INTERNATIONAL STANDARD

ISO/IEC 23009-6

First edition
2017-12

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 6: **DASH with server push and WebSockets**

Technologies de l'information — Diffusion adaptative dynamique sur HTTP (DASH) —

Partie 6: DASH avec serveur de poussée et protocoles WebSocket

Reference number
ISO/IEC 23009-6:2017(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

A list of all parts in the ISO/IEC 23009 series can be found on the ISO website.

Introduction

Dynamic Adaptive Streaming over HTTP (DASH) is intended to support a media-streaming model for delivery of media content in which the control lies exclusively with the client.

This document specifies carriage of MPEG DASH media presentations over full duplex HTTP-compatible protocols, particularly HTTP/2 (version 2 of the HTTP protocol as defined by the IETF in Reference [8]) and WebSocket (WebSocket protocol as defined by the IETF in RFC 6455). This carriage takes advantage of the capabilities of these protocols to optimize delivery of MPEG DASH media presentations.

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 6: DASH with server push and WebSockets

1 Scope

This document specifies carriage of MPEG-DASH media presentations over full duplex HTTP-compatible protocols, particularly HTTP/2 and WebSocket. This carriage takes advantage of the features these protocols support over HTTP/1.1 to improve delivery performance, while still maintaining backwards compatibility, particularly for the delivery of low latency live video.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEEE 1003.1-2008, *IEEE Standard for Information Technology — Portable Operating System Interface (POSIX), Base Specifications, Issue 7*

IETF RFC 3986, *Uniform Resource Identifiers (URI): Generic Syntax, January 2005*

IETF RFC 6455, *The WebSocket Protocol, December 2011*

IETF RFC 7158, *The JavaScript Object Notation (JSON) Data Interchange Format, March 2013*

IETF RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, June 2014*

3 Terms, definitions, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the following terms, definitions, abbreviated terms and conventions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
push acknowledgement
Push Ack
response modifier, sent from a server to a client, which enables a server to state the *push strategy* (3.1.3) used when processing a request

3.1.2
push directive
request modifier, sent from a client to a server, which enables a client to express its expectations regarding the server's *push strategy* (3.1.3) for processing a request

3.1.3
push strategy
segment transmission strategy, that defines the ways in which segments may be pushed from a server to a client

3.1.4
DASH server push
push
transmission of a segment from server to client based on a *push strategy* (3.1.3), as opposed to directly in response to a client request

3.2 Conventions

NOTE In this document, data formats are described using the ABNF method as described in RFC 5234.

STRING = 1* VCHAR

INTEGER = 1* DIGIT

PPCHAR= `%x21 / %x23-7E`

SQUOTE= `%x27`

UCHAR= `%x21 / %x23-7A / %x7C / %x7E`

4 Background

The basic mechanisms of MPEG-DASH over HTTP/1.1 can be augmented by utilizing the new features and capabilities that are provided by the more recent Internet protocols such as HTTP/2 and WebSocket; see Annex A for several illustrative use cases. While HTTP/2 and WebSocket are quite different in details, they both allow server-initiated and client-initiated transactions, data request cancelation and multiplexing of multiple data responses.

While in the case of HTTP/2 it is possible to carry DASH presentations without additional support, these new capabilities can be used to reduce the transmission delay (latency). Also, both HTTP/2 and WebSocket are designed to interoperate with existing HTTP/1.1 infrastructure, allowing for graceful fallback to HTTP/1.1 when the more recent protocol is not available.

The overall workflow of MPEG-DASH over these protocols is shown in Figure 1. The client and server first initiate a media channel, where the server can actively push data to the other (enabled by HTTP/2 server push or WebSocket messaging). The media channel may be established via the HTTP/1.1 protocol upgrade mechanism or by some other means. After the connection is established, the DASH client requests the media or the MPD from the server, with a URI and a push strategy. This strategy informs the server about how the client would like media delivery to occur (initiated by the server or initiated by the client). Once the server receives the request, it responds with the requested data and initializes the push cycle as defined in the push strategy. Annex B shows a typical end-to-end video streaming system over HTTP/2 that can benefit from signalling and messages defined in this document.

Figure 1 shows an example DASH session wherein the client requests the MPD first and then the media segments with a push strategy. Initialization data are pushed in response to a push strategy associated to the MPD request. After receiving the requested MPD, the client starts requesting video segments from the server with the respective DASH segment URL and a segment push strategy. Then, the server responds with the requested video segment, followed by the push cycles as indicated by the segment push strategy. Typically, the client starts playing back the video after a minimum amount of data is received and then the aforementioned process repeats until the end of the media streaming session.

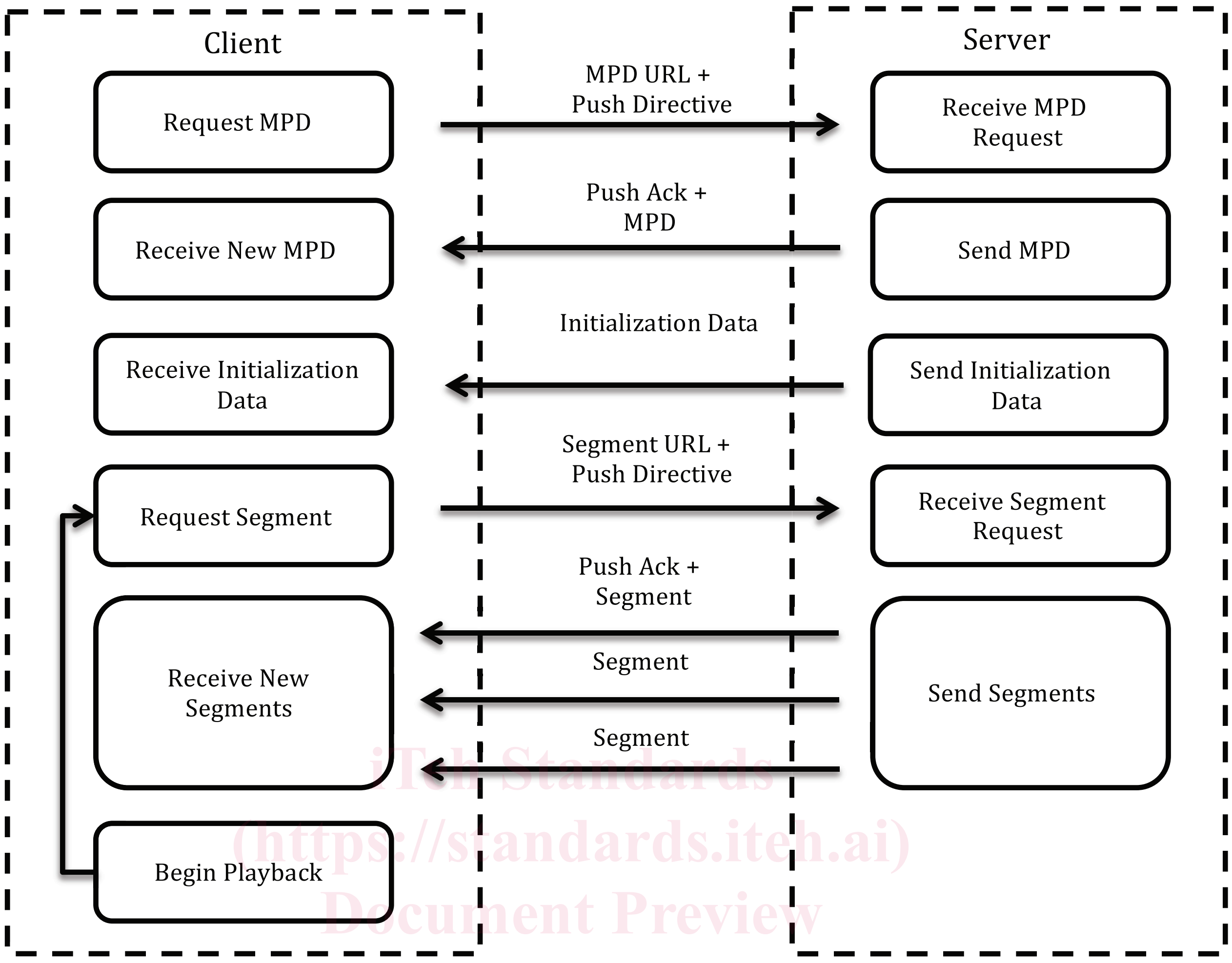

Figure 1 — Overall flow of video streaming using DASH server push

5 Specification structure

This document defines the signalling and message formats for driving the delivery of MPEG-DASH media presentations over full-duplex HTTP-compatible protocols. Details are provided for utilizing this signalling over the HTTP/2 (Clause 7) and the WebSocket (Clause 8) protocols.

Annex C provides examples of HTTP/2 client/server behaviour implementing signalling and message formats defined in this document. Annex D provides examples of WebSocket client/server behaviour implementing signalling and message formats defined in this document. Annex E illustrates the HTTP protocol upgrade and fallback procedure for WebSocket. These informational annexes are provided to demonstrate the use of the specified signalling and message formats to build streaming systems that take advantage of the full-duplex capabilities of the underlying transport protocol.

6 Definitions

6.1 Data type definitions

6.1.1 General

Clause 6 describes a number of primitive data types (see Table 1) used to define the signalling over protocols addressed in this document. Details for implementing these primitives for a given protocol may be found in the subclause of this document defining that binding.

Table 1 — Definitions of primitive data types

Data type	Base type	Description
`BinaryObject`	N/A	An untyped binary object made up of 0 or more bytes.
`Boolean`	N/A	A true or false value.
`MPD`	MPD	An MPEG-DASH Media Presentation Description (MPD), as defined in ISO/IEC 23009-1.
`Null`	N/A	An empty value.
`PushAck`	String	A response from the server acknowledging a push request. The PushAck contains the accepted values for the push strategy specified in the PushDirective. For details, see 6.1.4.
`PushDirective`	String	A directive describing the requested push strategy to be employed within the streaming session. For details, see 6.1.3.
`Segment`	Segment	An MPEG-DASH initialization or media segment, as defined in ISO/IEC 23009-1.
`String`	N/A	A UTF-8 character string.
`URI`	String	A Uniform Resource Identifier (URI), as defined in RFC 3986.
`URLList`	String	A list of URLs. For details, see 6.1.5.
`URLTemplate`	String	A URL template and corresponding parameters that describe a set of URLs. For details, see 6.1.6.

6.1.2 PushType

A `PushType` is the description of a push strategy. It contains a name identifying the push strategy and possibly its associated parameters.

The format of a `PushType` in the ABNF form is as follows:

PUSH_TYPE = PUSH_TYPE_NAME [OWS ";" OWS PUSH_PARAMS]

PUSH_TYPE_NAME = DQUOTE <URN> DQUOTE

PUSH_PARAMS = PUSH_PARAM *(OWS ";" OWS PUSH_PARAM)

PUSH_PARAM = 1*PPCHAR

Where,

'<URN>' syntax is defined in RFC 2141. Valid values for this URN according to this document are defined in Table 3,

'OWS' is defined in RFC 7230, 3.2.3 and represents optional whitespace.

The definition of PUSH_PARAMS is generic to allow the definition of new push strategies without any limitation on their parameters. Each push strategy adds some restriction on the number and on the definitions of the PUSH_PARAM instances used with it. Valid values for PUSH_PARAMS are defined in Table 4.

EXAMPLE If the push strategy expects a parameter of type 'INTEGER', then there is only one 'PUSH_PARAM' defined by 'INTEGER' as in 3.2. If it expects a parameter of type 'URLTemplate', then there is only one 'PUSH_PARAM' defined by 'URLTemplate' as in 6.1.6.

6.1.3 PushDirective

A `PushDirective` signals the push strategy that a client would like the server to use for delivery of one or more future segments. A `PushDirective` has a type (described in Table 3) and depending on the type, may have one or more additional parameters associated with it (described in Table 4).

In general, a client may signal one or more PushDirectives for a single message. The server may select at most one of the provided push strategies. This mechanism allows for clients to interoperate with servers that allow different push strategies and for forward compatibility, as the new types of push strategies are introduced.

The format of a `PushDirective` in the ABNF form is as follows:

PUSH_DIRECTIVE = PUSH_TYPE [OWS ";" OWS QVALUE]

PUSH_TYPE = <A PushType defined in 6.1.2>

QVALUE = <a qvalue, as defined in RFC 7231>

When multiple push directives are applied to a request, a client may apply a quality value ("qvalue") as is described for use in content negotiation in RFC 7231. A client may apply higher quality values to directives it wishes to take precedence over alternative directives with a lower quality value. Note that these values are hints to the server and do not imply that the server will necessarily choose the strategy with the highest quality value. If the quality value "qvalue" is not present, the default quality value is 1,0.

6.1.4 PushAck

A Push Acknowledgement (`PushAck`) is sent from the server to the client to indicate that the server intends to follow a given push strategy. At most, one Push Acknowledgment may be returned, indicating the push strategy that is in effect at the server. A Push Acknowledgment, depending on the type, may have one or more additional parameters associated with it (described in Table 4).

The format of the `PushAck` in the ABNF form is as follows:

PUSH_ACK = PUSH_TYPE

Where PUSH_TYPE is defined in 6.1.2.

6.1.5 URLList

A `URLList` describes a specific set of URLs as a delimited list. A client may use a list to explicitly signal the segments to be pushed during a push transaction. The list of URLs describes the sequence of segments to be pushed within this push transaction.

The `URLList` string format ABNF follows:

URL_LIST = LIST_ITEM *(OWS ";" OWS LIST_ITEM)

LIST_ITEM = 1*PPCHAR

Each list element is formed as a URL as defined in RFC 3986. If the URL is in relative form, it is considered relative to the segment being requested. See Annex F for examples of the URL list under various scenarios.

6.1.6 URLTemplate

A `URLTemplate` describes a specific set of URLs via a template and the corresponding parameters required to expand the template. A client may use a template to explicitly signal the segments to be pushed during a push transaction. The string is formed as a list of individual URL templates, each of which may be parameterized to signal one or more URL values. When fully evaluated, the complete list of URLs describes the sequence of segments to be pushed within this push transaction.

The `URLTemplate` format is inspired by the “level 1” URI template scheme defined in IETF RFC 6570.

NOTE The above template mechanism may be used to describe URLs contained in the MPEG-DASH MPD, whether they are formed using a SegmentTemplate or SegmentList. It is not possible to use `URLTemplate` to describe URLs formed via SegmentTemplate when they use `$Time$` variable, unless the time value of each segment can be predicted or is described via SegmentTimeline, typically when @r is present and is not negative.

In addition, each parameter may be suffixed with an additional format tag aligned with the printf() format tag as defined in IEEE 1003.1-2008 following this prototype:

%0[width]d

The width parameter is an unsigned integer that provides the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result shall be padded with zeros. The value is not truncated even if the result is larger.

The `URLTemplate` string format ABNF follows:

URL_TEMPLATE = TEMPLATE_ITEM *(OWS ";" OWS TEMPLATE_ITEM)

TEMPLATE_ITEM = SQUOTE TEMPLATE_ELEMENT SQUOTE [OWS “:” OWS “{“ OWS TEMPLATE_PARAMS OWS “}”]

TEMPLATE_ELEMENT = CLAUSE_LITERAL [CLAUSE_VAR [CLAUSE_LITERAL]]

CLAUSE_LITERAL = 1*UCHAR

CLAUSE_VAR = “{%0” 1*DIGIT “d}” / “{}”

TEMPLATE_PARAMS = VALUE_LIST / VALUE_RANGE

VALUE_LIST = 1*DIGIT *(OWS "," OWS 1*DIGIT)

VALUE_RANGE = 1*DIGIT OWS "-" OWS 1*DIGIT

Each template element is formed as a URL as defined in RFC 3986, containing up to one macro for parameterization. If the URL is in relative form, it is considered relative to the segment being requested.

The {} parameter is used to specify a specified list or range of URLs that differ by segment number or timestamp and is expanded using the provided value specifier. If no parameter is provided, the value specifier is optional. This makes it possible to provide a simple list of URLs.

The URL list will be generated from each template item by evaluating the provided parameter. For number ranges, this means generating a URL for each segment number in the range provided (inclusive).

The complete URL list is formed by expanding each URL template in turn, creating an ordered list of URLs. See Annex F for examples of the push template under various scenarios.

6.1.7 FastStartParams

A fast start parameter set (`FastStartParams`) is sent from the client to the server to signal the client’s preferences for initialization information and media, which may be used by a server to determine the most appropriate set of segments to push to the client in response to an MPD request.

The parameter set is expressed as a set attributes, made up of keys or key/value pairs. Each attribute shall be treated as AND conditions.

The `FastStartParams` string format ABNF is as follows:

FAST_START_PARAMS = ATTRIBUTE_LIST / ATTRIBUTE_ITEM