Consumer Technology Association™

CTA Specification

Web Application Video Ecosystem - Common Media Client Data

CTA-5004

September 2020

NOTICE

Consumer Technology Association (CTA)™ Standards, Bulletins and other technical publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards, Bulletins and other technical publications shall not in any respect preclude any member or nonmember of the Consumer Technology Association from manufacturing or selling products not conforming to such Standards, Bulletins or other technical publications, nor shall the existence of such Standards, Bulletins and other technical publications preclude their voluntary use by those other than Consumer Technology Association members, whether the standard is to be used either domestically or internationally.

Standards, Bulletins and other technical publications are adopted by the Consumer Technology Association in accordance with the American National Standards Institute (ANSI) patent policy. By such action, the Consumer Technology Association does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Standard, Bulletin or other technical publication.

This document does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this document to establish appropriate safety and health practices and to determine the applicability of regulatory limitations before its use.

(Formulated under the cognizance of the CTA **WAVE Project**; for information please see cta.tech/WAVE.)

Published by
CONSUMER TECHNOLOGY ASSOCIATION
Technology & Standards Department
www.cta.tech

FORWARD

This standard was developed by the Consumer Technology Association's Web Application Video Ecosystem (WAVE).

(This page intentionally left blank.)

TABLE OF CONTENTS

(This page intentionally left blank.)

Common Media Client Data

1 INTRODUCTION

Media player clients can convey information to Content Delivery Networks (CDNs) with each object request. This information can be useful in log analysis, QoS monitoring and delivery optimization. Session identification allows thousands of individual server log lines to be interpreted as a single user session, leading to a clearer picture of end-user quality of service. Bitrate, buffer and segment signaling allow CDNs to fine-tune and optimize their midgress traffic by intelligently reacting to the time constraints implicit in each request. Prefetch hints allow CDNs to have content ready at the edge ahead of the client request, improving delivery performance. Buffer starvation flags allow performance problems across a multi-CDN delivery surface to be identified in real-time. In combination, this transferred data should improve the quality of service offered by CDNs, which in turn will improve the quality of experience enjoyed by consumers.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [1].

This document outlines a simple means by which every media player can communicate data with each media object request and have it received and processed consistently by every CDN.

2 DATA TRANSITION MODES

The media client data can be sent by one of three means:

- As a custom HTTP request header,
- As a HTTP query argument,
- As a JSON object independent of the HTTP object request.

A HTTP request can carry a Common Media Client Data (CMCD) header or a CMCD query arg, but it MUST NOT carry both. The preferred mode of transmission for HTTP requests is to use custom headers.

> Note: Usage of a custom header from a web browser user-agent will trigger a preflight OPTIONS request before each unique media object request. This will lead to an increased request rate against the server. As a result, for CMCD transmissions from web browser user-agents that require CORS-preflighting per URL, the preferred mode of use is query arguments.

2.1 Header field definition

Four headers are defined to transmit the data. The payload key/value pairs are sharded over these headers based upon their expected level of entropy, in order to assist with HPACK/QPACK [2] header compression. The headers begin with the prefix "CMCD-" and have the following case-insensitive names:

`CMCD-Request`

`CMCD-Object`

`CMCD-Status`

`CMCD-Session`

2.2 Query argument definition

The query argument is case-sensitive and capitalization MUST be used.

`CMCD=<URL_encoded_concatenation_of_key_value_pairs><reserved_character>`

The reserved character is defined by RFC3986 [3]. This reserved character is optional at the end of the URL.

If the request already bears a query string, then an ampersand Unicode 0x26 character should precede the CMCD field.

2.3 JSON Object definition

{ <key>:<value>, … }

The JSON object MUST follow the formatting requirements defined by RFC8259 [4]. Key names and values are defined in Table 1.

3 DATA PAYLOAD DEFINITION

3.1 Payload definition for JSON transmission

1. The key names described in this specification are reserved. Custom key names may be used, but they MUST carry a hyphenated prefix to ensure that there will not be a namespace collision with future revisions to this specification. Clients SHOULD use a reverse-DNS syntax when defining their own prefix.
2. All key names are case-sensitive.
3. Values of type Token in Table 1 MUST be encoded as JSON type String.
4. Values of type Decimal and Integer in Table 1 MUST be encoded as JSON type Number.
5. Values of type Boolean in Table 1 MUST be encoded as JSON type Boolean [4].

6. All keys are OPTIONAL.
7. Key-value pairs SHOULD be sequenced in alphabetical order of the key name in order to reduce the fingerprinting surface exposed by the player.
8. Transport Layer Security SHOULD be used to protect all transmission of CMCD data.

3.2 Payload definition for Headers and Query Argument transmission

The data payload for Header and Query Argument transmission consists of a series of key/value pairs constructed according to the following rules:

1. All information in the payload MUST be represented as <key>=<value> pairs.
2. The key and value MUST be separated by an equals sign Unicode 0x3D. If the value type is BOOLEAN and the value is TRUE, then the equals sign and the value MUST be omitted.
3. Successive key/value pairs MUST be delimited by a comma Unicode 0x2C.
4. The key names described in this specification are reserved. Custom key names may be used, but they MUST carry a hyphenated prefix to ensure that there will not be a namespace collision with future revisions to this specification. Clients SHOULD use a reverse-DNS syntax when defining their own prefix.
5. If headers are used for data transmission, then custom keys SHOULD be allocated to one of the four defined header names based upon their expected level of variability:
    a. CMCD-Request: keys whose values vary with each request.
    b. CMCD-Object: keys whose values vary with the object being requested.
    c. CMCD-Status: keys whose values do not vary with every request or object.
    d. CMCD-Session: keys whose values are expected to be invariant over the life of the session.
6. All key names are case-sensitive.
7. Any value of type String MUST be enclosed by opening and closing double quotes Unicode 0x22. Double quotes and backslashes MUST be escaped using a backslash "\" Unicode 0x5C character. Any value of type Token does not require quoting.
8. All keys are OPTIONAL.
9. Key-value pairs SHOULD be sequenced in alphabetical order of the key name in order to reduce the fingerprinting surface exposed by the player.

10. If the data payload is transmitted as a query argument, then the entire payload string MUST be URLEncoded per [5]. Data payloads transmitted via headers MUST NOT be URLEncoded.
11. The data payload syntax is intended to be compliant with Structured Field Values for HTTP [6].
12. Transport Layer Security SHOULD be used to protect all transmission of CMCD data.

3.3 Reserved keys

The reserved keys and their definitions are defined in Table 1 below.

Table 1: Reserved Key and Value definitions

Description	Key Name	Header Name	Type & Unit	Value definition
Encoded bitrate	br	CMCD-Object	Integer kbps	The encoded bitrate of the audio or video object being requested. This may not be known precisely by the player; however, it MAY be estimated based upon playlist/manifest declarations. If the playlist declares both peak and average bitrate values, the peak value should be transmitted.
Buffer length	bl	CMCD-Request	Integer milliseconds	The buffer length associated with the media object being requested. This value MUST be rounded to the nearest 100 ms. This key SHOULD only be sent with an object type of ‘a’, ‘v’ or ‘av’.
Buffer starvation	bs	CMCD-Status	Boolean	Key is included without a value if the buffer was starved at some point between the prior request and this object request, resulting in the player being in a rebuffering state and the video or audio playback being stalled. This key MUST NOT be sent if the buffer was not starved since the prior request. If the object type ‘ot’ key is sent along with this key, then the ‘bs’ key refers to the buffer associated with the particular object type. If no object type is communicated, then the buffer state applies to the current session.

Description	Key Name	Header Name	Type & Unit	Value definition
Content ID	cid	CMCD-Session	String	A unique string identifying the current content. Maximum length is 64 characters. This value is consistent across multiple different sessions and devices and is defined and updated at the discretion of the service provider.
Object duration	d	CMCD-Object	Integer milliseconds	The playback duration in milliseconds of the object being requested. If a partial segment is being requested, then this value MUST indicate the playback duration of that part and not that of its parent segment. This value can be an approximation of the estimated duration if the explicit value is not known.
Deadline	dl	CMCD-Request	Integer milliseconds	Deadline from the request time until the first sample of this Segment/Object needs to be available in order to not create a buffer underrun or any other playback problems. This value MUST be rounded to the nearest 100ms. For a playback rate of 1, this may be equivalent to the player's remaining buffer length.
Measured throughput	mtp	CMCD-Request	Integer kilobits per second (kbps)	The throughput between client and server, as measured by the client and MUST be rounded to the nearest 100 kbps. This value, however derived, SHOULD be the value that the client is using to make its next Adaptive Bitrate switching decision. If the client is connected to multiple servers concurrently, it must take care to report only the throughput measured against the receiving server. If the client has multiple concurrent connections to the server, then the intent is that this value communicates the aggregate throughput the client sees across all those connections.
Next object request	nor	CMCD-Request	String	Relative path of the next object to be requested. This can be used to trigger pre-fetching by the CDN. This MUST be a path relative to the current request. This string MUST be URLEncoded [5]. The client SHOULD NOT depend upon any pre-fetch action being taken - it is merely a request for such a pre-fetch to take place.

Description	Key Name	Header Name	Type & Unit	Value definition
Next range request	nrr	CMCD-Request	String of the form "<range-start>-<range-end>"	If the next request will be a partial object request, then this string denotes the byte range to be requested. If the 'nor' field is not set, then the object is assumed to match the object currently being requested. The client SHOULD NOT depend upon any pre-fetch action being taken – it is merely a request for such a pre-fetch to take place. Formatting is similar to the HTTP Range header, except that the unit MUST be 'byte', the 'Range:' prefix is NOT required and specifying multiple ranges is NOT allowed. Valid combinations are: "<range-start>-" "<range-start>-<range-end>" "-<suffix-length>"
Object type	ot	CMCD-Object	Token - one of [m, a,v,av,i,c,tt,k,o]	The media type of the current object being requested: m = text file, such as a manifest or playlist a = audio only v = video only av = muxed audio and video i = init segment c = caption or subtitle tt = ISOBMFF timed text track k = cryptographic key, license or certificate. o = other If the object type being requested is unknown, then this key MUST NOT be used.
Playback rate	pr	CMCD-Session	Decimal	1 if real-time, 2 if double speed, 0 if not playing. SHOULD only be sent if not equal to 1.
Requested maximum throughput	rtp	CMCD-Status	Integer kilobits per second (kbps)	The requested maximum throughput that the client considers sufficient for delivery of the asset. Values MUST be rounded to the nearest 100kbps. For example, a client would indicate that the

Description	Key Name	Header Name	Type & Unit	Value definition
				current segment, encoded at 2Mbps, is to be delivered at no more than 10Mbps, by using rtp=10000. Note: This can benefit clients by preventing buffer saturation through over-delivery and can also deliver a community benefit through fair-share delivery. The concept is that each client receives the throughput necessary for great performance, but no more. The CDN may not support the rtp feature.
Streaming format	sf	CMCD-Session	Token - one of [d,h,s,o]	The streaming format that defines the current request. d = MPEG DASH h = HTTP Live Streaming (HLS) s = Smooth Streaming o = other If the streaming format being requested is unknown, then this key MUST NOT be used.
Session ID	sid	CMCD-Session	String	A GUID identifying the current playback session. A playback session typically ties together segments belonging to a single media asset. Maximum length is 64 characters. It is RECOMMENDED to conform to the UUID specification [7].
Stream type	st	CMCD-Session	Token - one of [v,l]	v = all segments are available – e.g., VOD l = segments become available over time – e.g., LIVE
Startup	su	CMCD-Request	Boolean	Key is included without a value if the object is needed urgently due to startup, seeking or recovery after a buffer-empty event. The media SHOULD not be rendering when this request is made. This key MUST not be sent if it is FALSE.

Description	Key Name	Header Name	Type & Unit	Value definition
Top bitrate	tb	CMCD-Object	Integer Kbps	The highest bitrate rendition in the manifest or playlist that the client is allowed to play, given current codec, licensing and sizing constraints.
CMCD version	v	CMCD-Session	Integer	The version of this specification used for interpreting the defined key names and values. If this key is omitted, the client and server MUST interpret the values as being defined by version 1. Client SHOULD omit this field if the version is 1.

A player SHOULD supply sid on all media object requests in a session, including playlists/manifests, init files, captioning files and DRM key requests. Other keys should be applied where they have contextual meaning. For example, a 'br' (bitrate) key on a manifest request is inappropriate and could be omitted.

4 SERVER PROCESSING REQUIREMENTS

1. Server MUST only process these requirements when data is received via a valid CMCD header or query arg.
2. A server, upon receiving common media client data, MUST interpret the keys according to their definition in this document.
3. Unknown keys, which the server does not understand, MUST be ignored.
4. Values that do not meet the structured data definition (such as an invalid token, or a string when an integer is expected) MUST be ignored.
5. Since there is no guarantee that keys are included, the server MUST be robust against the absence of individual keys on any given request.
6. The server MUST be able to correctly process the key-value pairs irrespective of the order in which they are defined.
7. If the sid key is present, the server SHOULD propagate that value to the server access logs. The server access logs SHOULD conform to RFC6302 [8].
8. The server MUST support both the header and query argument methods of data transmission. The server MUST process data from only the query arg or header channel, but never both for a given request. If any CMCD headers are present, then any CMCD query arg information MUST be ignored.
9. The server, upon receiving the requested throughput (rtp) attribute, is not required to throttle the response at the requested value. It is merely a request

from the client and the server may have other business requirements that dictate throttling at a different value, or not throttling the response at all.

10. The server, upon receiving the nor "next object request" or nrr "next range request" attributes, MAY optionally decide not to implement any pre-fetch action against that data.
11. Servers SHOULD provide the necessary CORS responses to allow browser-based clients to send custom headers, specifically:
    a. Access-Control-Allow-Headers response header with a value that contains "CMCD-Request, CMCD-Object, CMCD-Status, CMCD-Session".
    b. Access-Control-Allow-Methods with a value that includes GET.
12. Servers SHOULD be aware that malicious clients may send false key data with the objective of either attacking the server or gaining an unfair delivery advantage. The server SHOULD validate incoming key data before any performance impacting behaviors are executed.
13. Servers MUST ignore the entire data set if the signaled version is greater than they understand, as they cannot know which fields have been modified or deprecated. Servers SHOULD log the version incompatibility so that there is a record of why the data is not getting logged.

Note: Any caching proxy should be aware that the CMCD payload will be constantly changing and therefore has the potential to pollute cache keys. Implementers may wish to exclude CMCD query arguments from any cache key.

Note: Origin servers are advised to not include the CMCD-Request, CMCD-Object, CMCD-Status, CMCD-Session headers in the Vary header [9].

5 SECURITY AND PRIVACY CONSIDERATIONS

5.1 Threat environment

The data transmitted as defined by this specification is passed between a client device and a Content Delivery Network (CDN), as described in Section 2, Data Transmission Modes. Data is exchanged over HTTP/HTTPS during the regular process of a media client requesting content. The HTTP Request Headers and Query Arguments utilized for data transmission are mature and established technologies for data transmission over the web. Transport Layer Security (TLS) can provide confidentiality and integrity for data during transmission.

5.2 Threats to the server

A malicious client may inject false data. This tactic may be part of replay, message insertion, or modification attacks. If the server-client communication is delivered over HTTP, then man-in-the-middle attacks are feasible. Use of HTTPS for communications mitigates these attacks. All server responses are optional, which aids in server-based protection strategies. The server is not required to take any action upon receiving CMCD data.

Some limited denial-of-service amplification opportunity exists for malicious clients utilizing the next-object-request key. Requiring the next object to be a relative path to the current request, along with the prefetch operation being optional for the server, helps mitigate this amplification.

5.3 Threats to the client

Privacy concerns resulting from successful eavesdropping and man-in-the-middle attacks in this threat environment indicate the need to strictly limit device-identifiable data. Therefore, fingerprinting opportunities have been minimized; see sections 3.1 and 3.2. Identification confidentiality of content, as described above, is dependent upon protocol-layer protections since the playback client must by definition request playback of specific content pieces to operate.

Defense against malicious content injected via man-in-the-middle, message insertion, or message modification attacks is likewise dependent upon protocol-layer protections. HTTPS is strongly recommended over HTTP, when applicable, for all CMCD data transmissions over the web.

5.4 Specific mitigations

As discussed above, this specification does not expose any security issues that are not already exposed to a client making media object requests or to an edge server that answers all incoming requests. A number of steps have been taken to mitigate specific security and privacy concerns:

- The ‘nor’ key value must be a relative path to the current request. This makes it harder to inject false requests to arbitrary objects.
- Only one object can be requested by the ‘nor’ key value – this lowers amplification opportunity.
- All requests to the server are optionally executed by the server, meaning that a server can ignore them for security concerns (such as a rate-based threshold being exceeded) and still be compliant with the specification.
- Session ID is a GUID which is highly unlikely to repeat for a particular user.
- Measured throughput, requested maximum throughput, deadline and buffer length are bucketed to reduce fingerprinting surface.

- Key-value pairs should be sequenced in alphabetical order to reduce fingerprinting surface.
- Personally Identifiable Information fields, such as IP address, cookie information and location data, are intentionally not carried by the specification.

6 EXAMPLES

These examples illustrate valid data combinations across the three delivery modes.

6.1 Header examples

1. CMCD-Session:sid="6e2fb550-c457-11e9-bb97-0800200c9a66"

2. CMCD-Request:mtp=25400
   CMCD-Object:br=3200,d=4004,ot=v,tb=6000
   CMCD-Status:bs,rtp=15000
   CMCD-Session:sid="6e2fb550-c457-11e9-bb97-0800200c9a66"

3. CMCD-Status:bs,rtp=15000
   CMCD-Session:sid="6e2fb550-c457-11e9-bb97-0800200c9a66"

4. CMCD-Status:bs
   CMCD-Request:su

5. CMCD-Object:d=4004,
   CMCD-Session:com.example-myNumericKey=500,com.example-myStringKey="myStringValue"

6. CMCD-Session:sid="6e2fb550-c457-11e9-bb97-0800200c9a66"
   CMCD-Request:nor="..%2F300kbps%2Fsegment35.m4v"

7. CMCD-Session:sid="6e2fb550-c457-11e9-bb97-0800200c9a66"
   CMCD-Request:nrr="12323-48763"

8. CMCD-Session:sid="6e2fb550-c457-11e9-bb97-0800200c9a66"
   CMCD-Request:nor="..%2F300kbps%2Ftrack.m4v",nrr="12323-48763"

9. CMCD-Request:bl=21300,dl=18500,mtp=48100,nor="..%2F300kbps%2Ftrack.m4v",nrr="12323-48763",su
   CMCD-Object:br=3200,d=4004,ot=v,tb=6000
   CMCD-Status:bs,rtp=12000
   CMCD-Session:cid="faec5fc2-ac30-11ea-bb37-0242ac130002",pr=1.08,sf=d,sid="6e2fb550-c457-11e9-bb97-0800200c9a66",st=v

6.2 Corresponding Query Arg examples

1. ?CMCD=sid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22

2. ?CMCD=br%3D3200%2Cbs%2Cd%3D4004%2Cmtp%3D25400%2Cot%3Dv%2Crtp%3D15000%2Csid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22%2Ctb%3D6000

3. ?CMCD=b%2Crtp%3D15000%2Csid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22

4. ?CMCD=bs%2Csu

5. ?CMCD=d%3D4004%2Ccom.example-myNumericKey%3D500%2Ccom.example-myStringKey%3D%22myStringValue%22

6. ?CMCD=nor%3D%22..%252F300kbps%252Fsegment35.m4v%22%2Csid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22

7. ?CMCD=nrr%3D%2212323-48763%22%2Csid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22

8. ?CMCD=nor%3D%22..%252F300kbps%252Ftrack.m4v%22%2Cnrr%3D%2212323-48763%22%2Csid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22

9. ?CMCD=bl%3D21300%2Cbr%3D3200%2Cbs%2Ccid%3D%22faec5fc2-ac30-11ea-bb37-0242ac130002%22%2Cd%3D4004%2Cdl%3D18500%2Cmtp%3D48100%2Cnor%3D%22..%252F300kbps%252Ftrack.m4v%22%2Cnrr%3D%2212323-48763%22%2Cot%3Dv%2Cpr%3D1.08%2Crtp%3D12000%2Csf%3Dd%2Csid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22%2Cst%3Dv%2Csu%2Ctb%3D6000

6.3 Corresponding JSON examples

1. {"sid": "6e2fb550-c457-11e9-bb97-0800200c9a66"}

2. {"br": 3200,"bs":true,"d": 4004,"mtp": 25400, "ot": "v", "rtp":15000,"sid": "6e2fb550-c457-11e9-bb97-0800200c9a66","tb":6000}

3. {"bs":true,"rtp":15000,"sid": "6e2fb550-c457-11e9-bb97-0800200c9a66"}

4. {"bs":true,"su":true}

5. {"d":4004,"com.example-myNumericKey":500,"com.example-myStringKey":"myStringValue"}

6. {"nor":"..%2F300kbps%2Fsegment35.m4v","sid":"6e2fb550-c457-11e9-bb97-0800200c9a66"}

7. {"nrr":"12323-48763","sid":"6e2fb550-c457-11e9-bb97-0800200c9a66"}

8. {"nor":"..%2F300kbps%2Ftrack.m4v","nrr":"12323-48763","sid":"6e2fb550-c457-11e9-bb97-0800200c9a66"}

9. {"bl":21300,"br":3200,"bs":true,"cid":"faec5fc2-ac30-11ea-bb37-0242ac130002","d":4004,"dl":18500,"mtp":48100,"nor":"..%2F300kbps%2Ftrack.m4v","nrr":"12323-48763","ot":"v","pr":1.08,"rtp":12000,"sf":"d","sid":"6e2fb550-c457-11e9-bb97-0800200c9a66","st":"v","su":true,"tb":6000}

7 EXTERNAL REFERENCES

1. IETF RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*. https://tools.ietf.org/html/rfc2119.
2. IETF RFC 7541, *HPACK Header compression*. https://tools.ietf.org/html/rfc7541.
3. IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*. https://tools.ietf.org/html/rfc3986.
4. IETF RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format*. https://tools.ietf.org/html/rfc8259.
5. WHATWG *URL Living Standard*, Section 5, *URL Encoding*. Accessed 4 August 2020. https://url.spec.whatwg.org/#application/x-www-form-urlencoded.
6. IETF *Structured Field Values for HTTP* (work in progress). https://datatracker.ietf.org/doc/draft-ietf-httpbis-header-structure/.
7. IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*. https://tools.ietf.org/html/rfc4122.
8. IETF RFC 6302, *Logging Recommendations for Internet-Facing Servers.* https://tools.ietf.org/html/rfc6302.
9. IETF RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, Section 7.1.4, *Vary header*. http://tools.ietf.org/html/rfc7231#section-7.1.4.

Consumer Technology Association Document Improvement Proposal

If in the review or use of this document a potential change is made evident for safety, health or technical reasons, please email your reason/rationale for the recommended change to standards@CTA.tech.

Consumer Technology Association
Technology & Standards Department
1919 S Eads Street, Arlington, VA 22202
FAX: (703) 907-7693 standards@CTA.tech

Consumer Technology Association™