Media Fragments URI 1.0 (basic)

W3C Recommendation 25 September 2012

This version:
http://www.w3.org/TR/2012/REC-media-frags-20120925/
Latest version:
http://www.w3.org/TR/media-frags/
Previous version:
http://www.w3.org/TR/2012/PR-media-frags-20120315/

Editors:
Raphaël Troncy , EURECOM
Erik Mannens , IBBT Multimedia Lab, University of Ghent
Silvia Pfeiffer , W3C Invited Expert
Davy Van Deursen , IBBT Multimedia Lab, University of Ghent
Contributors:
Michael Hausenblas , DERI, National University of Ireland, Galway
Philip Jägenstedt , Opera Software
Jack Jansen , CWI, Centrum Wiskunde & Informatica, Amsterdam
Yves Lafon , W3C
Conrad Parker , W3C Invited Expert
Thomas Steiner , Google, Inc.

Please refer to the **errata** for this document, which may include normative corrections.

See also **translations**.

Abstract

This document describes the Media Fragments 1.0 (basic) specification. It specifies the syntax for constructing media fragment URIs and explains how to handle them when used over the HTTP protocol. The syntax is based on the specification of particular name-value pairs that can be used in URI fragment and URI query requests to restrict a media resource to a certain fragment. The Media Fragment WG has no authority to update registries of all targeted media types. We recommend media type owners to harmonize their existing schemes with the ones proposed in this document and update or add the fragment semantics specification to their media type registration.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is the Recommendation of the Media Fragments URI 1.0 (basic) specification. It has been produced by the Media Fragments Working Group, which is part of the W3C Video on the Web Activity. W3C publishes a technical report as a Recommendation to indicate that the document is believed to be stable, and to encourage implementation by the developer community.

If you wish to make comments regarding this document, please send them to public-media-fragment@w3.org mailing list (public archive).

Only editorial changes were made as a result of the Proposed Recommendation phase (See diff). An implementation report is available.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

Table of Contents

Appendices

1 Introduction

Audio and video resources on the World Wide Web are currently treated as "foreign" objects, which can only be embedded using a plugin that is capable of decoding and interacting with the media resource. Specific media servers are generally required to provide for server-side features such as direct access to time offsets into a video without the need to retrieve the entire resource. Support for such media fragment access varies between different media formats and inhibits standard means of dealing with such content on the Web.

This specification provides for a media-format independent, standard means of addressing media fragments on the Web using Uniform Resource Identifiers (URI). In the context of this document, media fragments are regarded along several different dimensions such as temporal, spatial and tracks. A temporal fragment can also be marked with a name and then addressed through a URI using that name, using the id dimension. The specified addressing schemes apply mainly to audio and video resources - the spatial fragment addressing may also be used on images.

The aim of this specification is to enhance the Web infrastructure for supporting the addressing and retrieval of subparts of time-based Web resources, as well as the automated processing of such subparts for reuse. Example uses are the sharing of such fragment URIs with friends via email, the automated creation of such fragment URIs in a search engine interface, or the annotation of media fragments with RDF. Such use case examples as well as other side conditions on this specification and a survey of existing media fragment addressing approaches are provided in the requirements Use cases and requirements for Media Fragments document that accompanies this specification document.

2 Standardisation Issues

2.1 Terminology

The keywords **MUST**, **MUST NOT**, **SHOULD** and **SHOULD NOT** are to be interpreted as defined in RFC 2119.

According to *RFC 3986*, the term "URI" does not include relative references. In this document, we consider both URIs and relative references. Consequently, we use the term "URI reference" as defined in *RFC 3986* (section 4.1). For simplicity reasons, this document, however, only uses the term "media fragment URI" in place of "media fragment URI reference".

The following terms are used frequently in this document and need to be clearly understood:

- URI fragment: The fragment component is indicated by the presence of a number sign ("#") character and terminated by the end of the URI.
- URI query: The query component is indicated by the first question mark ("?") character and terminated by a number sign ("#") character or by the end of the URI.
- Media fragment URI: a URI addressing subparts of a media resource - that could be through URI queries or URI fragments

2.2 Media Fragments Standardisation

The basis for the standardisation of media fragment URIs is the URI specification, RFC 3986. Providing media fragment identification information in URIs refers here to the specification of the structure of a URI fragment or a URI query. This document will explain how URI fragments and URI queries are structured to identify media fragments. It normalises the name-value parameters used in URI fragments and URI queries to address media fragments. These build on existing CGI parameter conventions. In this section, we look at implications of standardising the structure of media fragment URIs.

2.2.1 URI Fragments

The URI specification RFC 3986 says about the format of a URI fragment in Section 3.5:

"The fragment's format and resolution is [..] dependent on the media type [RFC2046] of a potentially retrieved representation. [..] Fragment identifier semantics are independent of the URI scheme and thus cannot be redefined by scheme specifications."

This essentially means that only media type definitions (as registered through the process defined in *RFC 4288*) are able to introduce a standard structure on URI fragments for that mime type. One part of the registration process of a media type can include information about how fragment identifiers in URIs are constructed for use in conjunction with this media type.

The registration of URI fragment construction rules, as expressed in Section 4.11 of *RFC 4288*, is a SHOULD-requirement. The Media Fragment Working Group has no authority to update registries of all targeted media types. An analysis of all media type registrations showed that only a few media type registration in the audio/*, image/*, video/* branches are currently defining fragments or fragment semantics (see for example the fragment definition for SVG in *SVG 1.1*, chapter 17). To the best of our knowledge there are only few media types that actually have a specified fragment format even if it is not registered with the media type: these include Ogg, MPEG-4, and MPEG-21. Further, only a small number of software packages actually supports these fragment formats. For all others, the semantics of the fragment are considered to be unknown.

As such, the intention of this document is to propose a specification to all media type owners in the audio/*, image/*, and video/* branches for a structured approach to URI fragments and for specification of commonly agreed dimensions to address media fragments (i.e. subparts of a media resource) through URI fragments. We recommend media type owners to harmonize their existing schemes with the ones proposed in this document and update or add the fragment semantics specification to their media type registration.

2.2.2 URI Queries

The URI specification RFC 3986 says about the format of a URI query in Section 3.4:

"The query component [..] serves to identify a resource within the scope of the URI's scheme and naming authority (if any). [..] Query components are often used to carry identifying information in the form of "key=value" pairs [..]."

URI query specifications are more closely linked to the URI scheme, some of which do not even use a query component. We are mostly concerned with the HTTP *RFC 2616* and the RTP/RTSP *rfc2326* protocols here, which both support query components. HTTP says nothing about how a URI query has to be interpreted. RTSP explicitly says that fragment and query identifiers do not have a well-defined meaning at this time, with the interpretation left to the RTSP server.

The URI specification *RFC 3986* says generally that the data within the URI is often parsed by both the user agent and one or more servers. It refers in particular to HTTP in Section 7.3:

"In HTTP, for example, a typical user agent will parse a URI into its five major components, access the authority's server, and send it the data within the authority, path, and query components. A typical server will take that information, parse the path into segments and the query into key/value pairs, and then invoke implementation-specific handlers to respond to the request."

Since the interpretation of query components resides with the functionality of servers, the intention of this document with respect to query components is to recommend standard name-value pair formats for use in addressing media fragments through URI queries. We recommend server and server-type software providers to harmonize their existing schemes in use with media resources to support the nomenclature proposed in this specification.

3 URI fragment and URI query

To address a media fragment, one needs to find ways to convey the fragment information. This specification builds on URIs RFC 3986. Every URI is defined as consisting of four parts, as follows:

```
<scheme name> : <hierarchical part> [ ? <query> ] [ # <fragment> ]
```

There are therefore two possibilities for representing the media fragment addressing in URIs: the *URI query part* or the *URI fragment part*.

3.1 When to choose URI fragments? When to choose URI queries?

For media fragment addressing, both approaches - URI query and URI fragment - are useful. The main difference between a URI query and a URI fragment is that a URI query produces a new resource, while a URI fragment provides a secondary resource that has a relationship to the primary resource. URI fragments are resolved from the primary resource without another retrieval action. This means that a user agent should be capable to resolve a URI fragment on a resource it has already received without having to fetch more data from the server.

This specification imposes a further constraint, which is that the media type of the retrieved fragment should be the same as the media type of the primary resource. Among other things, this means that a URI fragment that points to a single video frame out of a longer video results in a one-frame video, not in a still image. To extract a still image, one would need to create a URI query scheme - something not envisaged here, but easy to devise.

There are different types of media fragment addressing in this specification. As noted in the Use cases and requirements for Media Fragments document (section "Fitness Conditions on Media Containers/Resources"): not all container formats and codecs are "fit" for supporting the different types of fragment URIs. "Fitness" relates to the fact that a media fragment can be extracted from the primary resource without syntax element modifications or transcoding of the bitstream.

Resources that are "fit" can therefore be addressed with a URI fragment. Resources that are "conditionally fit" can be addressed with a URI fragment with an additional retrieval action that retrieves the modified syntax elements but leaves the codec data untouched. Resources that are "unfit" require transcoding. Such transcoded media fragments cannot be addressed with URI fragments, but only with URI queries.

Therefore, when addressing a media fragment with the URI mechanism, the author has to know whether this media fragment can be produced from the (primary) resource itself without any transcoding activities or whether it requires transcoding. In the latter case, the only choice is to use a URI query and to use a server that supports transcoding and delivery of a (primary) derivative resource to satisfy the query.

3.2 Resolving URI fragments within the user agent

A user agent may itself resolve and control the presentation of media fragment URIs. The simplest case arises where the user agent has already downloaded the entire resource and can perform the extraction from its locally cached copy. For some media types, it may also be possible to perform the extraction over the network without any special protocol assistance. For temporal fragments this requires a user agent to be able to seek on the media resource using existing protocol mechanisms.

An example of a URI fragment used to address a media fragment is `http://www.example.org/video.ogv#t=60,100`. In this case, the user agent knows that the primary resource is `http://www.example.org/video.ogv` and that it is only expected to display the portion of the primary resource that relates to the fragment `#t=60,100`, i.e. seconds 60-100. Thus, the relationship between the primary resource and the media fragment is maintained.

In traditional URI fragment retrieval, a user agent requests the complete primary resource from the server and then applies the fragmentation locally. In the media fragment case, this would result in a retrieval action on the complete media resource, on which the user agent would then locally perform its fragment extraction - something generally unviable for such large resources. Therefore, media resources are not always retrieved over HTTP using a single request. They may be retrieved as a sequence of byte range requests on the original resource URI, or may be retrieved as a sequence of requests to different URIs each representing a small part of the media. The reasons for such mechanisms include bandwidth conservation, where a client chooses to space requests out over time during playback in order to maximize bandwidth available for other activities, and bandwidth adaptation, where a client selects among various representations with varying bitrate depending on the current bandwidth availability.

A user agent that knows how to map media fragments to byte ranges will be able to satisfy a URI fragment request such as the above example by itself. This is typically the case for user agents that know how to seek to media fragments over the network. For example, a user agent that deals with a media file that includes an index of its seekable structures can resolve the media fragment addresses to byte ranges from the index. This is the case e.g. with seekable QuickTime files. Another example is a user agent that knows how to seek on a media file through a sequence of byte range requests and eventually receives the correct media fragment. This is the case e.g. with Ogg files in Firefox versions above 3.5.

Similarly, a user agent that knows how to map media fragments to a sequence of URIs can satisfy a URI fragment request by itself. This is typically the case for user agents that perform adaptive streaming. For example, a user agent that deals with a media resource that contains a sequence of URIs, each a media file of a few seconds duration, can resolve the media fragment addresses to a subsequence of those URIs. This is the case with QuickTime adaptive bitrate streaming or IIS Smooth Streaming.

If such a user agent natively supports the media fragment syntax as specified in this document, it is deemed conformant to this specification for fragments and for the particular dimension.

For user agents that natively support the media fragment syntax, but have to use their own seeking approach, a complementary specification provides an optimisation that can make the byte offset seeking more efficient. It requires a conformant server with which the user agent will follow a protocol defined in the separate Media Fragments 1.0 URI (advanced) document where the user agent asks the server to do the byte range mapping for the media fragment address itself and send back the appropriate byte ranges. This approach can not be done through the URI, but has to be done through adding protocol headers. User agents that interact with a conformant server to follow this protocol will receive the appropriate byte ranges directly and will not need to do costly seeking over the network.

3.3 Resolving URI queries

The described URI fragment addressing methods only work for byte-identical segments of a media resource, since we assume a simple mapping between the media fragment and bytes that each infrastructure element can deal with. Where it is impossible to maintain byte-identity and some sort of transcoding of the resource is necessary, the user agent is not able to resolve the fragmentation by itself and a server interaction is required. In this case, URI queries have to be used since they result in a server interaction and can deliver a transcoded resource.

Another use for URI queries is when a user agent actually wants to receive a completely new resource instead of just a byte range from an existing (primary) resource. This is, for example, the case for playlists of media fragment resources. Even if a media fragment could be resolved through a URI fragment, the URI query may be more desirable since it does not carry with itself the burden of the original primary resource - its file headers may be smaller, its duration may be smaller, and it does not automatically allow access to the remainder of the original primary resource.

When URI queries are used, the retrieval action has to additionally make sure to create a fully valid new resource. For example, for the Ogg format, this implies a reconstruction of Ogg headers to accurately describe the new resource (e.g. a non-zero start-time or different encoding parameters). Such a resource will be cached in Web proxies as a different resource to the original primary resource.

An example URI query that includes a media fragment specification is `http://www.example.org/video.ogv?t=60,100`. This results in a video of duration 40s (assuming the original video was more than 100s long). Note that this resource has no per-se relationship to the original primary resource. As a user agent uses such a URI with e.g. a HTML5 video element, the browser has no knowledge about the original resource and can only display this video as a 40s long video starting at 0s. The context of the original resource is lost.

A user agent may want to display the original start time of the resource as the start time of the video in order to be consistent with the information in the URI. It is possible to achieve this in one of two ways: either the video file itself has some knowledge that it is an extract from a different file and starts at an offset - or the user agent is told through the retrieval action which original primary resource the retrieved resource relates to and can find out information about it through another retrieval action.

An example for a media resource that has knowledge about itself of the required kind are Ogg files. Ogg files that have a skeleton track and were created correctly from the primary resource will know that their start time is not 0s but 60s in the above example. The browser can simply parse this information out of the received bitstream and may display a timeline that starts at 60s and ends at 100s in the video controls if it so desires. Another option is that the browser parses the URI and knows about how media resources have a fragment specification that follows a standard. Then the browser can interpret the query parameters and extract the correct start and end times and also the original primary resource. It can then also display a timeline that starts at 60s and ends at 100s in the video controls. Further it can allow a right-click menu to click through to the original resource if required.

A use case where the video controls may neither start at 0s nor at 60s is a mashed-up video created through a list of media fragment URIs. In such a playlist, the user agent may prefer to display a single continuous timeline across all the media fragments rather than a collection of individual timelines for each fragment. Thus, the 60s to 100s fragment may e.g. be mapped to an interval at 3min20 to 4min.

No new protocol headers are required to execute a URI query for media fragment retrieval. Some optional protocol headers that improve the information exchange will be recommended later in this document.

3.4 Combining URI fragments and URI queries

A combination of a URI query for a media fragment with a URI fragment yields a URI fragment resolution on top of the newly created resource. Since a URI with a query part creates a new resource, we have to do the fragment offset on the new resource. This is simply a conformant behaviour to the URI standard RFC 3986.

For example, `http://www.example.org/video.ogv?t=60,100#t=20` will lead to the 20s fragment offset being applied to the new resource starting at 60 going to 100. Thus, the reply to this is a 40s long resource whose playback will start at an offset of 20s.

4 Media Fragments Syntax

This section describes the syntax representation of a media fragment (MF) identifier and how this should be interpreted. The guiding principles for the definition of the media fragments syntax are:

- a. The MF syntax for queries and fragments should be identical
- b. The MF syntax should be unambiguous
- c. The MF syntax should allow any UTF-8 character for dimensions that need it
- d. The MF syntax should adhere to applicable formal standards
- e. The MF syntax should adhere to de-facto usage of queries and fragments
- f. The MF syntax should be as concise as possible

4.1 General Structure

A list of name-value pairs is encoded in the query or fragment component of a URI. The name and value components are separated by an equal sign (`=`), while multiple name-value pairs are separated by an ampersand (`&`).

```
name = fragment - "&" - "="
value = fragment - "&"
namevalue = name [ "=" value ]
namevalues = namevalue *( "&" namevalue )
```

The names and values can be arbitrary Unicode strings, encoded in UTF-8 and percent-encoded as per *RFC 3986*. Here are some examples of URIs with name-value pairs in the fragment component, to demonstrate the general structure:

```
http://www.example.com/example.ogv#t=10,20
http://www.example.com/example.ogv#track=audio&t=10,20
http://www.example.com/example.ogv#id=Cap%C3%ADtulo%202
```

While arbitrary name-value pairs can be encoded in this manner, this specification defines a fixed set of dimensions. The dimension keyword name is encoded in the name component, while dimension-specific syntax is encoded in the value component.

Section **5.1.1 Processing name-value components** defines in more detail how to process the name-value pair syntax, arriving at a list of name-value Unicode string pairs. The syntax definitions in **4.2 Fragment Dimensions** apply to these Unicode strings.

4.2 Fragment Dimensions

Media fragments support addressing the media along two dimensions (in the basic version):

temporal

This dimension denotes a specific time range in the original media, such as "starting at second 10, continuing until second 20";

spatial

this dimension denotes a specific range of pixels in the original media, such as "a rectangle with size (100,100) with its top-left at coordinate (10,10)";

Media fragments support also addressing the media along two additional dimensions (in the advanced version defined in Media Fragments 1.0 URI (advanced)):

track

this dimension denotes one or more tracks in the original media, such as "the english audio and the video track";

id

this dimension denotes a named temporal fragment within the original media, such as "chapter 2", and can be seen as a convenient way of specifying a temporal fragment.

All dimensions are logically independent and can be combined. The outcome is independent of the order of the dimensions. The `id` dimension is however a shortcut for the temporal dimension and combining both dimensions need to be treated as described in section **6.2.1 Errors on the general URI level**. The `track` dimension refers to one of a set of parallel media streams (e.g. "the english audio track for a video"), not to a (possibly self-contained) section of the source media (e.g. "Audio track 2 of a CD").

4.2.1 Temporal Dimension

Temporal clipping is denoted by the name `t`, and specified as an interval with a begin time and an end time (or an in-point and an out-point in video editing terms). Either one or both parameters may be omitted, with the begin time defaulting to 0 seconds and the end time defaulting to the duration of the source media. The interval is half-open: the begin time is considered part of the interval whereas the end time is considered to be the first time point that is not part of the interval. If a single number only is given, this corresponds to the begin time except if it is preceded by a comma that would in this case indicate the end time.

Examples:

```
t=10,20   # => results in the time interval [10,20)
t=,20     # => results in the time interval [0,20)
t=10      # => results in the time interval [10,end)
```

Temporal clipping is specified as Normal Play Time (npt) RFC 2326. It can also be specified as SMPTE timecodes *SMPTE* or as real-world clock time (clock) *RFC 2326* in the advanced version described in the *Media Fragments 1.0 URI (advanced)* document. Begin and end times are always specified in the same format. The format is specified by name, followed by a colon (`:`), with `npt:` being the default. In this version of the media fragments specification there is no extensibility mechanism to add time format specifiers.

```
timeprefix    = %x74                                     ; "t"
timeparam     = npttimedef
```

Normal Play Time can either be specified as seconds, with an optional fractional part to indicate miliseconds, or as colon-separated hours, minutes and seconds (again with an optional fraction). Minutes and seconds must be specified as exactly two digits, hours and fractional seconds can be any number of digits. The hours, minutes and seconds specification for NPT is a convenience only, it does not signal frame accuracy. The specification of the "npt:" identifier is optional since NPT is the default time scheme. This specification builds on the RTSP specification of NPT *RFC 2326*.

```
npt-sec       =  1*DIGIT [ "." *DIGIT ]                  ; definitions taken
npt-hhmmss    =  npt-hh ":" npt-mm ":" npt-ss [ "." *DIGIT] ; from RFC 2326,
npt-mmss      =  npt-mm ":" npt-ss [ "." *DIGIT]
npt-hh        =   1*DIGIT     ; any positive number
npt-mm        =   2DIGIT      ; 0-59
npt-ss        =   2DIGIT      ; 0-59
```

```
npttimedef    = [ deftimeformat ":"] ( npttime  [ "," npttime ] ) / ( "," npttime )

deftimeformat = %x6E.70.74                                  ; "npt"
npttime       = npt-sec / npt-mmss / npt-hhmmss
```

Examples:

```
t=npt:10,20          # => results in the time interval [10,20)
t=npt:,121.5         # => results in the time interval [0,121.5)
t=0:02:00,121.5      # => results in the time interval [120,121.5)
t=npt:120,0:02:01.5 # => also results in the time interval [120,121.5)
```

4.2.2 Spatial Dimension

Spatial clipping selects an area of pixels from visual media streams. For this version of the media fragment specification, only rectangular selections are supported. The rectangle can be specified as pixel coordinates or percentages.

Pixels coordinates are interpreted after taking into account the resource's dimensions, aspect ratio, clean aperture, resolution, and so forth, as defined for the format used by the resource. If an anamorphic format does not define how to apply the aspect ratio to the video data's dimensions to obtain the "correct" dimensions, then the user agent must apply the ratio by increasing one dimension and leaving the other unchanged.

Rectangle selection is denoted by the name `xywh`. The value is an optional format `pixel:` or `percent:` (defaulting to pixel) and 4 comma-separated integers. The integers denote x, y, width and height, respectively, with x=0, y=0 being the top left corner of the image. If percent is used, x and width are interpreted as a percentage of the width of the original media, and y and height are interpreted as a percentage of the original height.

```
xywhprefix    = %x78.79.77.68                               ; "xywh"
xywhparam     = [ xywhunit ":" ] 1*DIGIT "," 1*DIGIT "," 1*DIGIT "," 1*DIGIT
xywhunit      = %x70.69.78.65.6C                            ; "pixel"
              / %x70.65.72.63.65.6E.74                      ; "percent"
```

Examples:

```
xywh=160,120,320,240         # => results in a 320x240 box at x=160 and y=120
xywh=pixel:160,120,320,240   # => results in a 320x240 box at x=160 and y=120
xywh=percent:25,25,50,50     # => results in a 50%x50% box at x=25% and y=25%
```

If the clipping region is pixel-based and the image is multi-resolution (like an ICO file), the fragment MUST be ignored, so that the url represents the entire image. More generally, pixel-clip an image that does not have a single well defined pixel resolution (width and height) is not recommended.

5 Media Fragments Processing

This section defines the different exchange scenarios for the situations explained in section **3 URI fragment and URI query** over the HTTP protocol.

The formal grammar defined in the section **4 Media Fragments Syntax** describes what producers of media fragment should output. It is not taking into account possible percent-encoding that are valid according to *RFC 3986* and the grammar is not a specification of how a media fragment should be parsed. Therefore, section **5.1 Processing Media Fragment URI** defines how to parse media fragment URIs.

5.1 Processing Media Fragment URI

This sections defines how to parse media fragment URIs defined in section **4 Media Fragments Syntax**, along with notes on some of the caveats to be aware of. Implementors are free to use any equivalent technique(s).

5.1.1 Processing name-value components

This section defines how to convert an octet string (from the query or fragment component of a URI) into a list of name-value Unicode string pairs.

1. Parse the octet string according to the namevalues syntax, yielding a list of name-value pairs, where name and value are both octet string. In accordance with *RFC 3986*, the name and value components must be parsed and separated before percent-encoded octets are decoded.
2. For each name-value pair:
   a. Decode percent-encoded octets in name and value as defined by *RFC 3986*. If either name or value are not valid percent-encoded strings, then remove the name-value pair from the list.
   b. Convert name and value to Unicode strings by interpreting them as *UTF-8*. If either name or value are not valid UTF-8 strings, then remove the name-value pair from the list.

Note that the output is well defined for any input.

Examples:

Input	Output	Notes
"t=1"	[("t", "1")]	simple case
"t=1&t=2"	[("t", "1"), ("t", "2")]	repeated name
"a=b=c"	[("a", "b=c")]	"=" in value
"a&b=c"	[("a", ""), ("b", "c")]	missing value
"%74=%6ept%3A%310"	[("t", "npt:10")]	unnecssary percent-encoding
"id=%xy&t=1"	[("t", "1")]	invalid percent-encoding
"id=%E4r&t=1"	[("t", "1")]	invalid UTF-8

While the processing defined in this section is designed to be largely compatible with the parsing of the URI query component in many HTTP server environments, there are incompatible differences that implementors should be aware of:

- "&" is the only primary separator for name-value pairs, but some server-side languages also treat ";" as a separator.
- name-value pairs with invalid percent-encoding should be ignored, but some server-side languages silently mask such errors.
- The "+" character should not be treated specially, but some server-side languages replace it with a space (" ") character.
- Multiple occurrences of the same name must be preserved, but some server-side languages only preserve the last occurrence.

5.1.2 Processing name-value lists

This section defines how to convert a list of name-value Unicode string pairs into the media fragment dimensions.

Given the dimensions defined in section **4.2 Fragment Dimensions**, each has a pair of production rules that corresponds to the name and value component respectively:

Keyword	Dimension
t	**4.2.1 Temporal Dimension**
xywh	**4.2.2 Spatial Dimension**
track	*Media Fragments 1.0 URI (advanced)*
id	*Media Fragments 1.0 URI (advanced)*

1. Initially, all dimensions are undefined.
2. For each name-value pair:
   a. If name matches a keyword in the above table, interpret value as per the corresponding section.
   b. Otherwise, the name-value pair does not represent a media fragment dimension. Validators should emit a warning. User agents must ignore the name-value pair.

Note: Because the name-value pairs are processed in order, the last valid occurence of any dimension is the one that is used.

5.2 Protocol for URI fragment and query resolution in HTTP

The protocol steps to resolve and deliver a media fragment specified as a URI fragment or as a URI query are described through various recipes in the separate Media Fragments 1.0 URI (advanced) document.

6 Media Fragments Semantics

In this section, we discuss how media fragment URIs should be interpreted by user agents. Valid and error cases are presented. In case of errors, we distinguish between errors that can be detected solely based on the media fragment URI and errors that can only be detected when the user agent has information of the media resource (such as the duration of the media resource).

6.1 Valid Media Fragment URIs

For each dimension, a number of valid media fragments and their semantics are presented.

6.1.1 Valid temporal dimension

To describe the different cases for temporal media fragments, we make the following definitions:

- s: the start point of the media, which is always zero (in NPT);
- e: the end point of the media (i.e. duration) and e > 0;
- a: a positive integer, a >= 0;
- b: a positive integer, b >= 0.

Further, as stated in section **4.2.1 Temporal Dimension**, temporal intervals are half-open (i.e. the begin time is considered part of the interval whereas the end time is considered to be the first time point that is not part of the interval). Thus, if we state below that "the media is played from x to y", this means that the frame corresponding to y will not be played.
For t=a,b with a <= b, the following temporal fragments are valid:

- t=a with a < e: media is played from a to e.
- t=,b with b <= e: media is played from s to b.
- t=,b with e < b: media is played from s to e.
- t=a,b with a = 0, b = e: whole media resource is played.
- t=a,b with a < b, a < e and b <= e: media is played from a to b (the normal case).
- t=a,b with a < b, a < e and e < b: media is played from a to e.
- %74=10,20 resolve percent encoding to t=10,20.
- t=%31%30 resolve percent encoding to t=10.
- t=10%2C20 resolve percent encoding to t=10,20.
- t=%6ept:10 resolve percent encoding to t=npt:10.
- t=npt%3a10 resolve percent encoding to t=npt:10.

6.1.2 Valid spatial dimension

To describe the different cases for spatial media fragments, we make the following definitions:

- a: the x coordinate of the spatial region (a >= 0).
- b: the y coordinate of the spatial region (b >= 0).
- c: the width the spatial region (c > 0).
- d: the height of the spatial region (d > 0).
- w: the width of the media resource (w > 0).

- h: the height of the media resource (h > 0).

The following spatial fragments are valid:

- xywh=a,b,c,d with a+c <= w and b+d <= h: the user agent displays a spatial fragment with coordinates (in pixel xywh format) a,b,c,d (the normal pixel case).
- xywh=a,b,c,d with a+c > w, a < w, and b+d < h: the user agent displays a spatial fragment with coordinates (in pixel xywh format) a,b,w-a,d.
- xywh=a,b,c,d with a+c < w, b+d > h, and b < h: the user agent displays a spatial fragment with coordinates (in pixel xywh format) a,b,c,h-d.
- xywh=a,b,c,d with a+c > w, a < w, b+d > h, and b < h: the user agent displays a spatial fragment with coordinates (in pixel xywh format) a,b,w-a,h-d.
- xywh=pixel:a,b,c,d with a+c <= w and b+d <= h: the user agent displays a spatial fragment with coordinates (in pixel xywh format) a,b,c,d (the normal pixel case).
- xywh=percent:a,b,c,d with a+c <= 100, b+d <= 100: the user agent displays a spatial fragment with coordinates (in pixel xywh format) floor(a/w*100), floor(b/h*100), ceil(c/w*100), ceil(d/h*100) (the normal percent case).

The result of doing spatial clipping on a media resource that has multiple video tracks is that the spatial clipping is applied to all tracks.

6.2 Errors detectable based on the URI syntax

Both syntactical and semantical errors are treated similarly. More specifically, the user agent SHOULD ignore name-value pairs causing errors detectable based on the URI syntax. We provide below more details for each dimensions. We look at errors in the different dimensions and their values in the subsequent sub-sections. We start with errors on the more general levels.

6.2.1 Errors on the general URI level

The following list provides the different kind of errors that can occur on the general URI level and how they should be treated:

- *Unknown dimension:* only dimensions described in this specification (i.e. `t`, `xywh`, `track` and `id`) are considered as known dimensions. All other dimensions are considered as unknown. Unknown dimensions SHOULD be ignored by the user agent.
- *Multiple occurrences of the same dimension:* only the last valid occurrence of a dimension (e.g. t=10 in #t=2&t=10) is interpreted and all previous occurrences (valid or invalid) SHOULD be ignored by the user agent. The `track` dimension is an exception to this rule: multiple track dimensions are allowed (e.g. #track=1&track=2 selects both tracks 1 and 2).
- *Combining dimensions:* the `id` dimension combined with a temporal dimension results in multiple occurrences of the temporal dimension (see previous item).

6.2.2 Errors on the temporal dimension

The value cannot be parsed for the temporal dimension or the parsed value is invalid according to the specification. Invalid temporal fragments SHOULD be ignored by the user agent.
Examples:

- t=a,b with a >= b (the case of an empty temporal fragment (a=b) is also considered as an error)
- t=a,
- t=asdf
- t=5,ekj
- t=agk,9
- t='0'
- t=10-20
- t=10:20

- t=10,20,40
- t%3D10 where %3D is equivalent to =; percent encoding does not resolve

6.2.3 Errors on the spatial dimension

The value cannot be parsed for the spatial dimension or the parsed value is invalid according to the specification. Invalid spatial fragments SHOULD be ignored by the user agent.
Examples:

- xywh=4,5,abc,8
- xywh=4,5
- xywh=foo:4,5,7,8
- xywh=percent:400,5,6,8
- xywh=4,5,0,3

6.3 Errors detectable based on information of the source media

Errors that can only be detected when the uiser agent has information of the source media are treated differently. Examples of such information are the duration of a video, the resolution of an image, track information, or the mime type of the media resource (i.e. all information that is not detectable solely based on the URI). Note that a lot of this information is located within the setup information. We provide below more details for each of the dimensions.

6.3.1 Errors on the general level

The following errors can occur on the general level:

- *Non-existent dimension:* a dimension that does not exist in the source media (e.g. temporal clipping on a still image or spatial clipping on an audio file) is considered as a non-existent dimension. If the user agent knows the mime type, it is able to detect non-existent dimensions and SHOULD ignore them.

6.3.2 Errors on the temporal dimension

To describe the different cases for temporal media fragments, we use the definitions from **6.1.1 Valid temporal dimension**. The invalidity of the following temporal fragments can only be detected by the user agent if it knows the duration (for non-existent temporal fragments) and the frame rate of the source media.

- t=a,b with a > 0, a < b, a >= e and b > e: a non-existent temporal fragment, the user agent seeks to the end of the media `e`.
- t=a with a >= e: a non-existent temporal fragment, the user agent seeks to the end of the media `e`.

6.3.3 Errors on the spatial dimension

To describe the different cases for spatial media fragments, we use the definitions from **6.1.2 Valid spatial dimension**. The invalidity of the following spatial fragments can only be detected by the user agent if it knows the resolution of the source media.

- xywh=a,b,c,d with a >= w and/or b >= h: the top-left coordinate (a,b) of the rectangular lies outside the source media and is therefore invalid. The user agent SHOULD ignore this spatial fragment.

7 Notes to Implementors (non-normative)

This section contains notes to implementors. Some of the information here is already stated formally elsewhere in the document, and the reference here is mainly a heads-up. Other items are really outside the scope of this specification, but the notes here reflect what the authors think would be good practice.

The sub-sections are not mutually exclusive. Hence, an implementer of a web browser as a media fragment client should read the sections **7.1 Browsers Rendering Media Fragments**, **7.2 Clients Displaying Media Fragments** and **7.3 All Media Fragment Clients**.

7.1 Browsers Rendering Media Fragments

The pixel coordinates defined in the section **4.2.2 Spatial Dimension** are intended to be identical to the intrinsic width and height defined in HTML5. For spatial URI fragments, the next section describes two distinct use cases, highlighting and cropping. HTML rendering clients, however, are expected to implement cropping as the default rendering mechanism.

7.2 Clients Displaying Media Fragments

When dealing with media fragments, there is a question whether to display the media fragment in context or without context. In general, it is recommended to display a URI fragment in context since it is part of a larger resource. On the other hand, a URI query results in a new resource, so it is recommended to display it as a complete resource without context. The next paragraphs discuss for each axis the context of a media fragment and provides suggestions regarding the visualization of the URI fragment within its context.

For a temporal URI fragment, it is recommended to start playback at a time offset that equals to the start of the fragment and pause at the end of the fragment. When the "play" button is hit again, the resource will continue loading and play back beyond the end of the fragment. When seeking to specific offsets, the resource will load and play back from those seek points. It is also recommended to introduce a "reload" button to replay just the URI fragment. In this way, a URI fragment basically stands for "focusing attention". Additionally, temporal URI fragments could be highlighted on the transport bar.

For a spatial URI fragment, we foresee two distinct use cases: highlighting the spatial region in-context and cropping to the region. In the first case, the spatial region could be indicated by means of a bounding box or the background (i.e. all the pixels that are not contained within the region) could be blurred or darkened. In the second case, the region alone would be presented as a cropped area. How a document author specifies which use case is intended is outside the scope of this specification, we suggest implementors of the specification provide a means for this, for example through attributes or stylesheet elements.

Finally, for track URI fragments, it is recommended to play only the tracks identified by the track URI fragment. If no tracks are specified, the default tracks should be played. Different tracks could be selected using drop-down boxes or buttons, with the selected tracks highlighted during playback. The way the user agent retrieves information regarding the available tracks of a particular resource is out of scope for this specification.

7.3 All Media Fragment Clients

Resolution Order: Where multiple dimensions are combined in one URI fragment request, implementations are expected to first do temporal, id, and track selection on the container level, and then do spatial clipping on the codec level.

Media Fragment Grammar: Note that the grammar for Media Fragment URI only specifies the grammar for features standardised by this specification. If a string does not parse correctly, it does not necessarily mean the URI is wrong, it only means it is not a media fragment URI according to this specification. It may be correct for some extended form, or for a completely different fragment specification method. For this reason, error recovery on syntax errors in media fragment specifiers is unwise.

External Clipping: There is no obligatory resolution method for a situation where a media fragment URI is being used in the context of another clipping method. Formally, it is up to the context embedding the media fragment URI to decide whether the outside clipping method overrides the media fragment URI or cascades, i.e. is defined on the resulting resource. In the absence of strong reasons to do otherwise we suggest cascading. An example is a SMIL element as follows: `<smil:video clipBegin="5" clipEnd="15" src="http://www.example.com/example.mp4#t=100,200"/>`. This should start playback of the original media resource at second 105, and stop at 115.

7.4 Media Fragment Servers

Media type: The media type of a resource retrieved through a URI fragment request is the same as that of the primary resource. Thus, the retrieval of a single frame from a video will result in a one-frame-long video. The retrieval of all the audio tracks from a video resource will result in a video and not a audio resource. When using a URI query approach, media type changes are possible. For example, a spatial fragment from a video at a certain time offset could be retrieved as a jpeg using a specific HTTP "Accept" header in the request.

Synchronisation: Synchronisation between different tracks of a media resource needs to be maintained when retrieving media fragments of that resource. This is true for both URI fragment and URI query retrieval. With URI queries, when transcoding is required, a non-perceivable change in the synchronisation is acceptable.

Embedded Timecodes: When a media resource contains embedded time codes, these need to be maintained for media fragment retrieval, in particular when the URI fragment method is used. When URI queries are used and transcoding takes place, the embedded time codes should remain when they are useful and required.

Reasonable Clipping: Temporal clipping needs to be as close as reasonably possible to what the media fragment specified, and not omit any requested data. "Reasonably close" means the nearest compression entity to the requested fragment that completely contains the requested fragment. For temporal fragments, this means that if a request is made for `http://www.example.org/video.ogv#t=60,100`, but the closest decodable range is `t=58,102` because this is where a packet boundary lies for audio and video, then it will be this range that is returned. The user agent is then capable of displaying only the requested subpart and should also just do that. For some container formats this is a non-issue, because the container format allows specification of logical begin and end.

Reasonable byte ranges: If a single temporal range request would result in a disproportionally large number of byte ranges it may be better for the server to return a redirect to the query form of the media fragment. This situation could happen if the underlying media file is organized in a strange way.

7.5 Media Fragment Web Applications

Media Fragment URIs are only defined on media resources. However, many Web developers that create Web pages with video or audio want to provide their users the ability to jump directly to media fragments - in particular to time offsets in a video - through providing a URI scheme for the Web page. The way in which to realize this without requiring an extra server interaction is by using a URI fragment scheme on the Web page which is parsed by JavaScript and communicates the media fragment to the audio or video resource loader. In HTML5, it would need to change the @src attribute of the appropriate <audio> or <video> element with the appropriate URI fragment and then call the load() function to make the element (re)load the resource with that URI.

A URI scheme for such a Web page may involve ampersand-separated name-value pairs as defined in this specification, e.g. http://example.com/videopage.html#t=60,100&xywh=12,12,42,42. However, the Web developer has to create a scheme that works with the remainder of the Web page fragment addressing functionality. If, for example, the Web page makes use of the ID attributes of the elements on the page for scrolling down on the page, adding media fragment URI addressing to the Web page addressing will fail. For example, if http://example.com/videopage.html#first works and scrolls to an offset on that Web page, http://example.com/videopage.html#first&t=60,100 will not do the same scrolling. The Web developer will then need to parse the fragment parameter and implement the scrolling functionality in JavaScript manually using the scrollTo() or scrollTop() functions.

A References

[RFC 2119] S. Bradner. *Key Words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt.
[RFC 2326] *Real Time Streaming Protocol (RTSP)*. IETF RFC 2326, April 1998. Available at http://www.ietf.org/rfc/rfc2326.txt.
[RFC 2327] *Session Description Protocol (SDP)*. IETF RFC 2327, April 1998. Available at http://www.ietf.org/rfc/rfc2327.txt.

[RFC 2616] *Hypertext Transfer Protocol -- HTTP/1.1*. IETF RFC 2616, June 1999. Available at http://www.ietf.org/rfc/rfc2616.txt.
[RFC 3339] G. Klyne and C. Newman. *Date and Time on the Internet: Timestamps*. IETF RFC 3339, July 2002. Available at http://www.ietf.org/rfc/rfc3339.txt.
[RFC 3533] *The Ogg Encapsulation Format Version 0*. IETF RFC 3533, May 2003. Available at http://www.ietf.org/rfc/rfc3533.txt.
[RFC 3986] T. Berners-Lee and R. Fielding and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. IETF RFC 3986, January 2005. Available at http://www.ietf.org/rfc/rfc3986.txt.
[RFC 5234] D. Crocker. *Augmented BNF for Syntax Specifications: ABNF*. IETF RFC 5234, January 2008. Available at http://tools.ietf.org/html/rfc5234.
[RFC 4288] N. Freed and J. Klensin. *Media Type Specifications and Registration Procedures*. IETF RFC 4288, December 2005. Available at http://www.ietf.org/rfc/rfc4288.txt.
[RFC 5147] E. Wilde and M. Duerst. *URI Fragment Identifiers for the text/plain Media Type*. IETF RFC 5147, April 2008. Available at http://tools.ietf.org/html/rfc5147.
[HTML 4.0] D. Ragett and A. Le Hors and I. Jacobs. *HTML Fragment identifiers*. W3C Recommendation, December 1999. Available at http://www.w3.org/TR/REC-html40/intro/intro.html#fragment-uri.
[HTML 5] Ian Hickson. *HTML5*. W3C Working Draft, May 2011. Available at http://www.w3.org/TR/2011/WD-html5-20110525/.
[SVG 1.1] J. Ferraiolo.*Linking into SVG content: IRI fragments and SVG views*. W3C Recommendation, August 2011. Available at http://www.w3.org/TR/2011/REC-SVG11-20110816/linking.html#LinksIntoSVG.
[SMIL] Sjoerd Mullender. *Synchronized Multimedia Integration Language (SMIL 3.0)*. W3C Recommendation, December 2008. Available at http://www.w3.org/TR/2008/REC-SMIL3-20081201/.
[xpointer] P. Grosso and E. Maler and J. Marsh and N. Walsh. *XPointer Framework*. W3C Recommendation, March 2003. Available at http://www.w3.org/TR/xptr-framework/.
[MPEG-7] *Information Technology - Multimedia Content Description Interface (MPEG-7)*. Standard No. ISO/IEC 15938:2001, International Organization for Standardization(ISO), 2001.
[temporal URI] S. Pfeiffer and C. Parker and A. Pang. *Specifying time intervals in URI queries and fragments of time-based Web resources*. Internet Draft, March 2005. Available at http://annodex.net/TR/draft-pfeiffer-temporal-fragments-03.html.
[CMML] *Continuous Media Markup Language (CMML), Version 2.1*. Internet-Draft, March 2006 http://www.annodex.net/TR/draft-pfeiffer-cmml-03.txt.
[ROE] *Rich Open multitrack media Exposition (ROE)*. Xiph Wiki. Available at http://wiki.xiph.org/index.php/ROE.
[Skeleton] *Ogg Skeleton*. Xiph Wiki. Available at http://wiki.xiph.org/OggSkeleton.
[MPEG-21] *Information Technology - Multimedia Framework (MPEG-21)*. Standard No. ISO/IEC 21000:2002, International Organization for Standardization(ISO), 2002. Available at http://www.chiariglione.org/mpeg/working_documents/mpeg-21/fid/fid-is.zip.
[SMPTE] *SMPTE RP 136 Time and Control Codes for 24, 25 or 30 Frame-Per-Second Motion-Picture Systems*
[ISO Base Media File Format] *Information technology - Coding of audio-visual objects - Part 12: ISO base media file format*. Available at http://standards.iso.org/ittf/PubliclyAvailableStandards/c051533_ISO_IEC_14496-12_2008.zip
[Use cases and requirements for Media Fragments] *Use cases and requirements for Media Fragments*. W3C Working Draft, December 2009. Available at http://www.w3.org/TR/2009/WD-media-frags-reqs-20091217
[Media Fragments 1.0 URI (advanced)] *Media Fragments 1.0 URI (advanced)*. W3C Working Draft, December 2011. Available at http://www.w3.org/TR/2011/WD-media-frags-recipes-20111201/
[UTF-8] *UTF-8, a transformation format of ISO 10646*. Available at http://tools.ietf.org/html/rfc3629
[ECMA-262 5th edition] *ECMA-262 5th edition*. Available at http://www.ecma-international.org/publications/standards/Ecma-262.htm
[Media Annotations] *API for Media Resource 1.0*. W3C Candidate Recommendation, November 2011. Available at http://www.w3.org/TR/2011/CR-mediaont-api-1.0-20111122/
[Web Linking] *Web Linking*. Internet Draft, May 2010. Available at http://tools.ietf.org/html/draft-nottingham-http-link-header-10

B Collected ABNF Syntax for URI (Non-Normative)

```
unichar       = <any Unicode code point>
unistring     = *unichar

; defined in RFC 5234
ALPHA         =  %x41-5A / %x61-7A   ; A-Z / a-z
DIGIT         =  %x30-39 ; 0-9
HEXDIG        =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

; defined in RFC 3986
unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
pct-encoded   = "%" HEXDIG HEXDIG
sub-delims    = "!" / "$" / "&" / "'" / "(" / ")" / "*" / "+" / "," / ";" / "="
pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"
fragment      = *( pchar / "/" / "?" )

; defined in RFC 2326
npt-sec       = 1*DIGIT [ "." *DIGIT ]                     ; definitions taken
npt-hhmmss    = npt-hh ":" npt-mm ":" npt-ss [ "." *DIGIT] ; from RFC 2326
npt-mmss      = npt-mm ":" npt-ss [ "." *DIGIT]
npt-hh        =   1*DIGIT     ; any positive number
npt-mm        =   2DIGIT      ; 0-59
npt-ss        =   2DIGIT      ; 0-59

; defined in RFC 3339
date-fullyear   = 4DIGIT
date-month      = 2DIGIT  ; 01-12
date-mday       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
                          ; month/year
time-hour       = 2DIGIT  ; 00-23
time-minute     = 2DIGIT  ; 00-59
time-second     = 2DIGIT  ; 00-58, 00-59, 00-60 based on leap second
                          ; rules
time-secfrac    = "." 1*DIGIT
time-numoffset  = ("+" / "-") time-hour ":" time-minute
time-offset     = "Z" / time-numoffset

partial-time    = time-hour ":" time-minute ":" time-second
                  [time-secfrac]
full-date       = date-fullyear "-" date-month "-" date-mday
full-time       = partial-time time-offset

date-time       = full-date "T" full-time

; Mediafragment definitions
segment       = mediasegment / *( pchar / "/" / "?" )      ; augmented fragment
                                                           ; definition taken from
                                                           ; RFC 3986

;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Common Prefixes ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;

deftimeformat    = %x6E.70.74                                 ; "npt"
pfxdeftimeformat = %x74.3A.6E.70.74                           ; "t:npt"
smpteformat      = %x73.6D.70.74.65                           ; "smpte"
                  / %x73.6D.70.74.65.2D.32.35                 ; "smpte-25"
                  / %x73.6D.70.74.65.2D.33.30                 ; "smpte-30"
                  / %x73.6D.70.74.65.2D.33.30.2D.64.72.6F.70  ; "smpte-30-drop"
pfxsmpteformat   = %x74.3A.73.6D.70.74.65                     ; "t:smpte"
                  / %x74.3A.73.6D.70.74.65.2D.32.35           ; "t:smpte-25"
                  / %x74.3A.73.6D.70.74.65.2D.33.30           ; "t:smpte-30"
                  / %x74.3A.73.6D.70.74.65.2D.33.30.2D.64.72.6F.70 ; "t:smpte-30-drop"
clockformat      = %x63.6C.6F.63.6B                           ; "clock"
pfxclockformat   = %x74.3A.63.6C.6F.63.6B                     ; "clock"

;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Media Segment ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
mediasegment      = ( timesegment / spacesegment / tracksegment / idsegment )
               *( "&" ( timesegment / spacesegment / tracksegment  / idsegment )
timesegment      = timeprefix "=" timeparam
timeprefix       = %x74                                      ; "t"
timeparam        = npttimedef / smptetimedef / clocktimedef
npttimedef       = [ deftimeformat ":"] ( npttime  [ "," npttime ] ) / ( "," npttime )

npttime       = npt-sec / npt-mmss / npt-hhmmss

smptetimedef  = smpteformat ":"( frametime [ "," frametime ] ) / ( "," frametime )
frametime     = 1*DIGIT ":" 2DIGIT ":" 2DIGIT [ ":" 2DIGIT [ "." 2DIGIT ] ]

clocktimedef  = clockformat ":"( clocktime [ "," clocktime ] ) / ( "," clocktime )
clocktime     = (datetime / walltime / date)
datetime      = date-time                           ; inclusion of RFC 3339

spacesegment  = xywhprefix   "=" xywhparam
xywhprefix    = %x78.79.77.68                       ; "xywh"
xywhparam     = [ xywhunit ":" ] 1*DIGIT "," 1*DIGIT "," 1*DIGIT "," 1*DIGIT
xywhunit      = %x70.69.78.65.6C                    ; "pixel"
              / %x70.65.72.63.65.6E.74              ; "percent"

tracksegment  = trackprefix "=" trackparam
trackprefix   = %x74.72.61.63.6B                    ; "track"
trackparam    = unistring

idsegment   = idprefix "=" idparam
idprefix    = %x69.64                                               ; "id"
idparam     = unistring
```

C Acknowledgements (Non-Normative)

This document is the work of the W3C Media Fragments Working Group. Members of the Working Group are (at the time of writing, and in alphabetical order): Eric Carlson (Apple, Inc.), Chris Double (Mozilla Foundation), Michael Hausenblas (DERI Galway at the National University of Ireland, Galway, Ireland), Jack Jansen (CWI), Philip Jägenstedt (Opera Software), Yves Lafon (W3C), Erik Mannens (IBBT), Thierry Michel (W3C/ERCIM), Guillaume (Jean-Louis) Olivrin (Meraka Institute), Soohong Daniel Park (Samsung Electronics Co., Ltd.), Conrad Parker (W3C Invited Experts), Silvia Pfeiffer (W3C Invited Experts), Nobuhisa Shiraishi (NEC Corporation), David Singer (Apple, Inc.), Thomas Steiner (Google, Inc.), Raphaël Troncy (EURECOM), Davy Van Deursen (IBBT),

The people who have contributed to discussions on public-media-fragment@w3.org are also gratefully acknowledged. In particular: Olivier Aubert, Werner Bailer, Tobias Bürger, Pierre-Antoine Champin, Cyril Concolato, Fantasai, Franck Denoual, Martin J. Dürst, Jean Pierre Evain, Ken Harrenstien, Kilroy Hughes, Ryo Kawaguchi, Wim Van Lancker, Véronique Malaisé, Henrik Nordstrom, Christoph Päper, Yannick Prié, Yves Raimond, Julian Reschke, Geoffrey Sneddon, Felix Sasaki, Jakub Sendor, Philip Taylor, Christian Timmerer, Jorrit Vermeiren, Jeroen Wijering, Munjo Yu and Boris Zbarsky.