processing. If the CDN validating the signed JWT does not support Expiry Time validation, or if the Expiry Time in the signed JWT corresponds to a time earlier than the time of the content request, the CDN rejects the request.
- *Not Before (`nbf`) [optional]* – Time before which the JWT is not yet accepted for processing. If the CDN validating the signed JWT does not support Not Before time validation, or if the Not Before time in the signed JWT corresponds to a time later than the time of the content request, the CDN rejects the request.
- *Nonce (`jti`) [optional]* – A unique identifier for the JWT. Can be used to prevent replay attacks if the CDN stores a list of all previously used Nonce values, and validates that the Nonce in the current JWT has never been used before.

See Section 2.1 in [URISigning] for further details on each claim.

B.3 Signature

The [URISigning] specification leverages the signature feature of the JWT.

B.4 Transport claims

In addition, the [URISigning] specification specifies parameters pertaining to the transport mechanism:

- *CDNI Expiration Time Setting (`cdniets`) [optional]*: It denotes the number of seconds to be added to the time at which the JWT is validated that gives the value of the Expiry Time (*`exp`*) claim of the next signed JWT.
- *CDNI Signed Token Transport (STT) [optional]*: The CDNI Signed Token Transport (*`cdnistt`*) claim provides a means of signalling the method through which a new signed JWT is transported from the CDN to the UA and vice versa for the purpose of Signed Token Renewal.

B.5 Signed Token Example from [URISigning]

B.5.1 Simple Example

In this example, the Access Token only limits which resource can be requested but does not provide client enforcement rules not expiration time of the token. The JWT Claim Set before signing would be:

```
{
  "exp": 1474243500,
  "iss": "uCDN Inc",
  "cdniuc": "hash:sha-256;2tderfWPa86Ku7YnzW51YUp7dGUjBS_3SW3ELx4hmWY"
}
```

B.5.2 Advanced Example

In this example, the Access Token contains an encrypted client IP, an expiration time, a unique identifier and a not-before time. The set of resources that can be requested are identified by a regex. The JWT Claim Set before signing would be:

```
{
  "aud": "dCDN LLC",
  "sub": "eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIiwia2lkIjoiZi1XYmp4
QkMzZFB1STNkMjRrUDJoZnZvczdRejY4OFVUaTZhQjBoTjk5OCJ9..XsJ7ySeChORS
Iojp.R1U8ESGU2NnW.DWR8pTbeCwQZca6SitfX_g",
  "cdniip": "eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIiwia2lkIjoiZi1XY
mp4QkMzZFB1STNkMjRrUDJoZnZvczdRejY4OFVUaTZhQjBoTjk5OCJ9..SuzoOnfg-
GVh-BOc.wQ9iSR1sTj-A04CiDmvcgg.9Ts_cIEUw6Yc6U5HaH1UPQ",
  "cdniv": 1,
  "exp": 1474243500,
  "iat": 1474243200,
  "iss": "uCDN Inc",
  "jti": "5DAafLhZAfhsbe",
  "nbf": 1474243200,
  "cdniuc": "regex:http://cdni\\.example/foo/bar/[0-9]{3}\\.png"
}
```

C Annex C - Overview of Signaling and Exchange Mechanisms in MPEG DASH (informative)

C.1 Introduction

This section provides an overview of the signaling and exchange mechanism in [DASH-AMD3].

C.2 Extended UrlQueryInfo in ISO/IEC 23009-1:2014 AMD 3:2016

The Amendment 3 [DASH-AMD3] introduces the **`ExtUrlQueryInfo`** element. It exhibits features to support advanced workflows desirable for the exchange of AA Tokens. For instance, the MPD author can indicate that the value of a query string parameter of a segment request is to be found in headers of HTTP responses. In addition, the type of HTTP responses to be inspected for values can also be explicitly signaled, namely “segment”, “xlink”, “mpd” and “callback”. This typically enables the retrieval of AA Tokens in HTTP responses headers and its insertion as query string parameters in future segment requests by the DASH client.

Here is an example of implementing a token exchange between server and DASH client. Let us assume that the CDN provides an access token in the HTTP header named "AA-token" in a MPD response, the following MPD example instructs the DASH client to extract the value of this header from MPD and segment responses and to insert it back in the query string parameter "AA-token" for MPD and segment requests.

In particular, the DASH client parses the `@queryTemplate` attribute of the **`ExtUrlQueryInfo`** element. The header key (left to “:”) indicates that the values have to be extracted from headers of HTTP responses. The header name (right to “:”) indicates the name of the header whose values needs to be extracted by the DASH client. In this example, the DASH client will extract the value of the HTTP header “AA-token-server” from MPD responses (see `@headerParamSource` attribute) and will insert it in every segment and MPD request (see `@includeInRequests`) in the query string “AA-token”.

```
<AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1" maxWidth="1280"
maxHeight="720" maxFrameRate="25" par="16:9">
  <EssentialProperty schemeIdUri="urn:mpeg:dash:urlparam:2016:querystring"
xmlns:up="urn:mpeg:dash:schema:urlparam:2016">
    <up:ExtUrlQueryInfo
        headerParamSource="mpd"
        includeInRequests="segment mpd"
        queryTemplate="AA-token=$header:AA-token-server$"/>
  </EssentialProperty>
  <SegmentTemplate duration="2" startNumber="1" media="video_$Number$_$Bandwidth$bps.mp4">
  </SegmentTemplate>
    <Representation id="v0" codecs="avc3.4d401f" width="1280" height="720" frameRate="25"
sar="1:1" bandwidth="3000000"/>
    <Representation id="v1" codecs="avc3.4d401f" width="640" height="360" frameRate="25"
sar="1:1" bandwidth="1500000"/>
</AdaptationSet>
```

Let us assume that the HTTP response for the MPD is as follows:

```
HTTP/1.1 200 OK
Content-Length: 3458
Cache-Control: max-age=86400
Content-Type: application/dash+xml
AA-token-server: abcdef

<?xml version="1.0" encoding="UTF-8"?>
<MPD>
```

```
…
</MPD>
```

1. Computation of an initial query string:

   initialQueryString = "AA-token-server=abcdef"
2. Computation of a final query string:

   finalQueryString = "AA-token=abcdef"
3. Modified media segment URLs building process:

```
http://www.example.com/dash/video_1_3000000bps.mp4?AA-token=abcdef
```

```
http://www.example.com/dash/video_2_3000000bps.mp4?AA-token=abcdef
```

```
http://www.example.com/dash/video_3_3000000bps.mp4?AA-token=abcdef
```

```
http://www.example.com/dash/video_4_3000000bps.mp4?AA-token=abcdef
```

C.3 Client Authentication and Content Authorization in ISO/IEC 23009-1:2014 AMD 3:2016

The following examples illustrate how client authentication and content access authorization information is signaled in the MPD according to [DASH-AMD3].

```
<EssentialProperty  schemeIdUri="urn:org:example:plan-a"
  id="mpeg:dash:client-authentication:2015"      `
  value="http://authentication.serverA.com/protocolA?=ServiceSpecificInfoA"/>

<EssentialProperty  schemeIdUri="urn:org:example:plan-b
  id="mpeg:dash:client-authentication:2015"
  value="http:// authentication.serverA.com/protocolB?=ServiceSpecificInfoB"/>

<EssentialProperty  schemeIdUri="urn:org:example:plan-c"
  id="mpeg:dash:content-authorization:2015"
  value="http://authorization.serverC.com/protocolC?=ContentSpecificInfoC"/>

<EssentialProperty  schemeIdUri="urn:org:example:plan-d"
  id="mpeg:dash:content-authorization:2015"
  value="http://authorization.serverD.com/protocolD?=ContentSpecificInfoD"/>
```

The `@id` attribute have specific values that indicate the type of the scheme, namely authentication or authorization with respectively, `mpeg:dash:client-authentication:2015` and `mpeg:dash:client-authorization:2015`.