**IN THE UNITED STATES DISTRICT COURT**
**FOR THE DISTRICT OF DELAWARE**

|  |  |
|---|---|
| UNILOC 2017 LLC,<br>Plaintiff,<br>v. | C.A. No. 19-cv-179-CFC |
| BITMOVIN, INC.,<br>Defendant. | JURY TRIAL DEMANDED |

## FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Uniloc 2017 LLC ("Uniloc"), by and through the undersigned counsel, hereby files this First Amended Complaint and makes the following allegations of patent infringement relating to U.S. Patent Nos. 6,628,712 (the "'712 patent"), 6,895,118 (the "'118 patent"), 6,519,005 (the "'005 patent") and 6,470,345 (the "'345 patent") (collectively "the Asserted Patents") against Defendant Bitmovin, Inc. ("Bitmovin") and alleges as follows upon actual knowledge with respect to itself and its own acts, and upon information and belief as to all other matters.

## NATURE OF THE ACTION

1. This is an action for patent infringement. Uniloc alleges that Bitmovin has infringed and/or is infringing one or more of the '712 patent, the '118 patent, the '005 patent and the '345 patent, copies of which are attached as Exhibits A-D, respectively.

2. Uniloc alleges that Bitmovin directly infringes and/or has infringed the Asserted Patents by making, using, offering for sale, selling, and/or importing various products and services that: (1) dynamically switch and transcode program video and advertisement videos, (2) perform a method of coding a digital image comprising macroblocks in a binary data stream, (3)

perform a method for motion coding an uncompressed (pixel level) digital video data stream perform a method for providing content via a computer network and a computer system and (4) perform a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, such as the Bitmovin DASH compatible video player. Uniloc seeks damages and other relief for Bitmovin's infringement of the Asserted Patents.

## THE PARTIES

3. Uniloc 2017 LLC is a Delaware corporation having places of business at 1209 Orange Street, Wilmington, Delaware 19801 and 620 Newport Center Drive, Newport Beach, California 92660.

4. Upon information and belief, Bitmovin is a Delaware corporation with a place of business at 301 Howard Street, Suite 1800, San Francisco, California 94015. Bitmovin may be served through its registered agent at The Company Corporation, 251 Little Falls Drive, Wilmington, Delaware 19808.

## JURISDICTION AND VENUE

5. This action for patent infringement arises under the Patent Laws of the United States, 35 U.S.C. § 1 et. seq. This Court has original jurisdiction under 28 U.S.C. §§ 1331 and 1338.

6. This Court has both general and specific personal jurisdiction over Bitmovin because Bitmovin is a Delaware corporation that has committed acts within this District giving rise to this action and has established minimum contacts with this forum such that the exercise of jurisdiction over Bitmovin would not offend traditional notions of fair play and substantial justice. Bitmovin, directly and through subsidiaries and intermediaries (including distributors, retailers, franchisees and others), has committed and continues to commit acts of infringement in

this District by, among other things, making, using, testing, selling, importing, and/or offering for sale products that infringe the Asserted Patents.

7. Venue is proper in this District and division under 28 U.S.C. §§1391(b)-(d) and 1400(b) because Bitmovin is incorporated in this District, transacts business in this District and has committed and continues to commit acts of direct infringement in this District.

## **COUNT I: INFRINGEMENT OF THE '712 PATENT**

8. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

9. Uniloc owns by assignment the entire right, title, and interest in the '712 patent.

10. The '712 patent, titled "Seamless Switching of MPEG Video Streams," issued on September 30, 2003. A copy of the '712 patent is attached as Exhibit A. The priorty date for the '712 patent is November 23, 1999. The inventions of the '712 patent were developed by an inventor at Koninklijke Philips Electronics N.V.

11. Pursuant to 35 U.S.C. § 282, the '712 patent is presumed valid.

12. Claim 4 of the '712 patent reads as follows:

> 4. A method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, said method of switching comprising the steps of:
>
> buffering, in which the data contained in the first and the second input stream are stored,
>
> controlling the storage of the input streams during the buffering step in order to switch, at a switch request, from the first input stream to the second input stream,
>
> transcoding the stream provided by the control step, the transcoding includes controlling occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way.

13. The invention of claim 4 of the '712 patent concerns a novel method for

switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream. ’712 patent at 1:6-9. Such an invention is useful in switching and editing MPEG compressed video signals. ’712 patent at 1:10-11.

14. At the time of invention of the ’712 patent, encoding/decoding systems included a method of switching from a first encoded video sequence to a second one. ’712 patent at 1:15-19. In order to avoid underflow or overflow of the decoded buffer, transcoding of the input streams is used to shift the temporal position of the switching point and to obtain at the output of the transcoders, streams containing an identical entry point and the same decoder buffer characteristics. *Id*. at 1:19-24. This prior art method has several major drawbacks. According to the background art, the output bit rate of each transcoder is equal to its input bit rate, which makes the switching method not very flexible. *Id*. at 1:15-28. Finally, the solution of the background art is rather complex and costly to implement as the switching device needs two transcoders. *Id*. at 1:32-35.

15. As demonstrated below, the claimed invention of claim 4 of the ’712 patent provides a technological solution to the problem faced by the inventors—transcoding the stream provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way. This technological solution of claim 4 of the ’712 patent provides an improved method of switching between encoded video streams that is “both flexible and easy to implement” and overcomes the disadvantages of the prior art. *Id*. at 1:38-40. For example, the solution of the ’712 patent allows switching from a first compressed data stream encoded at a bit rate R1 to a second compressed data stream encoded at a bit rate R2, the output stream resulting from the switch being encoded again, using the transcoding system, at a bit rate R where R may

be different from R1 and R2. *Id.* at 1:52-59. Thus, the patented solution has greater flexibility than the prior art and its "implementation will be less complex and less expensive" than the prior art in addition to being more flexible. *Id.* at 1:39-40, 1:52-59, 2:9-10, 2:33.

16. A person of ordinary skill in the art reading the '712 patent and its claims would understand that the patent's disclosure and claims are drawn to solving a specific, technical problem arising in the field of video compression. In particular, the present invention relates to the technical problem involved in switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, and is applicable, for example, to switching and editing MPEG compressed video signals. *Id.* at 1:6-12.

17. As detailed in the specification, the invention of claim 4 of the '712 patent provides a technological solution to the specific technological problems faced by the inventor that existed at the time of the invention. First the specification describes the prior art and the drawbacks associated with the prior art:

> International patent application WO 99/05870 describes a method and device of the above kind. This patent application relates, in encoding/decoding systems, to an improved method of switching from a first encoded video sequence to a second one. In order to avoid underflow or overflow of the decoded buffer, a transcoding of the input streams is used to shift the temporal position of the switching point and to obtain at the output of the transcoders, streams containing an identical entry point and the same decoder buffer characteristics.
>
> The previously described method has several major drawbacks. According to the background art, the output bit rate of each transcoder is equal to its input bit rate, which makes the switching method not very flexible. Moreover, said method implies that the first picture of the second video sequence just after the switch will be an Intra-coded (I) picture.
>
> Finally, the solution of the background art is rather complex and costly to implement as the switching device needs two transcoders.

'712 patent at 1:15-35.

18. In light of the drawbacks with the prior art, the inventor of the ’712 patent claimed a new method where transcoding of the output stream is provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way:

> To prevent overflow or underflow of this buffer, a regulation REG is performed; the buffer occupancy is controlled by a feedback to the DCT coefficient quantization. When switching from a video sequence encoded at a bit rate R**1** to another one that has been separately encoded at a bit rate R**2**, the respective decoder buffer delays at the switching point do not match. The role of the transcoder is to compensate the difference between these buffer delays in order to provide the output stream OS in a seamless way. Furthermore, the encoded bit rate R of the output stream can be chosen by the user.

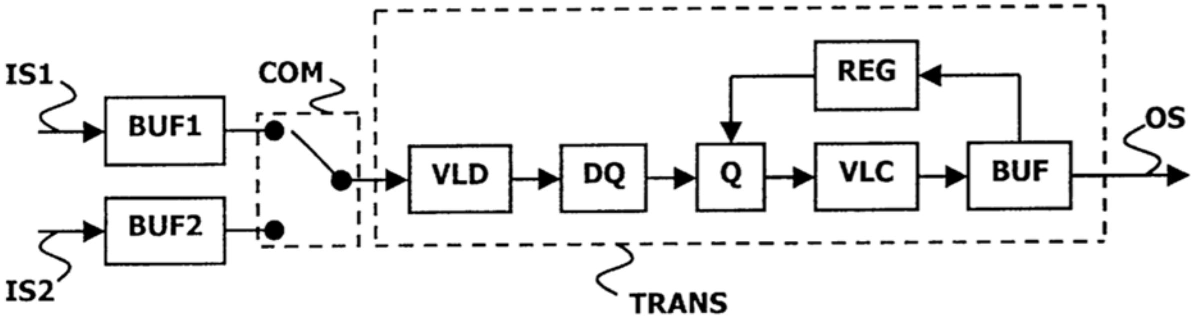


FIG. 2

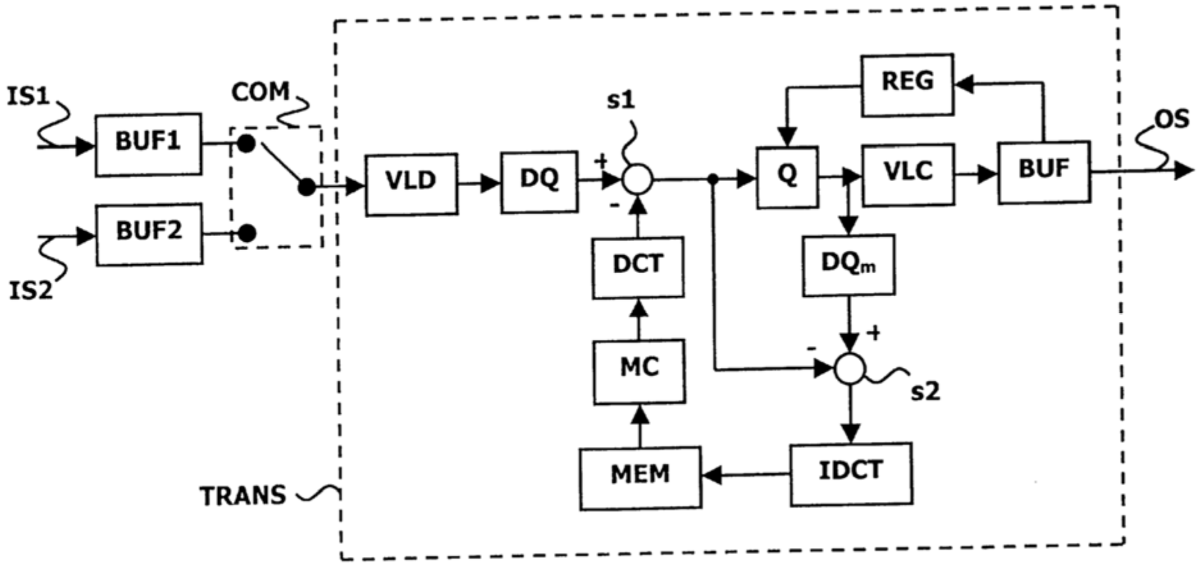


FIG. 3

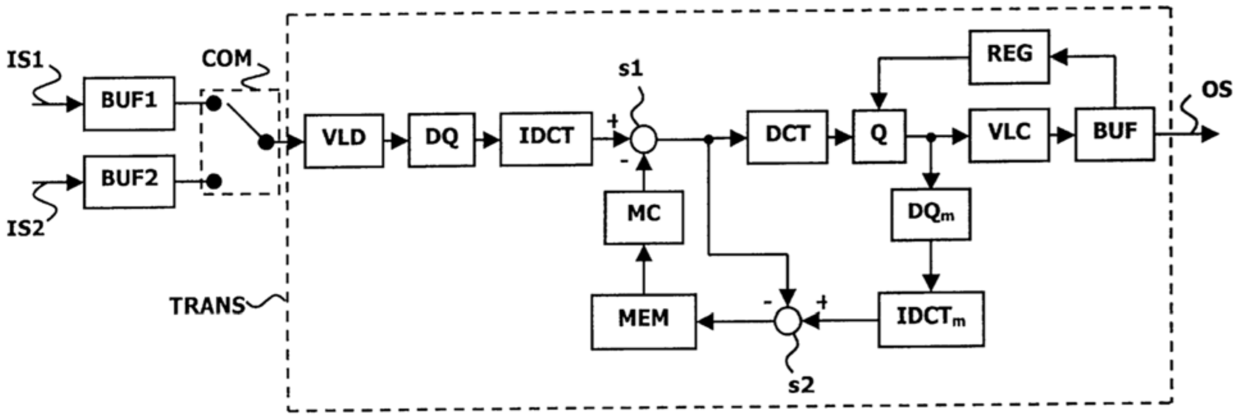


FIG. 4

’712 patent at 4:15-25, Figs. 2-4.

19. The claimed invention of claim 4 of the ’712 patent improves the functionality of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream. ’712 patent at 1:5-2:37; 2:66-4:32. The claimed invention of claim 4 of the ’712 patent also was not well-understood, routine or conventional at

the time of invention. Rather, the claimed invention was a departure from the conventional way of switching from a first encoded video sequence to a second one.

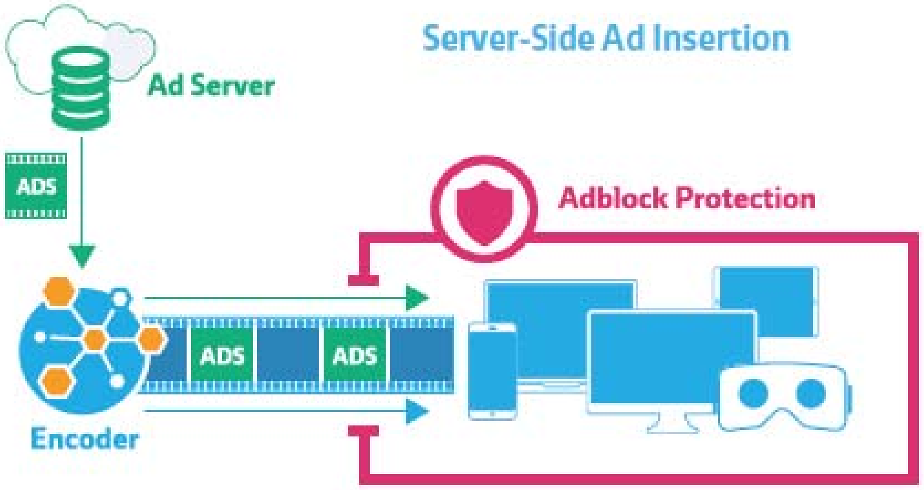
20. In light of the foregoing, a person of ordinary skill in the art would understand that the claimed subject matter of the ’712 patent presents advancements in the field of image compression. A person of ordinary skill in the art would understand that claim 4 of the ’712 patent is directed to a method of transcoding a stream provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way. Moreover, a person of ordinary skill in the art would understand that claim 4 of the ’712 patent contains the inventive concept of transcoding a stream provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way.

21. Upon information and belief, Bitmovin has directly infringed at least claim 4 of the ’712 patent by making, using, testing, selling, offering for sale, importing and/or licensing in the United States without authority products and services that dynamically switch and transcode program videos and advertisement videos (collectively “the ’712 Accused Infringing Devices”) in an exemplary manner as described below.

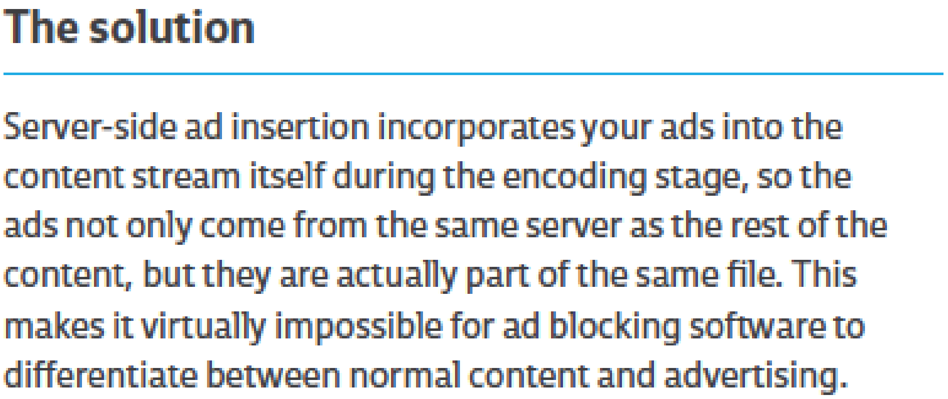
22. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, practice the method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream.

23. The ’712 Accused Infringing Devices implement a Server-Side Ad Insertion (SSAI) algorithm that switches from the programming video to the ad video at the beginning of

an ad break and from the ad video back to the programing video at the end of an ad break. The output video is a compressed video data stream encoded in, for example, the H.264 standard.



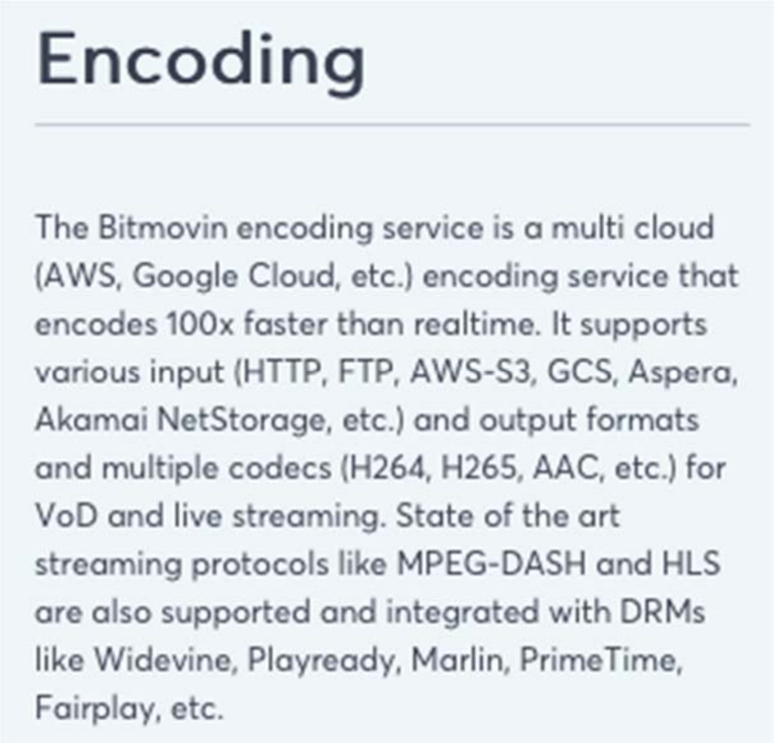
**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.



## The solution

Server-side ad insertion incorporates your ads into the content stream itself during the encoding stage, so the ads not only come from the same server as the rest of the content, but they are actually part of the same file. This makes it virtually impossible for ad blocking software to differentiate between normal content and advertising.

**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

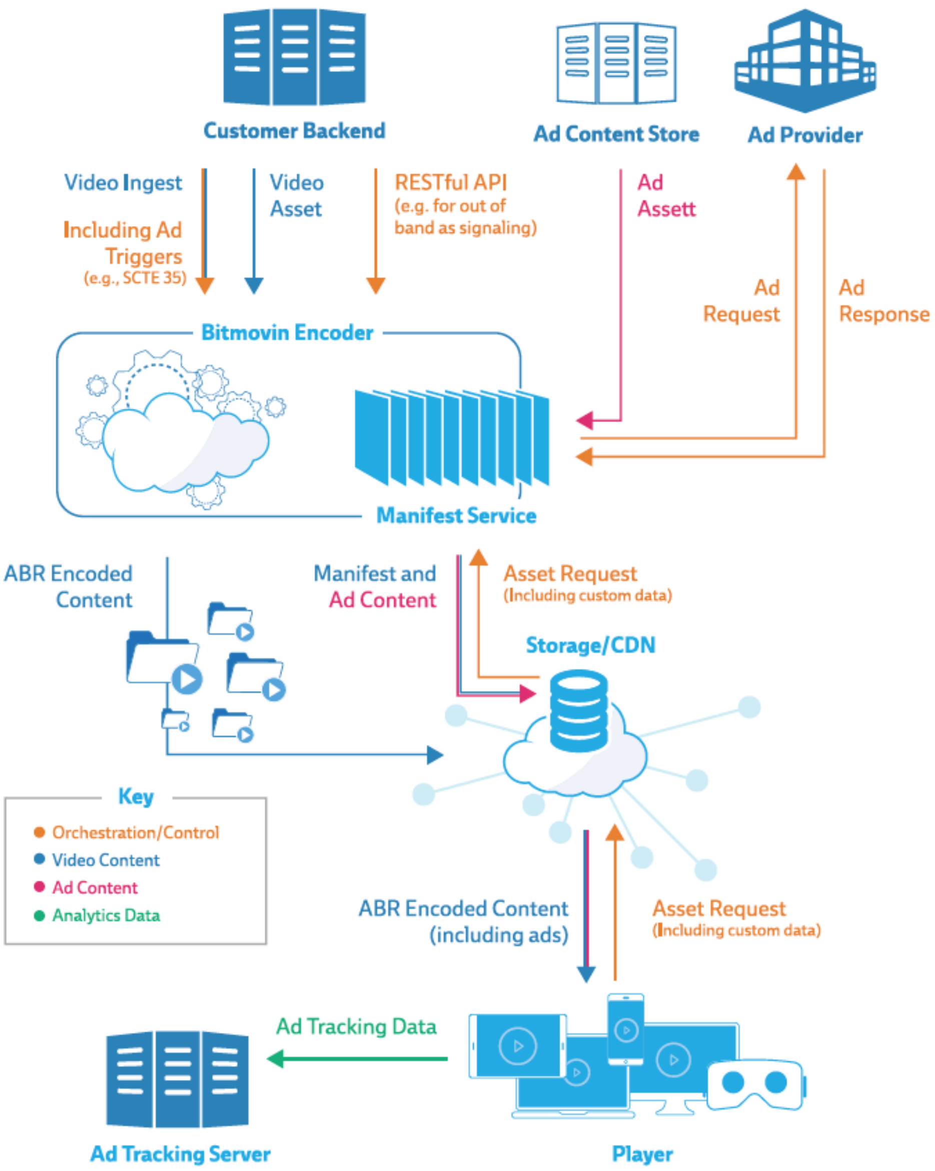


# Encoding

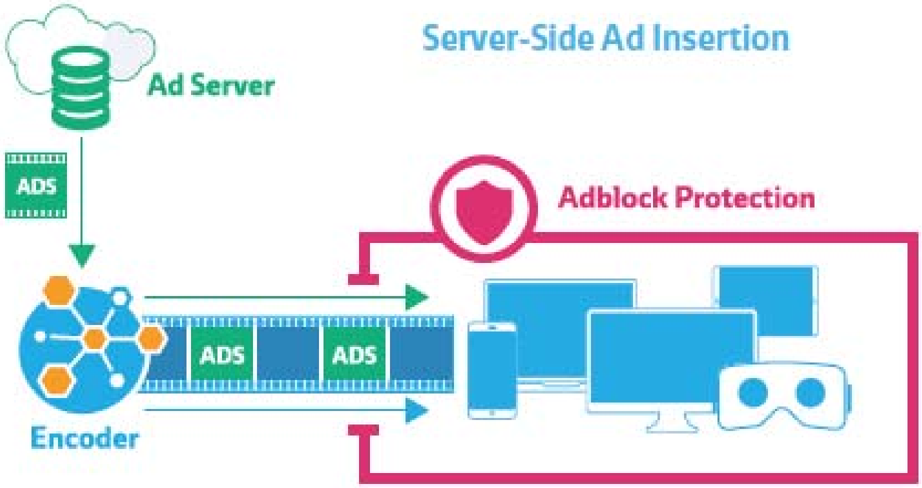
The Bitmovin encoding service is a multi cloud (AWS, Google Cloud, etc.) encoding service that encodes 100x faster than realtime. It supports various input (HTTP, FTP, AWS-S3, GCS, Aspera, Akamai NetStorage, etc.) and output formats and multiple codecs (H264, H265, AAC, etc.) for VoD and live streaming. State of the art streaming protocols like MPEG-DASH and HLS are also supported and integrated with DRMs like Widevine, Playready, Marlin, PrimeTime, Fairplay, etc.

**Source:** https://bitmovin.com/docs/encoding/api-reference#/reference/encoding, last accessed Nov. 29, 2018.

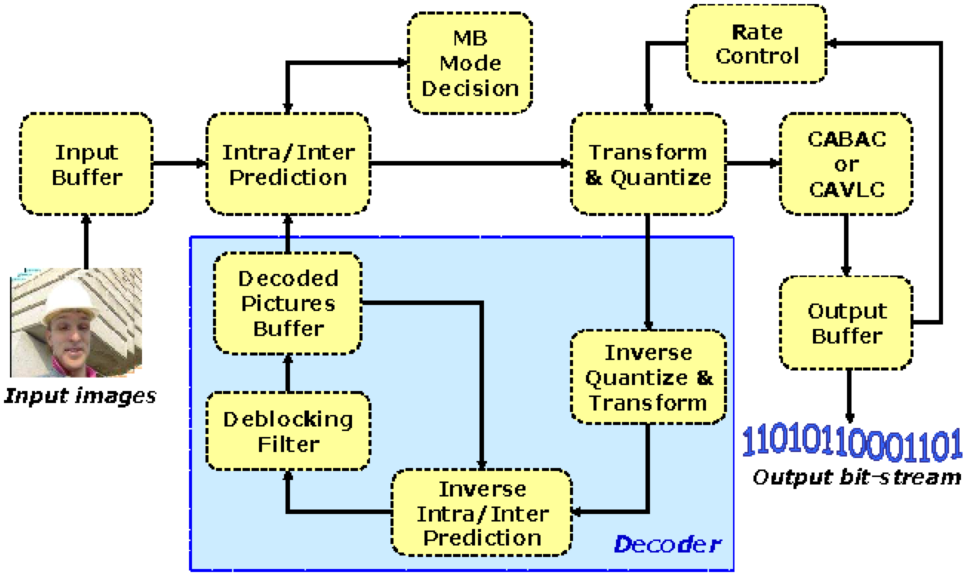
24. The ’712 Accused Infringing Devices buffer and store the data contained in the first and second input streams. The programming videos, both live and on demand, and the ad videos are buffered, in which they are also stored before they are encoded by Bitmovin’s encoding products.



**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

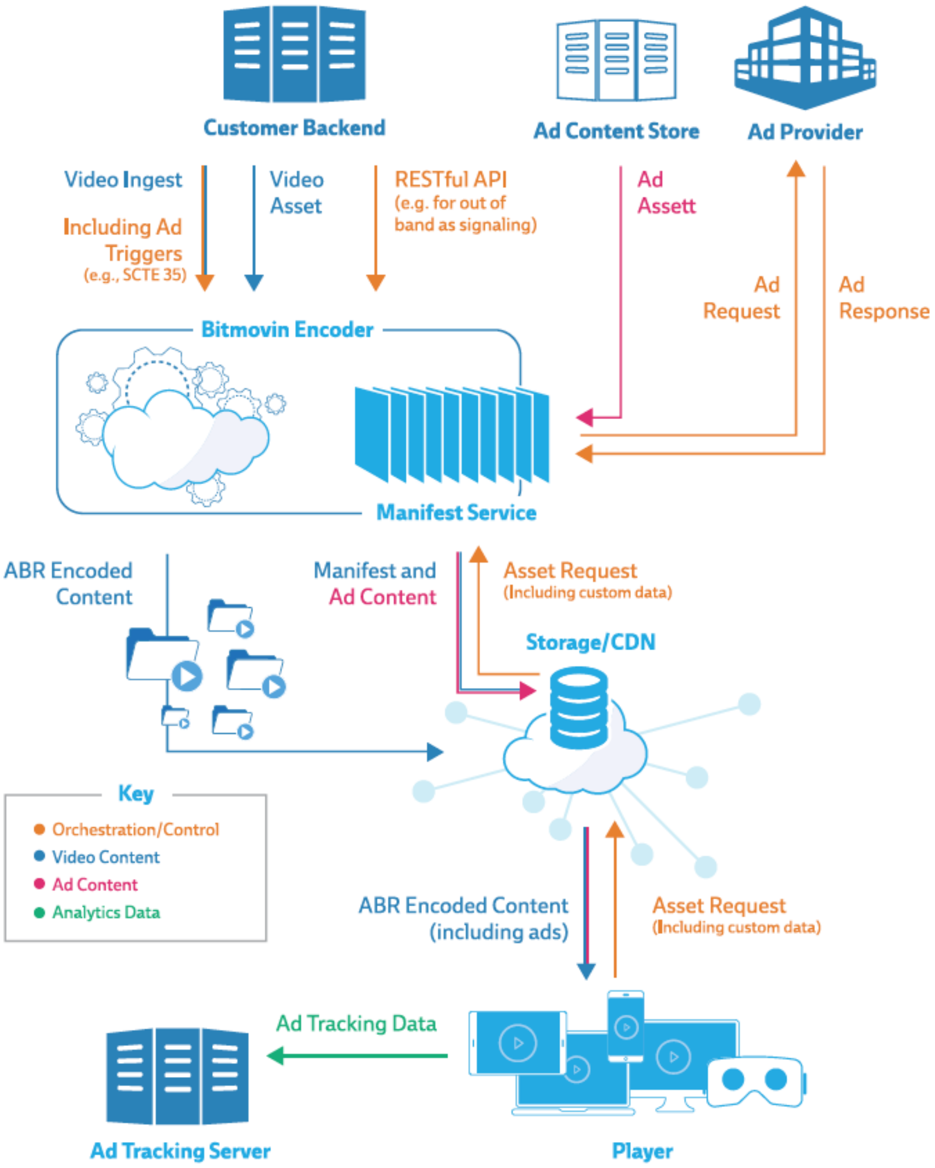


**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

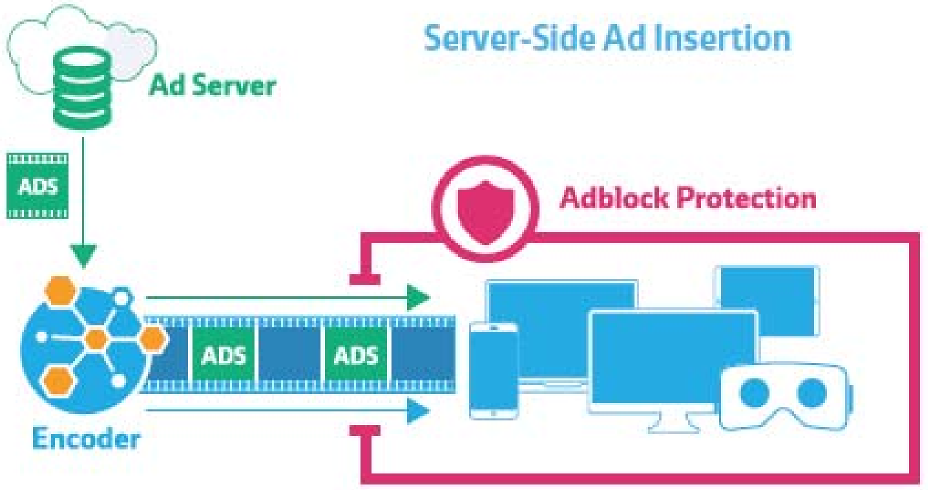


**Source:** https://ieeexplore.ieee.org/document/7080395, last accessed Nov. 29, 2018.

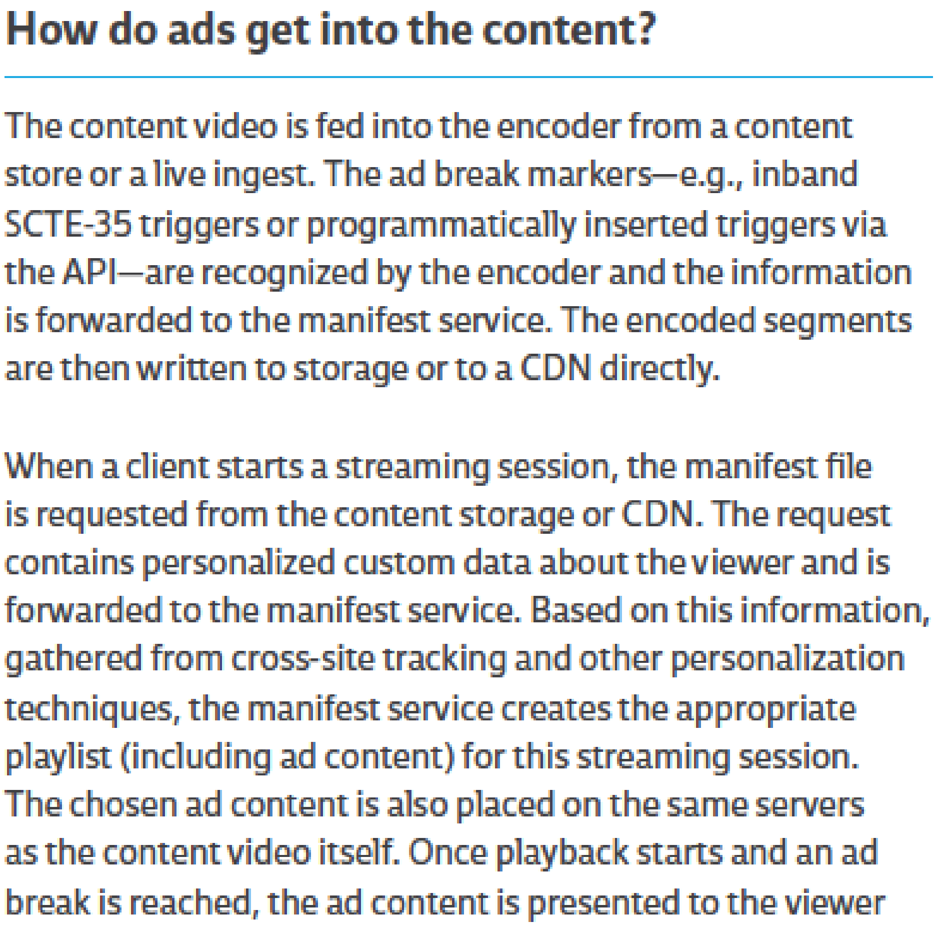
25. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, control the storage of the input streams in the buffer system in order to switch, at a switch request, from the first input stream to the second input stream for its server-side dynamic ad insertion or ad stitching.



**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.



**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

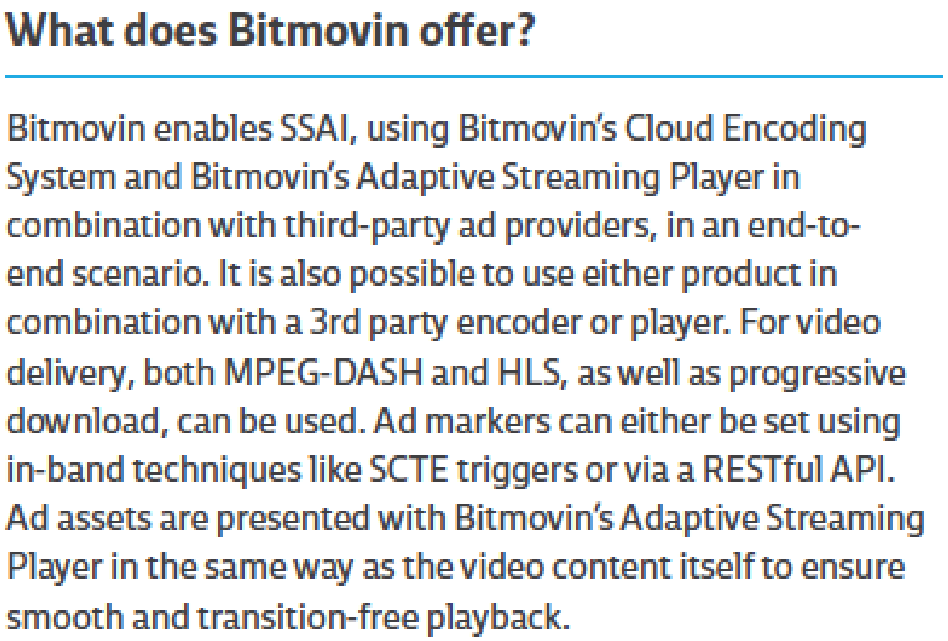


## How do ads get into the content?

The content video is fed into the encoder from a content store or a live ingest. The ad break markers—e.g., inband SCTE-35 triggers or programmatically inserted triggers via the API—are recognized by the encoder and the information is forwarded to the manifest service. The encoded segments are then written to storage or to a CDN directly.

When a client starts a streaming session, the manifest file is requested from the content storage or CDN. The request contains personalized custom data about the viewer and is forwarded to the manifest service. Based on this information, gathered from cross-site tracking and other personalization techniques, the manifest service creates the appropriate playlist (including ad content) for this streaming session. The chosen ad content is also placed on the same servers as the content video itself. Once playback starts and an ad break is reached, the ad content is presented to the viewer

**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

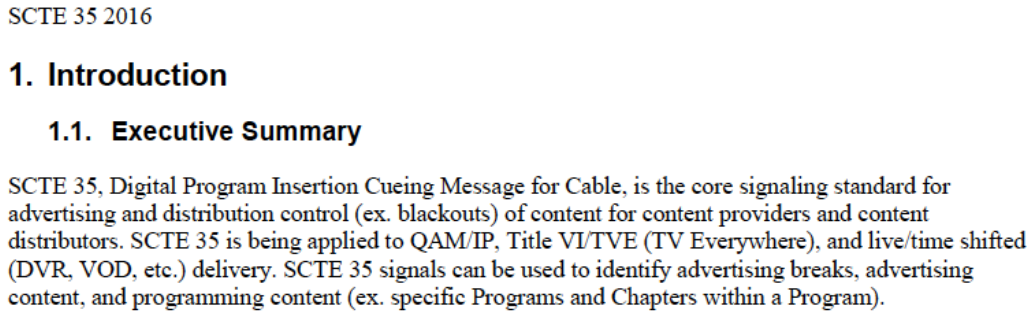


## What does Bitmovin offer?

Bitmovin enables SSAI, using Bitmovin's Cloud Encoding System and Bitmovin's Adaptive Streaming Player in combination with third-party ad providers, in an end-to-end scenario. It is also possible to use either product in combination with a 3rd party encoder or player. For video delivery, both MPEG-DASH and HLS, as well as progressive download, can be used. Ad markers can either be set using in-band techniques like SCTE triggers or via a RESTful API. Ad assets are presented with Bitmovin's Adaptive Streaming Player in the same way as the video content itself to ensure smooth and transition-free playback.

**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

26. The switch request in the '712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm use the Society of Cable Telecommunications Engineers (SCTE) triggers for identifying an impending ad break. The Society of Cable Telecommunications Engineers standard 35 defines a family of markers (or triggers), such #EXT-X-SCTE35, #OATCLS-SCTE35, #ASSET, #CUE-OUT, #CUE-OUT-CONT, and #CUE-IN that are associated with switching between different video streams.



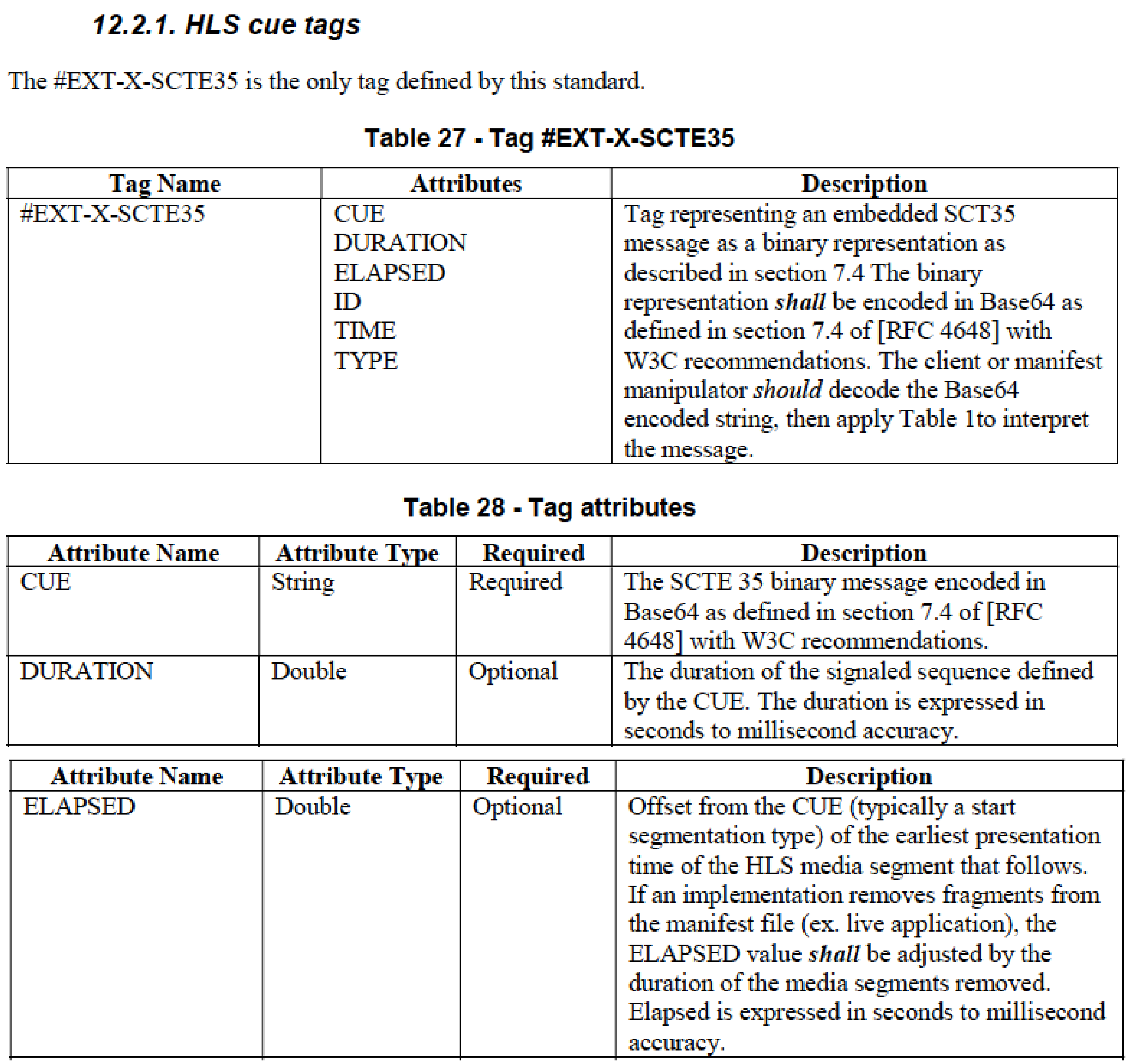
SCTE 35 2016

# 1. Introduction

## 1.1. Executive Summary

SCTE 35, Digital Program Insertion Cueing Message for Cable, is the core signaling standard for advertising and distribution control (ex. blackouts) of content for content providers and content distributors. SCTE 35 is being applied to QAM/IP, Title VI/TVE (TV Everywhere), and live/time shifted (DVR, VOD, etc.) delivery. SCTE 35 signals can be used to identify advertising breaks, advertising content, and programming content (ex. specific Programs and Chapters within a Program).

**Source:** https://www.scte.org/SCTEDocs/Standards/SCTE%2035%202016.pdf, Page 7, last accessed Oct. 1, 2018.



### *12.2.1. HLS cue tags*

The #EXT-X-SCTE35 is the only tag defined by this standard.

**Table 27 - Tag #EXT-X-SCTE35**

| Tag Name | Attributes | Description |
|---|---|---|
| #EXT-X-SCTE35 | CUE<br>DURATION<br>ELAPSED<br>ID<br>TIME<br>TYPE | Tag representing an embedded SCT35 message as a binary representation as described in section 7.4 The binary representation ***shall*** be encoded in Base64 as defined in section 7.4 of [RFC 4648] with W3C recommendations. The client or manifest manipulator *should* decode the Base64 encoded string, then apply Table 1to interpret the message. |

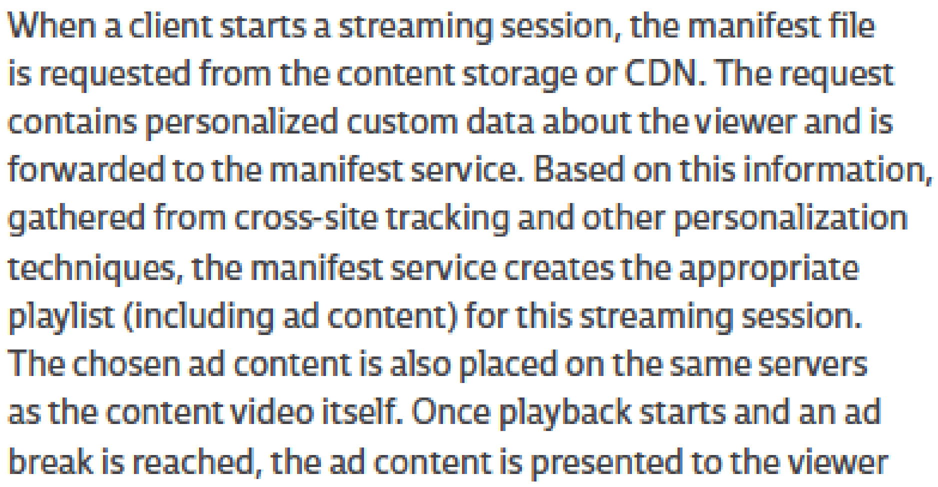
**Table 28 - Tag attributes**

| Attribute Name | Attribute Type | Required | Description |
|---|---|---|---|
| CUE | String | Required | The SCTE 35 binary message encoded in Base64 as defined in section 7.4 of [RFC 4648] with W3C recommendations. |
| DURATION | Double | Optional | The duration of the signaled sequence defined by the CUE. The duration is expressed in seconds to millisecond accuracy. |
| ELAPSED | Double | Optional | Offset from the CUE (typically a start segmentation type) of the earliest presentation time of the HLS media segment that follows. If an implementation removes fragments from the manifest file (ex. live application), the ELAPSED value ***shall*** be adjusted by the duration of the media segments removed. Elapsed is expressed in seconds to millisecond accuracy. |

**Source:** https://www.scte.org/SCTEDocs/Standards/SCTE%2035%202016.pdf, Pages 70-71, last accessed Oct. 1, 2018.

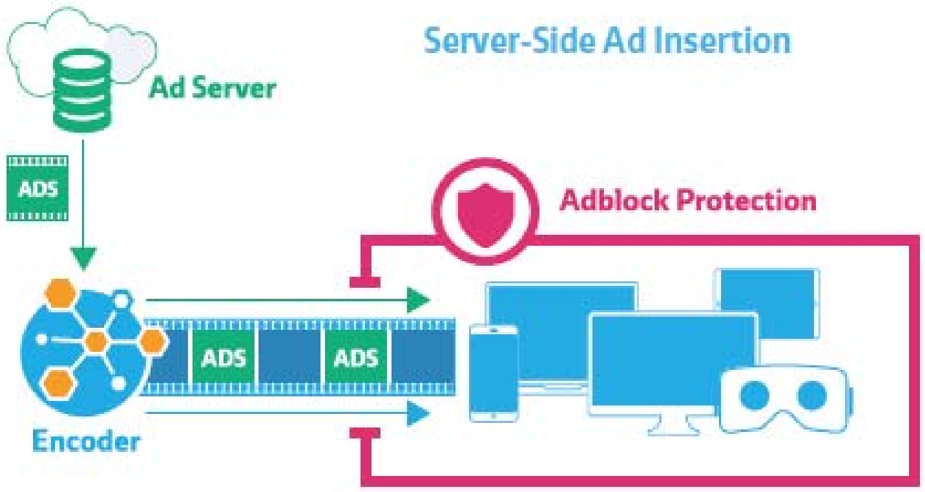
27. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, provide a transcoding system (TS) including a quantization block and a buffer, wherein occupancy of the buffer in the transcoding system is controlled by feedback to the quantization block to provide the output stream in a seamless way from the output of the commutation device.

28. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, transcode the compressed ad videos retrieved from 3rd party ad servers so that the ad break and ad content are “presented to the viewer in a smooth and transition-free manner.”



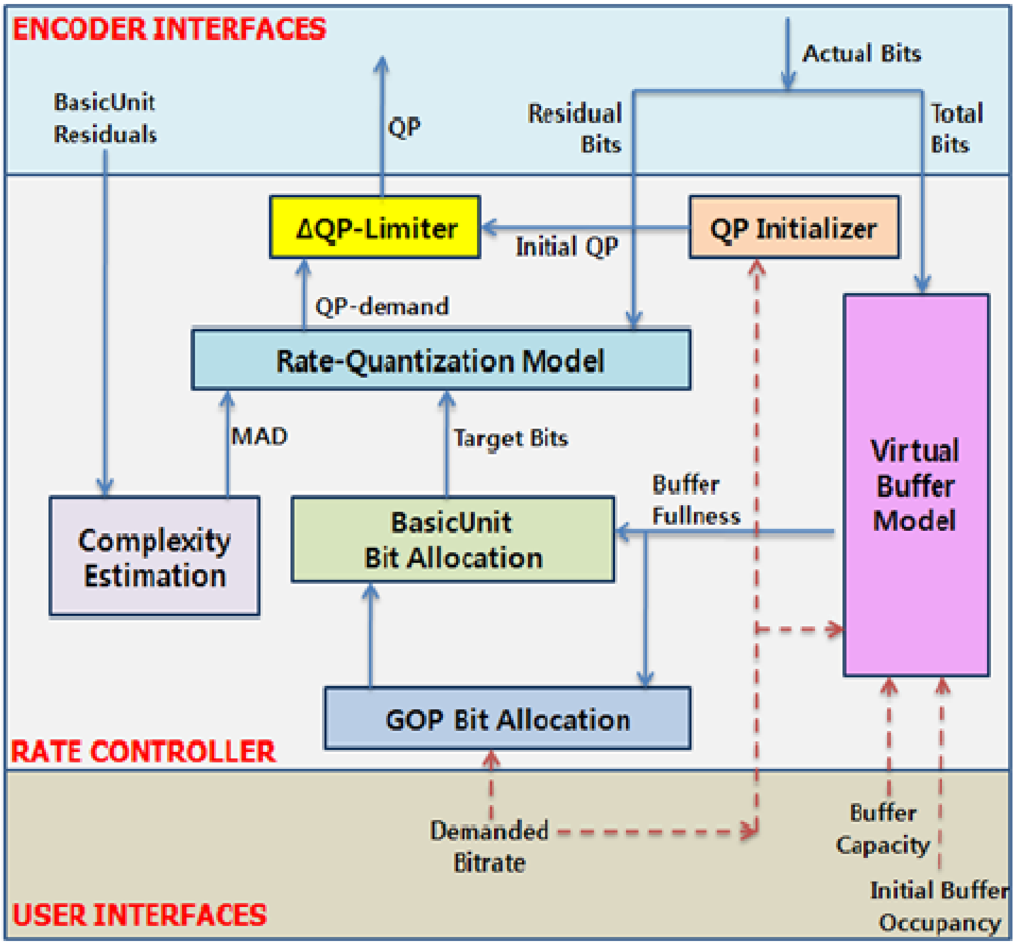
When a client starts a streaming session, the manifest file is requested from the content storage or CDN. The request contains personalized custom data about the viewer and is forwarded to the manifest service. Based on this information, gathered from cross-site tracking and other personalization techniques, the manifest service creates the appropriate playlist (including ad content) for this streaming session. The chosen ad content is also placed on the same servers as the content video itself. Once playback starts and an ad break is reached, the ad content is presented to the viewer

**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.



**Source:** https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

29. The H.264 video codec supported in the ’712 Accused Infringing Devices controls occupancy of the encoded bit stream buffer by feedback to DCT coefficient quantization as part of rate control and rate distortion optimization in the video encoders.



**Source:** https://www.researchgate.net/figure/Rate-control-structure-of-H264-AVC-JM-reference-model_fig1_260585793, last accessed Oct. 1, 2018.

30. Bitmovin has thus infringed at least claim 4 of the ’712 patent by making, using, testing, selling, offering for sale, importing and/or licensing the ’712 Accused Infringing Devices, and operating them such that all steps of at least claim 4 are performed.

31. Bitmovin's acts of direct infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin's wrongful acts in an amount subject to proof at trial.

## **COUNT II: INFRINGEMENT OF THE '118 PATENT**

32. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

33. The '118 patent, titled "Method Of Coding Digital Image Based on Error Concealment," issued on May 17, 2005. A copy of the '118 patent is attached as Exhibit B. The priority date for the '118 patent is March 6, 2001. The inventions of the '118 patent were developed by inventors at Koninklijke Philips Electronics N.V.

34. Pursuant to 35 U.S.C. § 282, the '118 patent is presumed valid.

35. Claim 1 of the '118 patent addresses a technological problem indigenous to coding macroblocks in a binary digital stream where certain macroblocks have been excluded.

36. Claim 1 of the '118 patent reads as follows:

> 1. A method of coding a digital image comprising macroblocks in a binary data stream, the method comprising:
>
> an estimation step, for macroblocks, of a capacity to be reconstructed via an error concealment method,
>
> a decision step for macroblocks to be excluded from the coding, a decision to exclude a macroblock from coding being made on the basis of the capacity of such macroblock to be reconstructed,
>
> characterized in that it also includes a step of inserting a resynchronization marker into the binary data stream after the exclusion of one or more macroblocks.

37. The invention of claim 1 of the '118 patent concerns a novel method for digital coding of macroblocks within a data stream.

38. Just prior to the invention of the ’118 patent, in June 1999, a then novel method for coding involved the exclusion of certain macroblocks in a digital image based upon the capacity of the macroblocks to be reconstructed via error concealment (“the June 1999 Method”). ’118 patent at 1:14-21. In the June 1999 Method, the excluded macroblocks were replaced with “uncoded blocks with constant blocks, black blocks for example, subsequently detected by the receiver.” ’118 patent at 1:21-25. Alternatively, the June 1999 Method provided for allocating bits to communicate the address of the excluded blocks in interceded macroblocks that were not excluded. ’118 patent at 1:26-32.

39. Both means of replacing the excluded blocks in the June 1999 Method suffered from significant drawbacks. For example, if constant blocks or black blocks were used as replacements for the excluded macroblocks there would be “graphical errors on most receivers.” ’118 patent at 1:62-67. Likewise, allocating bits to communicate the address of excluded blocks gave “rise to graphical ‘lag’ errors of image elements if macroblocks have been excluded.” ’118 patent at 1:56-62.

40. As demonstrated below, the claimed invention of claim 1 of the ’118 patent provides a technological solution to the problem faced by the inventors— using resynchronization markers after the exclusion of a macroblock rather than replacing macroblocks with constant blocks, black blocks or bits allocated to communicate the address of the excluded blocks. This technological solution resulted in reduction in lag and graphical errors and improved bandwidth because of a reduction in the binary data stream.

41. As detailed in the specification, the invention of claim 1 of the ’118 patent provides a technological solution to the specific technological problems faced by the inventors

that existed at the time of the invention. First, the specification describes the June 1999 Method and the drawbacks associated with that method.

> A coding method of such type is known from the document "Geometric-Structure-Based Error Concealment with Novel Applications in Block-Based Low-Bit-Rate Coding" by W. Zeng and B. Liu in IEEE Transactions on Circuits and Systems For Video Technology, Vol. 9, No. 4, Jun. 1999. That document describes exclusions of blocks belonging to macroblocks, block combination, said macroblocks being capable of being intercoded or intracoded. That document proposes harmonizing this block exclusion with video coding standards, either, in a **first solution**, by replacing uncoded blocks with constant blocks, black blocks for example, subsequently detected by the receiver, or, in a **second solution**, by modifying the word that defines which blocks are coded within a macroblock, such modification taking place at the same time as a modification of the address words of the macroblocks when all the blocks in a macroblock are excluded. A certain number of bits are allocated to communicate the address of the excluded blocks in the interceded macroblocks.

'118 patent at 1:14-31 (emphasis added).

42. Both of these means of dealing with the excluded macroblocks in the June 1999 Method were disadvantageous and suffered from serious drawbacks that thwarted the purpose of excluding macroblocks (i.e., to further compress the data stream):

> In this case it is therefore impossible to change the addresses of the macroblocks or indicate which blocks are not coded, according to the **second solution** proposed in the document cited in the foregoing. All macroblocks are thus decoded and placed sequentially, giving rise to graphical "lag" errors of image elements if macroblocks have been excluded. The **first solution** proposed in the document cited involves detection by the decoder of the constant blocks replacing the excluded blocks. No provision for such detection is made in the MPEG-4 syntax, and this will cause graphical errors on most receivers.

'118 patent at 1:56-67 (emphasis added).

43. In light of the drawbacks with the June 1999 Method, the inventors of the '118 patent claimed a new method where resynchronization markers included in header elements were

used instead of constant blocks, black blocks and bits allocated to communicate the address of the excluded blocks:

> It is an object of the present invention to suggest a coding method that includes an exclusion of macroblocks having a certain capacity to be reconstructed from the coding compatible with coding standards which include point resynchronization means.
>
> Indeed, a coding method as defined in the introductory paragraph is characterized according to the invention in that it also includes a step of inserting a resynchronization marker into the binary data stream after the exclusion of one or more macroblocks.
>
> The resynchronization marker represents a certain number of bits in the data stream (at least between 17 and 23 bits). It is a further object of the present invention to reduce the binary data stream associated with the transmission of digital images by excluding macroblocks**.**

’118 patent at 2:1-15 (emphasis added).

44. The reduction in the data stream using the claimed method—as opposed to the June 1999 Method which added constant blocks, black blocks and other bits for excluded macroblocks—is depicted in Figure 2 and described in the specification:

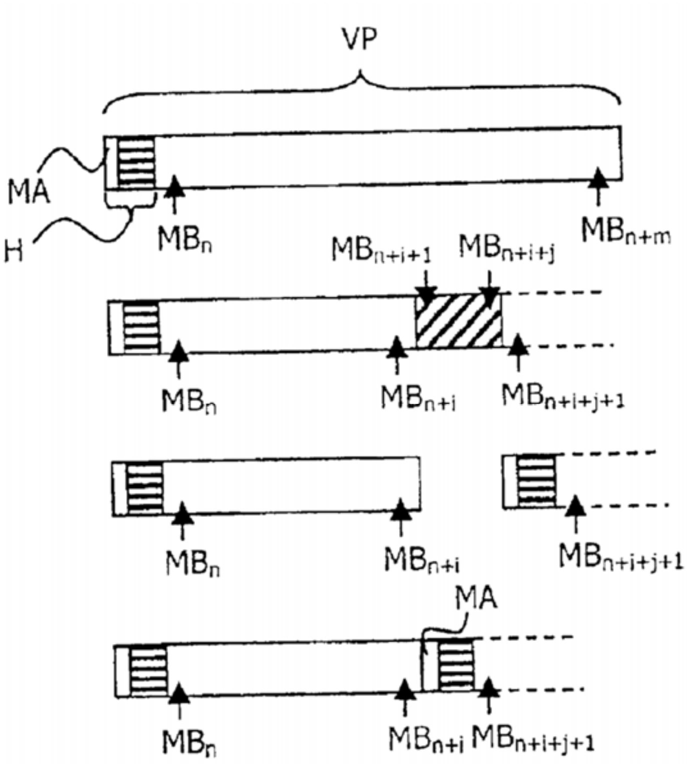


FIG.2a

FIG.2b

FIG.2c

FIG.2d

> The resulting binary data stream in such case is shown in FIG. 2*d*. A resynchronization marker MA and the associated header element have been inserted in the stream at the point where the first one of the excluded macroblocks should have been, and before macroblock $MB_{n+i+j+1}$. Here, the reduction in the size of the binary data stream caused by the insertion of resynchronization marker MA and the associated header element is not zero according to FIG. **2**: the bloc representing excluded macroblocks $MB_{n+i+1}$ to $MB_{n+i+j}$is larger than the size of the inserted header element.
>
> * * *
>
> Since the binary data stream includes coded data of a digital image comprising macroblocks, said binary data stream being such that macroblocks $MB_{n+i+1}$ to $MB_{n+i+j}$ are not coded in the binary data stream for at least one point in the binary data stream and since such uncoded macroblocks are capable of being reconstructed by an error concealment method, said binary data stream is thus characterized according to the invention in that a resynchronization marker MA is present in the binary data stream at the location in the binary data stream where the macroblocks are not coded.

’118 patent at 5:37-46.

45. The claimed invention of claim 1 of the ’118 patent improves the functionality of coding macroblocks in a binary digital stream where certain macroblocks have been excluded. The claimed invention of claim 1 of the ’118 patent also was not well-understood, routine or

conventional at the time of invention. Rather, the claimed invention was a departure from the conventional way of performing coding macroblocks in a binary digital stream where certain macroblocks have been excluded.

46. A person of ordinary skill in the art reading claim 1 of the ’118 patent and the corresponding specification would understand that claim 1 improves the functionality of coding macroblocks in a binary digital stream where certain macroblocks have been excluded. This is because, as noted above, the June 1999 Method suffered from drawbacks including (1) lag errors; (2) graphical errors; and (3) no reduction in the size of the data stream because of the use of constant blocks, black blocks and allocating bits to communicate the address of the excluded blocks. A person of ordinary skill in the art would further understand that the claimed invention of claim 1 of the ’118 patent resolved these problems by using resynchronization markers in a way they had not been used before—as replacements for excluded blocks.

47. A person of ordinary skill in the art reading claim 1 of the ’118 patent and the corresponding specification would further understand that claim 1 of the ’118 patent represents a departure from convention by (1) coding a data stream with excluded macroblocks in a way that is different from the recent June 1999 Method and (2) using resynchronization markers in a manner that had not been used before—as replacements for excluded macroblocks.

48. In light of the foregoing, a person of ordinary skill in the art reading the ’118 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in achieving more efficient video compression. Moreover, a person of ordinary skill in the art would understand that the claimed subject matter of the ’118 patent presents advancements in the field of digital image coding.

49. In light of the foregoing, a person of ordinary skill in the art would understand that claim 1 of the ’118 patent is directed to a method of coding macroblocks in a binary digital stream where certain macroblocks have been excluded. Moreover, a person of ordinary skill in the art would understand that claim 1 of the ’118 patent contains the inventive concept of using resynchronization markers after the exclusion of a macroblock rather than replacing macroblocks with constant blocks, black blocks or bits allocated to communicate the address of the excluded blocks.

50. Upon information and belief, Bitmovin makes, uses, offers for sale, and/or sells in the United States and/or imports into the United States products and services such as H.264 encoders that practice a method for coding a digital image comprising macroblocks in a binary data stream (collectively the “’118 Accused Infringing Devices”).

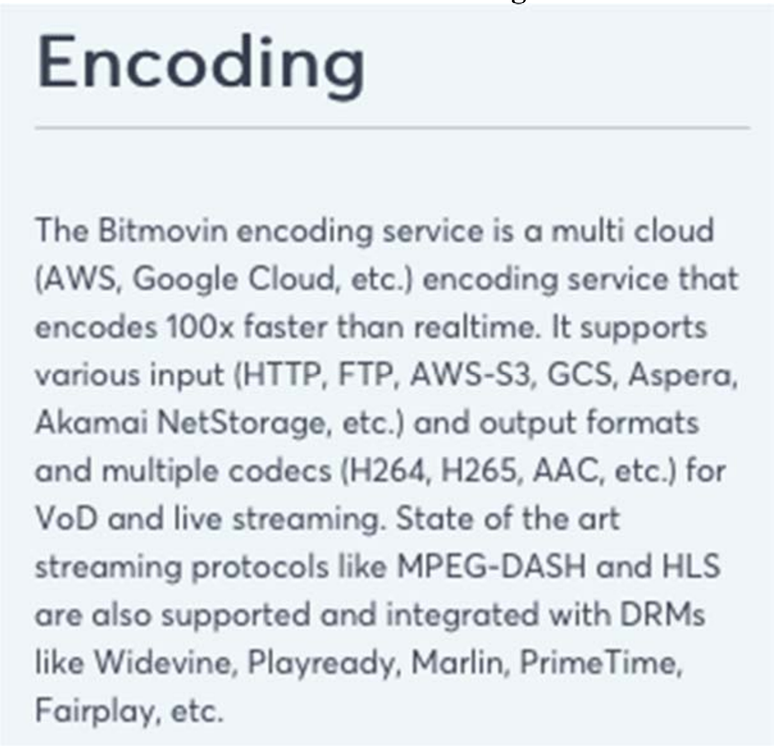
51. Upon information and belief, the ’118 Accused Infringing Devices infringe at least claim 1 in the exemplary manner described below.

52. The ’118 Accused Infringing Devices use H.264 (AVC) streams for coding video data (digital images) including macroblocks embedded in a binary stream.

53. H.264 is a widely used video compression format with decoder support on web browsers, TVs and other consumer devices. Moreover, H.264 codes digital images comprising macroblock streams.

54. The ’118 Accused Infringing Devices receive input video streams which are then encoded and/or transcoded using at least the H.264 standard. This is a widely used video compression format with decoder support on web browsers, TVs and other consumer devices. Moreover, H.264 uses motion compressor and estimator for motion coding video streams.

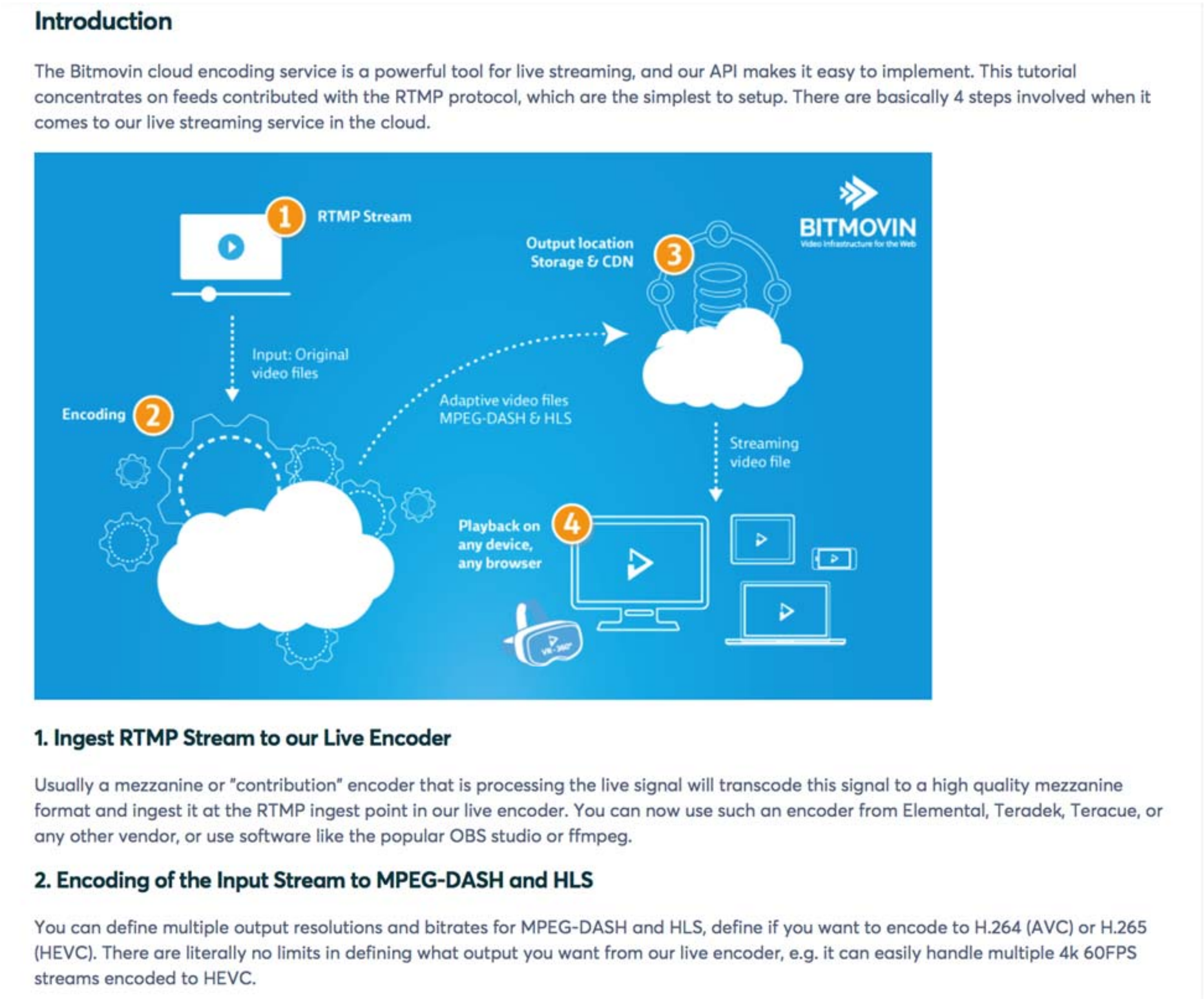
**Bitmovin encodes video streams using H.264 encoders**



Encoding

The Bitmovin encoding service is a multi cloud (AWS, Google Cloud, etc.) encoding service that encodes 100x faster than realtime. It supports various input (HTTP, FTP, AWS-S3, GCS, Aspera, Akamai NetStorage, etc.) and output formats and multiple codecs (H264, H265, AAC, etc.) for VoD and live streaming. State of the art streaming protocols like MPEG-DASH and HLS are also supported and integrated with DRMs like Widevine, Playready, Marlin, PrimeTime, Fairplay, etc.

**Source:** https://bitmovin.com/docs/encoding/api-reference#/reference/encoding, last accessed Nov. 29, 2018.



## Introduction

The Bitmovin cloud encoding service is a powerful tool for live streaming, and our API makes it easy to implement. This tutorial concentrates on feeds contributed with the RTMP protocol, which are the simplest to setup. There are basically 4 steps involved when it comes to our live streaming service in the cloud.

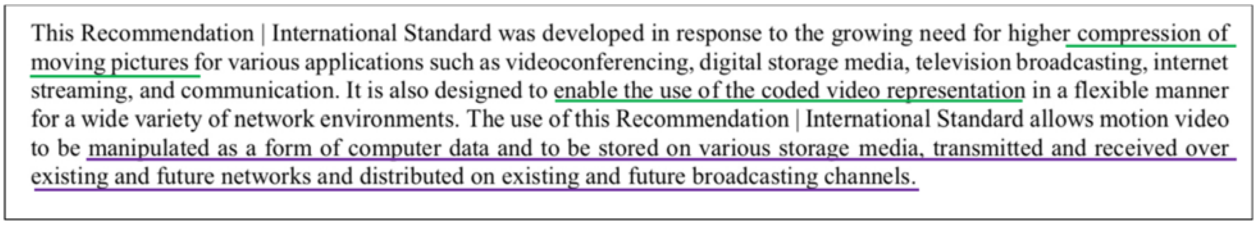
### 1. Ingest RTMP Stream to our Live Encoder

Usually a mezzanine or "contribution" encoder that is processing the live signal will transcode this signal to a high quality mezzanine format and ingest it at the RTMP ingest point in our live encoder. You can now use such an encoder from Elemental, Teradek, Teracue, or any other vendor, or use software like the popular OBS studio or ffmpeg.

### 2. Encoding of the Input Stream to MPEG-DASH and HLS

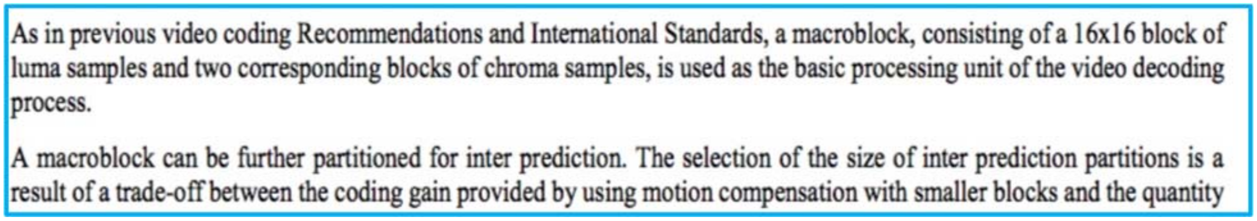
You can define multiple output resolutions and bitrates for MPEG-DASH and HLS, define if you want to encode to H.264 (AVC) or H.265 (HEVC). There are literally no limits in defining what output you want from our live encoder, e.g. it can easily handle multiple 4k 60FPS streams encoded to HEVC.

**Source:** https://bitmovin.com/docs/encoding/quickstarts/create-a-live-encoding-from-an-rtmp-stream



This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming, and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

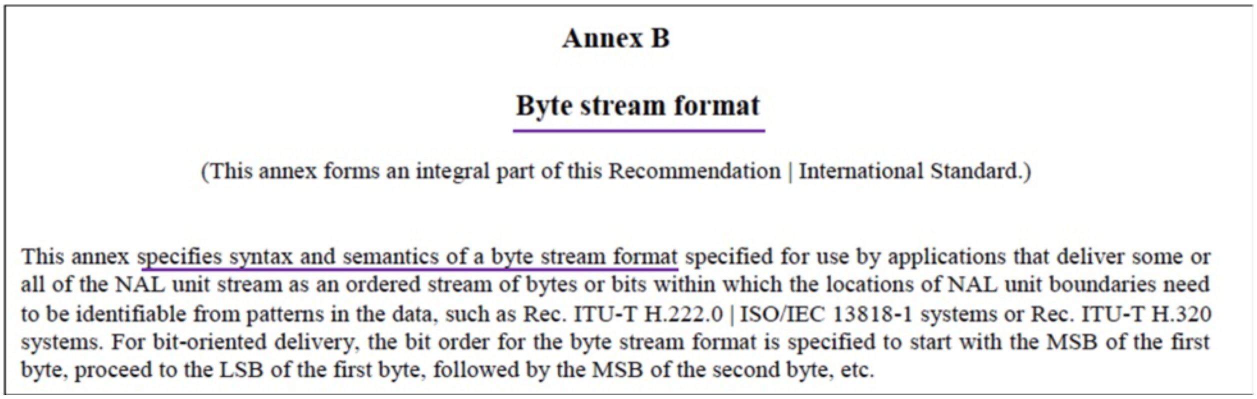
**Source**: https://www.itu.int/rec/T-REC-H.264-201704-I/en , p. i



As in previous video coding Recommendations and International Standards, a macroblock, consisting of a 16x16 block of luma samples and two corresponding blocks of chroma samples, is used as the basic processing unit of the video decoding process.

A macroblock can be further partitioned for inter prediction. The selection of the size of inter prediction partitions is a result of a trade-off between the coding gain provided by using motion compensation with smaller blocks and the quantity

**Source:** https://www.itu.int/rec/T-REC-H.264-201704-I/en, section 0.6.3



Annex B

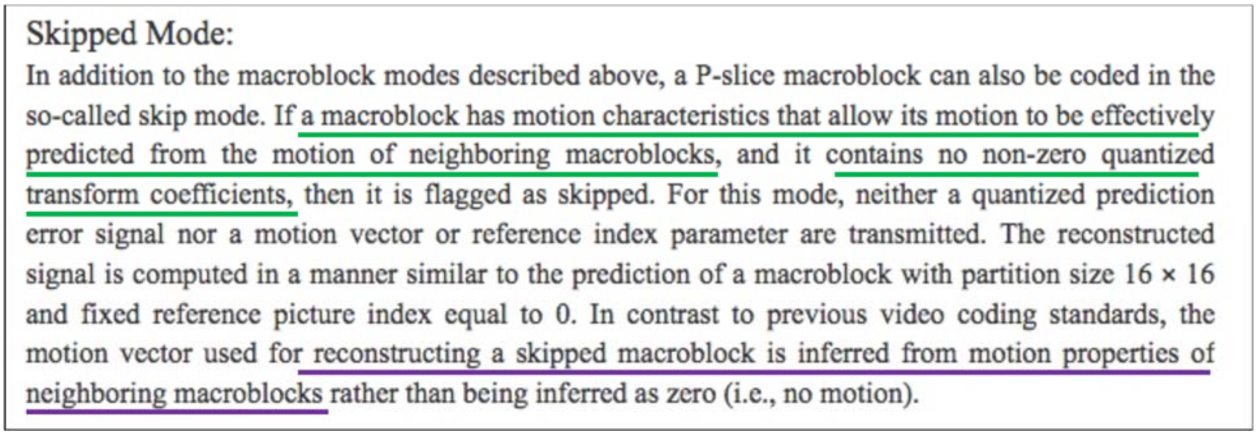
Byte stream format

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies syntax and semantics of a byte stream format specified for use by applications that deliver some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as Rec. ITU-T H.222.0 | ISO/IEC 13818-1 systems or Rec. ITU-T H.320 systems. For bit-oriented delivery, the bit order for the byte stream format is specified to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

**Source**: https://www.itu.int/rec/T-REC-H.264-201704-I/en, Annex B

55. H.264 coding in the ’118 Accused Infringing Devices supports skipped macroblocks. Before a macroblock is coded, an estimation is made of whether that macroblock can be reconstructed with an error concealment method by examining its motion characteristics, and checking to see that the resulting prediction contains no non-zero (i.e. all zero) quantized transform coefficients. This estimation provides an indication of the capacity for the macroblock to be reconstructed from properties of neighboring macroblocks, allowing the missing block to be concealed by inferring its properties.

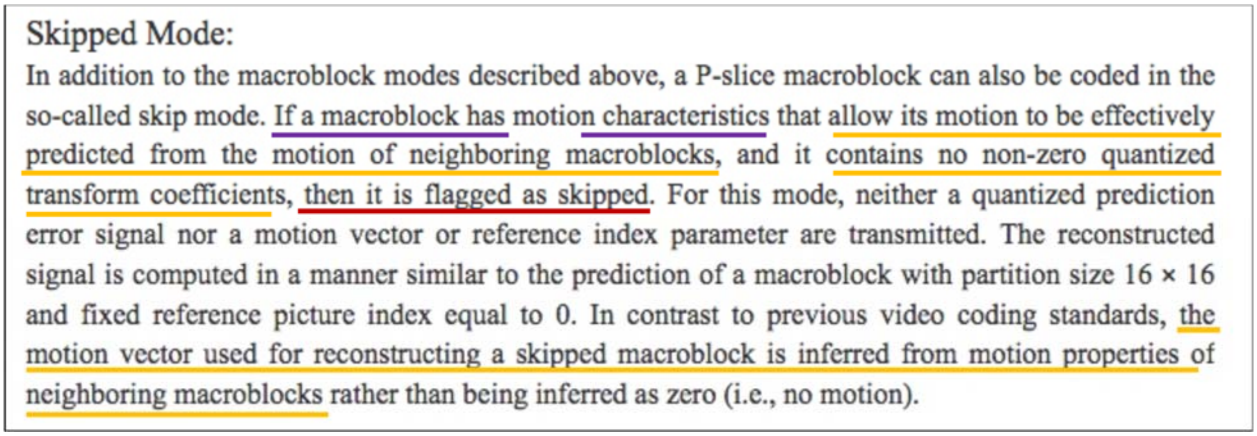


Skipped Mode:

In addition to the macroblock modes described above, a P-slice macroblock can also be coded in the so-called skip mode. If a macroblock has motion characteristics that allow its motion to be effectively predicted from the motion of neighboring macroblocks, and it contains no non-zero quantized transform coefficients, then it is flagged as skipped. For this mode, neither a quantized prediction error signal nor a motion vector or reference index parameter are transmitted. The reconstructed signal is computed in a manner similar to the prediction of a macroblock with partition size 16 × 16 and fixed reference picture index equal to 0. In contrast to previous video coding standards, the motion vector used for reconstructing a skipped macroblock is inferred from motion properties of neighboring macroblocks rather than being inferred as zero (i.e., no motion).

**Source:** http://mrutyunjayahiremath.blogspot.com/2010/09/h264-inter-predn.html

56. H.264 encoders in the ’118 Accused Infringing Devices perform a decision step to determine if a macroblock should be excluded from coding (skipped), with the decision to exclude made on the basis of its capacity to be reconstructing by inferring its motion properties from neighboring macroblocks, and based on all zero quantized transform coefficients.

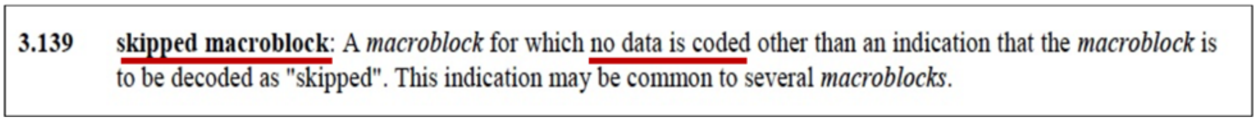


Skipped Mode:

In addition to the macroblock modes described above, a P-slice macroblock can also be coded in the so-called skip mode. If a macroblock has motion characteristics that allow its motion to be effectively predicted from the motion of neighboring macroblocks, and it contains no non-zero quantized transform coefficients, then it is flagged as skipped. For this mode, neither a quantized prediction error signal nor a motion vector or reference index parameter are transmitted. The reconstructed signal is computed in a manner similar to the prediction of a macroblock with partition size 16 × 16 and fixed reference picture index equal to 0. In contrast to previous video coding standards, the motion vector used for reconstructing a skipped macroblock is inferred from motion properties of neighboring macroblocks rather than being inferred as zero (i.e., no motion).

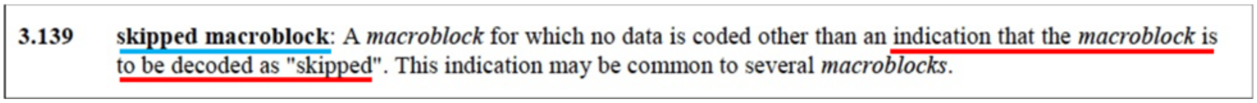
**Source:** http://mrutyunjayahiremath.blogspot.com/2010/09/h264-inter-predn.html

57. Skipped macroblocks are communicated with a mb_skip_flag = 1 (resynchronization marker at the point where the macroblocks are not coded (skipped)) in the binary data stream.



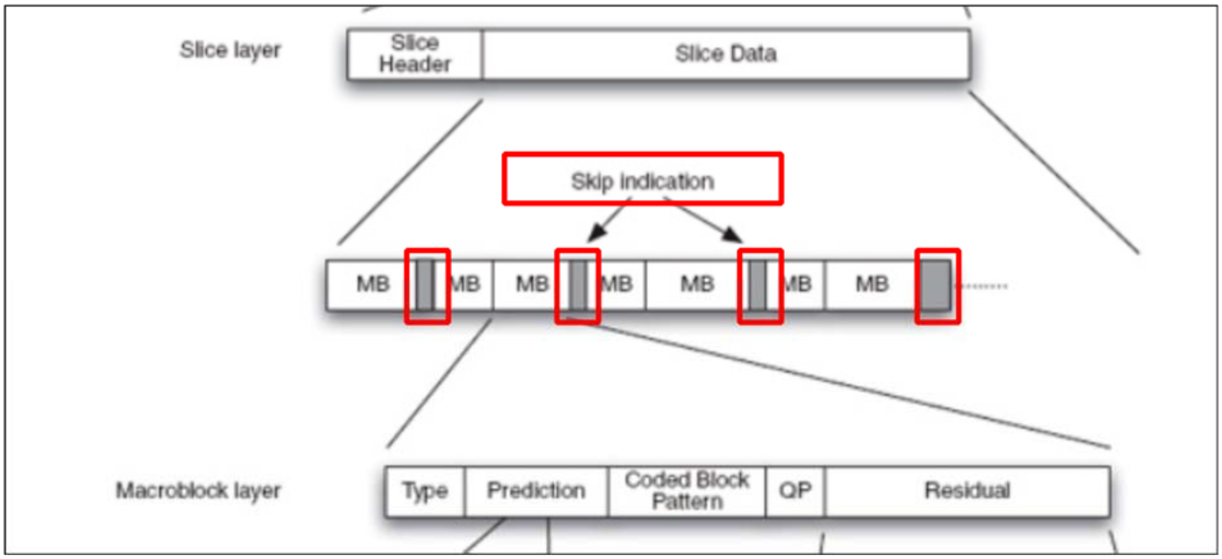
**3.139** **skipped macroblock**: A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.

**Source**: https://www.itu.int/rec/T-REC-H.264-201704-I/en, p13



**3.139** **skipped macroblock**: A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.

**Source:** https://www.itu.int/rec/T-REC-H.264-201704-I/en, p13



**Source:** https://www.safaribooksonline.com/library/view/the-h264 advanced/9780470516928/ch05.html#macroblock_layer

58. Bitmovin has thus infringed at least claim 1 of the ’118 patent by making, using, testing, selling, offering for sale, importing and/or licensing the ’118 Accused Infringing Devices, and operating them such that all steps of at least claim 1 are performed.

59. Bitmovin’s acts of direct infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin’s wrongful acts in an amount subject to proof at trial.

## COUNT III: INFRINGEMENT OF THE ’005 PATENT

60. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

61. The ’005 patent, titled “Method of Concurrent Multiple-Mode Motion Estimation For Digital Video,” issued on February 11, 2003. A copy of the ’005 patent is attached as Exhibit C. The priority date for ’005 patent is April 30, 1999. The inventions of the ’005 patent were developed by inventors at Koninklijke Philips Electronics N.V.

62. Pursuant to 35 U.S.C. § 282, the ’005 patent is presumed valid.

63. Claim 1 of the ’005 patent addresses a technological problem indigenous to motion coding in uncompressed digital video streams.

64. Claim 1 of the ’005 patent reads as follows:

> 1. A method for motion coding an uncompressed digital video data stream, including the steps of:
>
> comparing pixels of a first pixel array in a picture currently being coded with pixels of a plurality of second pixel arrays in at least one reference picture and concurrently performing motion estimation for each of a plurality of different prediction modes in order to determine which of the prediction modes is an optimum prediction mode;
>
> determining which of the second pixel arrays constitutes a best match with respect to the first pixel array for the optimum prediction mode; and,
>
> generating a motion vector for the first pixel array in response to the determining step.

65. The invention of claim 1 of the ’005 patent concerns “digital video compression” and, more particularly, “a motion estimation method and search engine for a digital video encoder that is simpler, faster, and less expensive than the presently available technology permits, and that permits concurrent motion estimation using multiple prediction modes.” ’005 patent at 1:6-11.

66. Data compression is the encoding of data using fewer “bits” than the original representation. Data compression is useful because it reduces the resources required to store and transmit data, and allows for faster retrieval and transmission of video data.

67. In the context of digital video with which the ’005 patent is concerned, a video codec is electronic circuitry or software that compresses and/or decompresses digital video for storage and/or transmission. Video codecs refer to video encoders and decoders.

68. Prior to digital video, video was typically stored as an analog signal on magnetic tape. Then, around the time of the development of compact discs (CDs), it became more feasible

to store and convey video in digital form. However, a large amount of storage and communications bandwidth was needed to record and convey raw video. Thus, what was needed was a method to reduce the amount of data used to represent the raw video. Accordingly, numerous engineers and many companies worked to develop solutions for compressing digital video data.

69. "Practical digital video compression started with the ITU H.261 standard in 1990." *A Brief History of Video Coding*, ARC International, Marco Jacobs and Jonah Probell (2007). Numerous other video compression standards thereafter were created and evolved. The innovation in digital video compression continues to this day.

70. In April 1999, at the time of the invention of claim 1 of the '005 patent, "different compression algorithms ha[d] been developed for digitally encoding video and audio information (hereinafter referred to generically as the 'digital video data stream') in order to minimize the bandwidth required to transmit this digital video data stream for a given picture quality." '005 patent at 1:11-17.

71. At the time of the invention of claim 1 of the '005 patent, the "most widely accepted international standards [for compression of digital video for motion pictures and television were] proposed by the Moving Pictures Expert Group (MPEG)." '005 patent at 1:20-24. Two such standards that existed at the time of the invention were MPEG-1 and MPEG-2.

72. In accordance with MPEG-1 and MPEG-2—and other compression standards for digital video—the video stream is "encoded/compressed . . . using a compression technique generally known as 'motion coding.'" '005 patent at 1:40-44. More particularly, rather than transmitting each video frame in its entirety, the standards at the time used motion estimation for

only those parts of sequential pictures that varied due to motion, where possible. ’005 patent at 1:45-48.

73. In general, the picture elements or “pixels” within a block of a picture are specified relative to those of a previously transmitted reference or “anchor” picture using differential or “residual” video, as well as so-called “motion vectors” that specify the location of an array (e.g., 16-by-16) of pixels or “macroblock” within the current picture relative to its original location within the anchor picture. ’005 patent at 1:48-55. A macroblock is a unit in image and video compression that typically consists of 16x16 samples of pixels. A motion vector is used to represent a macroblock in a picture based on the position of that same or similar macroblock in another picture (known as the reference picture).

74. At the time of the invention, there were various “prediction modes” that could be used for each macroblock that was to be encoded. ’005 patent at 3:7-11. Prediction modes are techniques for predicting image pixels or groups of pixels, and examples of prediction modes in MPEG include frame and field prediction modes. ’005 patent at 4:64-67. Moreover, at that time, motion coding allowed for the use of different prediction modes within the same frame, but required one prediction mode to be specified for a macroblock in advance of performing the motion estimation that results in a motion vector. ’005 patent at 3:12-15. Given that there are multiple prediction modes, the optimum prediction mode could not be known prior to encoding unless multiple motion estimations were performed on each macroblock sequentially. ’005 patent at 3:15-20. Then, after determining the optimum prediction mode based on multiple and sequential motion estimations, the optimal prediction mode would be selected and only then would the motion estimation that results in the generation of a motion vector occur.

75. In this prior art method, numerous and sequential motion estimations would have to run to find the optimal prediction mode. Only after these sequential motion estimations have been run and the optimal prediction mode selected could the motion estimation that results in the motion vector for the macroblock be carried out. Because "motion estimation usually consists of an exhaustive search procedure in which all 256 pixels of the two corresponding macroblocks are compared, and which is repeated for a large number of macroblocks," having to sequentially run numerous motion estimations to find the optimal prediction mode and only then performing the motion estimation using the optimal prediction mode to generate the motion vector is very computationally intensive, complex, inefficient, lengthy and cost ineffective. '005 patent at 3:20-43.

76. As demonstrated below, the claimed invention of claim 1 of the '005 patent provides a technological solution to the problem faced by the inventors, namely concurrently determining the optimal prediction mode while performing motion estimation along with generating the motion vector more simply, faster and in a less expensive way.

77. As detailed in the specification, the invention of claim 1 of the '005 patent provides a technological solution to the problems faced by the inventors.

> Based on the above and foregoing, it can be appreciated that there presently exists a need in the art that overcomes the disadvantages and shortcomings of the presently available technology. The present invention fulfills this need in the art by performing motion coding of an uncompressed digital video sequence in such a manner that the prediction mode for each individual macroblock is determined as part of the motion estimation process, along with the actual motion vector(s), and need not be specified in advance; only the type of picture currently being coded need be known. Since the latter must be determined at a higher level of video coding than the macroblock layer, this method makes possible a much more efficient, as well as optimal, degree of video compression than would otherwise be possible using conventional methods of motion estimation. Further, the present invention provides a novel scheme for concurrently searching for the optimum macroblock match within the appropriate anchor picture according to each of a plurality of motion prediction

> modes during the same search operation for the given macroblock, without the need for a separate search to be performed on the same macroblock for each such mode. Since this search procedure is the single most complex and expensive aspect of motion estimation, in both time and hardware, such a method as the present invention will clearly result in a more efficient video image coding and compression than would otherwise be possible given the aforementioned practical limitations of the presently available technology.

’005 patent at 3:40-67 (emphasis added).

78. The technological solution of claim 1 of the ’005 patent is further shown in Figure 3 which visually depicts a motion estimation process for concurrently performing motion estimation for frame prediction mode and field prediction modes for frame pictures:

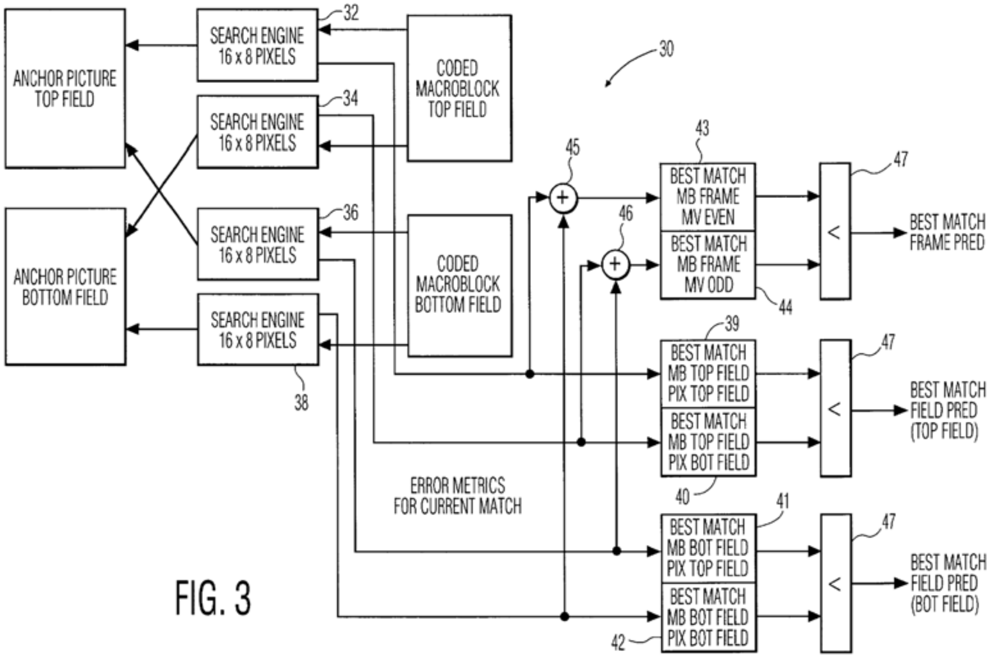

ANCHOR PICTURE TOP FIELD
ANCHOR PICTURE BOTTOM FIELD
SEARCH ENGINE 16 x 8 PIXELS
32
SEARCH ENGINE 16 x 8 PIXELS
34
SEARCH ENGINE 16 x 8 PIXELS
36
SEARCH ENGINE 16 x 8 PIXELS
38
CODED MACROBLOCK TOP FIELD
CODED MACROBLOCK BOTTOM FIELD
30
45
46
43
BEST MATCH MB FRAME MV EVEN
BEST MATCH MB FRAME MV ODD
44
47
<
BEST MATCH FRAME PRED
39
BEST MATCH MB TOP FIELD PIX TOP FIELD
BEST MATCH MB TOP FIELD PIX BOT FIELD
40
47
<
BEST MATCH FIELD PRED (TOP FIELD)
41
BEST MATCH MB BOT FIELD PIX TOP FIELD
BEST MATCH MB BOT FIELD PIX BOT FIELD
42
47
<
BEST MATCH FIELD PRED (BOT FIELD)
ERROR METRICS FOR CURRENT MATCH

FIG. 3

79. Claim 1 of the ’005 patent improves the functionality of motion coding in video compression by performing the concurrent determination of the optimal prediction mode while performing motion estimation along with generating the motion vector. The claimed invention of claim 1 of the ’005 patent also was not well-understood, routine or conventional at the time of

the invention. Rather, as set forth below, the claimed invention was a departure from the conventional ways of performing motion coding in video compression.

> Based on the above and foregoing, it can be appreciated that there presently exists a need in the art that overcomes the disadvantages and shortcomings of the presently available technology. The present invention fulfills this need in the art by performing motion coding of an uncompressed digital video sequence in such a manner that the prediction mode for each individual macroblock is determined as part of the motion estimation process, along with the actual motion vector(s), and need not be specified in advance; only the type of picture currently being coded need be known. Since the latter must be determined at a higher level of video coding than the macroblock layer, this method makes possible a much more efficient, as well as optimal, degree of video compression than would otherwise be possible using conventional methods of motion estimation. Further, the present invention provides a novel scheme for concurrently searching for the optimum macroblock match within the appropriate anchor picture according to each of a plurality of motion prediction modes during the same search operation for the given macroblock, without the need for a separate search to be performed on the same macroblock for each such mode. Since this search procedure is the single most complex and expensive aspect of motion estimation, in both time and hardware, such a method as the present invention will clearly result in a more efficient video image coding and compression than would otherwise be possible given the aforementioned practical limitations of the presently available technology.

’005 patent at 3:40-67 (emphasis added).

> The present invention relates generally to digital video compression, and, more particularly, to a motion estimation method and search engine for a digital video encoder that is simpler, faster, and less expensive than the presently available technology permits, and that permits concurrent motion estimation using multiple prediction modes.

’005 patent at 1:7-11 (emphasis added).

> In either case, the methods and architectures of the present invention result in a means of significantly improving the video compression efficiency and, hence, the resulting picture quality, without the need for either greater hardware costs or higher computational complexity.

’005 patent at 14:62-67 (emphasis added).

> In all known motion estimation methods, the prediction mode must be specified for every macroblock before the motion estimation, with its constituent search, is performed. However, in accordance with the present invention, in one of its

> aspects, the motion estimation may be performed, in a frame picture, forth both frame and field prediction modes simultaneously, during the same search for the anchor picture.

’005 patent at 8:6-13 (emphasis added).

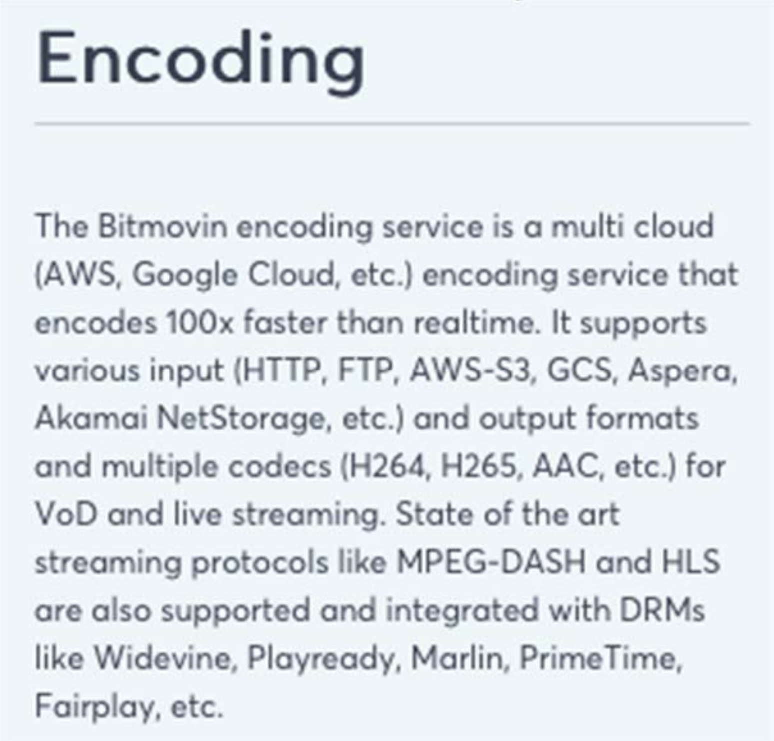
80. In light of the foregoing, and the general knowledge of a person of ordinary skill in the art, a person of ordinary skill in the art reading the ’005 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in the field of digital video compression. Moreover, a person of ordinary skill in the art would understand that the claimed subject matter of the ’005 patent presents advancements in the field of digital video compression, and more particularly to a motion estimation method and search engine for a digital video encoder that is simpler, faster, and less expensive than prior art technology, and that permits concurrent motion estimation using multiple prediction modes. A person of ordinary skill in the art would understand that claim 1 of the ’005 patent is directed to a method for motion coding an uncompressed digital video data stream, which provides concurrent motion estimation using multiple prediction modes along with the generation of motion vectors. Moreover, a person of ordinary skill in the art would understand that claim 1 of the ’005 patent contains that corresponding inventive concept.

81. Upon information and belief, Bitmovin makes, uses, offers for sale, and/or sells in the United States and/or imports into the United States products and services such as its H.264 encoders that practice a method for motion coding an uncompressed digital video data stream (collectively the “’005 Accused Infringing Devices”).

82. Upon information and belief, the ’005 Accused Infringing Devices infringe at least claim 1 in the exemplary manner described below.

83. The ’005 Accused Infringing Devices provide a method for motion coding an uncompressed (pixel level) digital video data stream. The ’005 Accused Infringing Devices receive input video streams which are then encoded and/or transcoded using at least the H.264 (AVC) standard. The H.264 standard is a widely used video compression format with decoder support on web browsers, TVs and other consumer devices. Moreover, H.264 uses motion compressor and estimator for motion coding video streams.

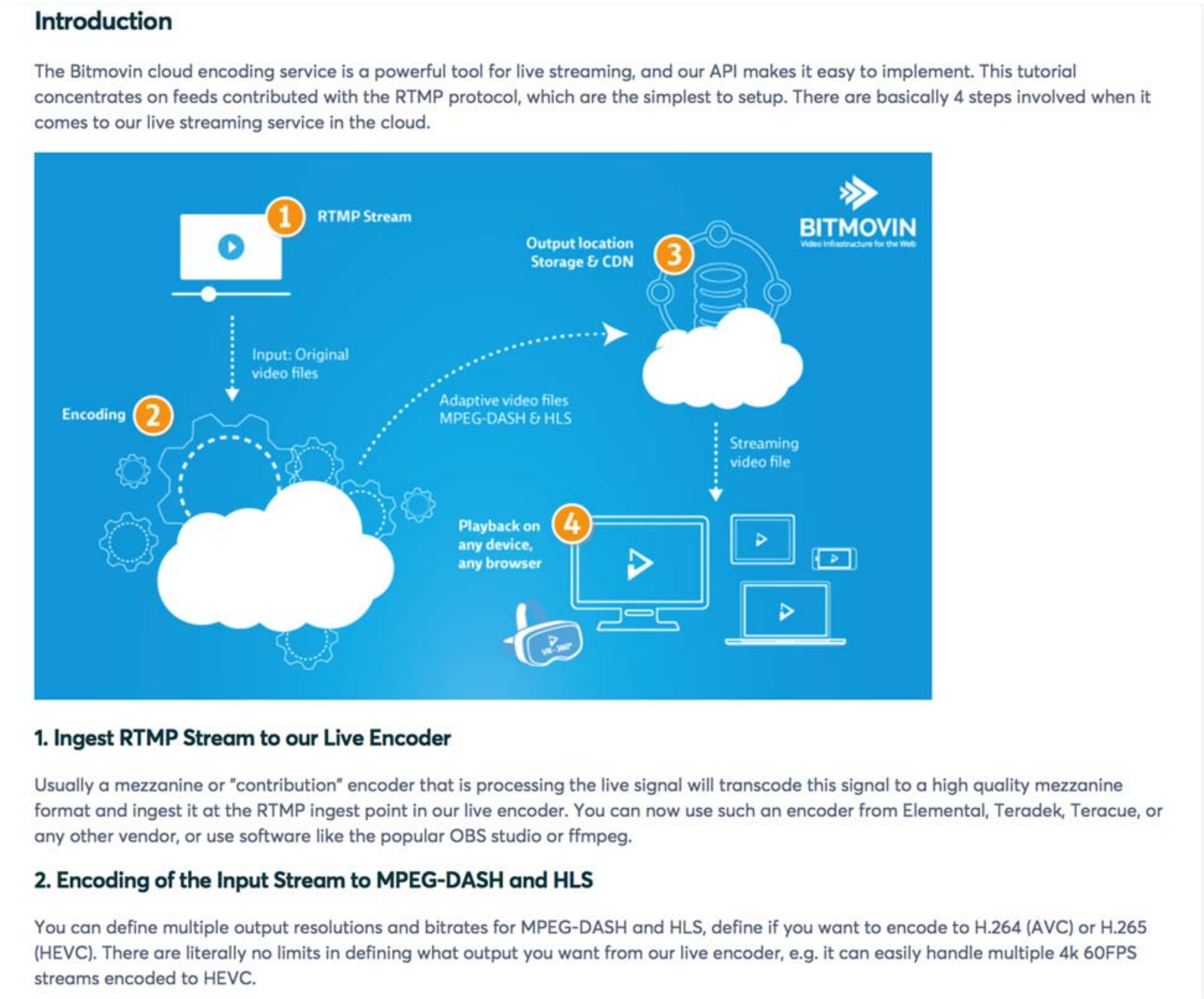
**Bitmovin encodes video streams using H.264 encoders**



# Encoding

The Bitmovin encoding service is a multi cloud (AWS, Google Cloud, etc.) encoding service that encodes 100x faster than realtime. It supports various input (HTTP, FTP, AWS-S3, GCS, Aspera, Akamai NetStorage, etc.) and output formats and multiple codecs (H264, H265, AAC, etc.) for VoD and live streaming. State of the art streaming protocols like MPEG-DASH and HLS are also supported and integrated with DRMs like Widevine, Playready, Marlin, PrimeTime, Fairplay, etc.

**Source:** https://bitmovin.com/docs/encoding/api-reference#/reference/encoding, last accessed Nov. 29, 2018.



## Introduction

The Bitmovin cloud encoding service is a powerful tool for live streaming, and our API makes it easy to implement. This tutorial concentrates on feeds contributed with the RTMP protocol, which are the simplest to setup. There are basically 4 steps involved when it comes to our live streaming service in the cloud.

1 RTMP Stream
Input: Original video files
Encoding 2
Adaptive video files MPEG-DASH & HLS
Output location Storage & CDN 3
BITMOVIN
Video Infrastructure for the Web
Streaming video file
Playback on any device, any browser 4
VR 360°

### 1. Ingest RTMP Stream to our Live Encoder

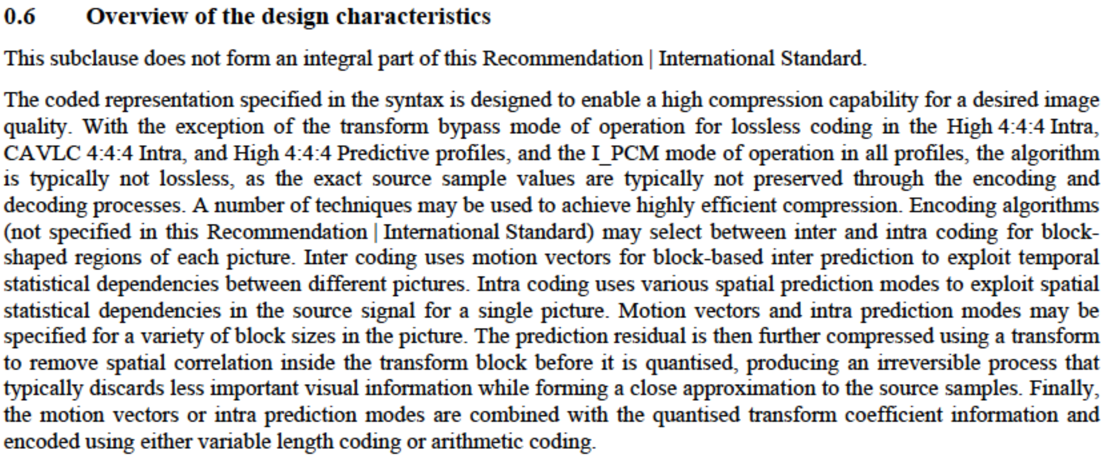
Usually a mezzanine or "contribution" encoder that is processing the live signal will transcode this signal to a high quality mezzanine format and ingest it at the RTMP ingest point in our live encoder. You can now use such an encoder from Elemental, Teradek, Teracue, or any other vendor, or use software like the popular OBS studio or ffmpeg.

### 2. Encoding of the Input Stream to MPEG-DASH and HLS

You can define multiple output resolutions and bitrates for MPEG-DASH and HLS, define if you want to encode to H.264 (AVC) or H.265 (HEVC). There are literally no limits in defining what output you want from our live encoder, e.g. it can easily handle multiple 4k 60FPS streams encoded to HEVC.

**Source:** https://bitmovin.com/docs/encoding/quickstarts/create-a-live-encoding-from-an-rtmp-stream

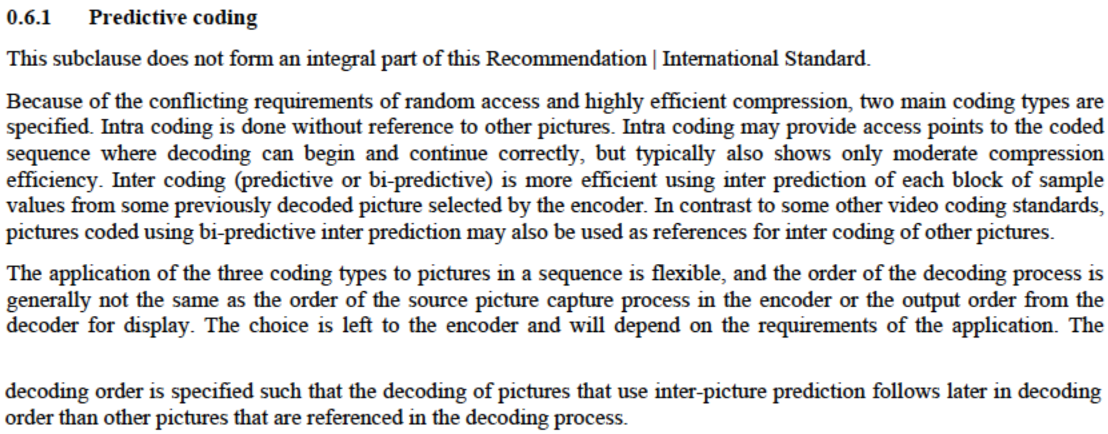
## H.264 Uses Predictive Coding



### 0.6 Overview of the design characteristics

This subclause does not form an integral part of this Recommendation | International Standard.

The coded representation specified in the syntax is designed to enable a high compression capability for a desired image quality. With the exception of the transform bypass mode of operation for lossless coding in the High 4:4:4 Intra, CAVLC 4:4:4 Intra, and High 4:4:4 Predictive profiles, and the I_PCM mode of operation in all profiles, the algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes. A number of techniques may be used to achieve highly efficient compression. Encoding algorithms (not specified in this Recommendation | International Standard) may select between inter and intra coding for block-shaped regions of each picture. Inter coding uses motion vectors for block-based inter prediction to exploit temporal statistical dependencies between different pictures. Intra coding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. Motion vectors and intra prediction modes may be specified for a variety of block sizes in the picture. The prediction residual is then further compressed using a transform to remove spatial correlation inside the transform block before it is quantised, producing an irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes are combined with the quantised transform coefficient information and encoded using either variable length coding or arithmetic coding.



**0.6.1 Predictive coding**

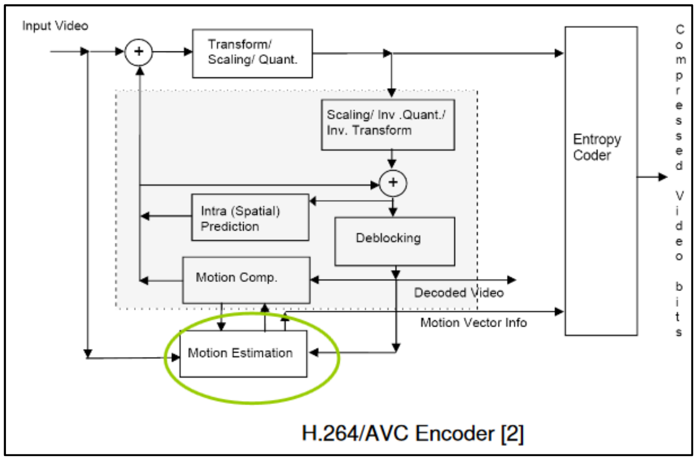
This subclause does not form an integral part of this Recommendation | International Standard.

Because of the conflicting requirements of random access and highly efficient compression, two main coding types are specified. Intra coding is done without reference to other pictures. Intra coding may provide access points to the coded sequence where decoding can begin and continue correctly, but typically also shows only moderate compression efficiency. Inter coding (predictive or bi-predictive) is more efficient using inter prediction of each block of sample values from some previously decoded picture selected by the encoder. In contrast to some other video coding standards, pictures coded using bi-predictive inter prediction may also be used as references for inter coding of other pictures.

The application of the three coding types to pictures in a sequence is flexible, and the order of the decoding process is generally not the same as the order of the source picture capture process in the encoder or the output order from the decoder for display. The choice is left to the encoder and will depend on the requirements of the application. The

decoding order is specified such that the decoding of pictures that use inter-picture prediction follows later in decoding order than other pictures that are referenced in the decoding process.

**Source:** H.264 Standard (03-2010) at pp. 3-4

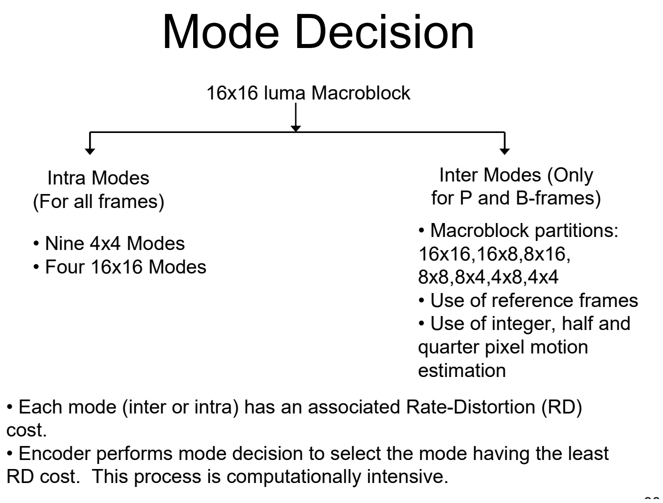


**Source:** https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf

84. The ’005 Accused Infringing Devices provide a method for comparing pixels of a first pixel array (e.g., a macroblock) in a picture currently being coded with pixels of a plurality of second pixel arrays in at least one reference picture and concurrently performing motion estimation for each of a plurality of different prediction modes in order to determine which of the prediction modes is an optimum prediction mode.

85. H.264 uses different motion estimation modes in inter-frame prediction. These modes are commonly referred to as inter-frame prediction modes, or inter modes. Each inter mode involves partitioning the current macroblock into a different combination of sub blocks,

and selecting the optimum motion vector for the current macroblock based on the partition. The inter-frame prediction modes, or inter modes, can be further categorized by the number and position of the reference frames, as well as the choice of integer pixel, half pixel and quarter pixel values in motion estimation. The Bitmovin H.264 encoders concurrently perform motion estimation of a macroblock for all inter-modes and select the most optimum prediction mode with least rate distortion cost.



Mode Decision

16x16 luma Macroblock

Intra Modes
(For all frames)

- Nine 4x4 Modes
- Four 16x16 Modes

Inter Modes (Only
for P and B-frames)

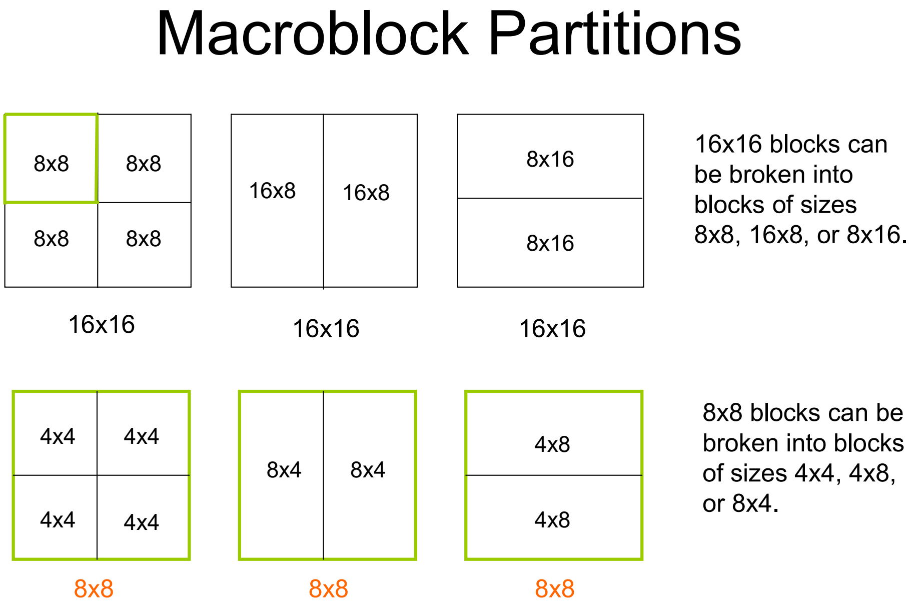
- Macroblock partitions: 16x16,16x8,8x16, 8x8,8x4,4x8,4x4
- Use of reference frames
- Use of integer, half and quarter pixel motion estimation

- Each mode (inter or intra) has an associated Rate-Distortion (RD) cost.
- Encoder performs mode decision to select the mode having the least RD cost. This process is computationally intensive.

**Source:** https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 30

86. H.264 provides a hierarchical way to partition a macroblock, with the available partitions shown in the following two figures. An exemplary inter-frame prediction mode, or inter mode, can be for a macroblock to be partitioned to encompass a 16x8 sub block on the left, and two 8x8 sub blocks on the right.

**Macroblock partitions for inter-frame prediction modes**



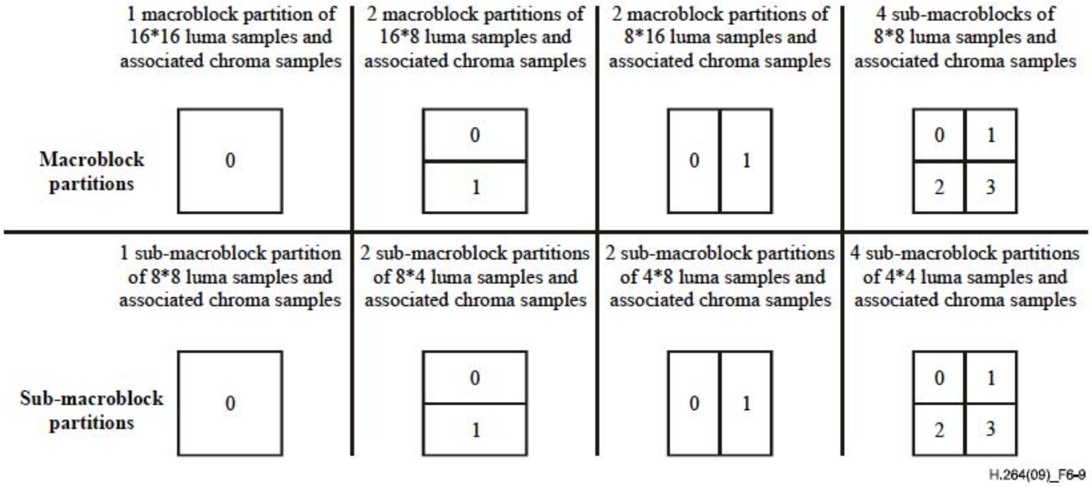
Macroblock Partitions

16x16 blocks can be broken into blocks of sizes 8x8, 16x8, or 8x16.

8x8 blocks can be broken into blocks of sizes 4x4, 4x8, or 8x4.

**Source:** https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 4

**H.264 provides macroblock partitions for inter-frame prediction modes**



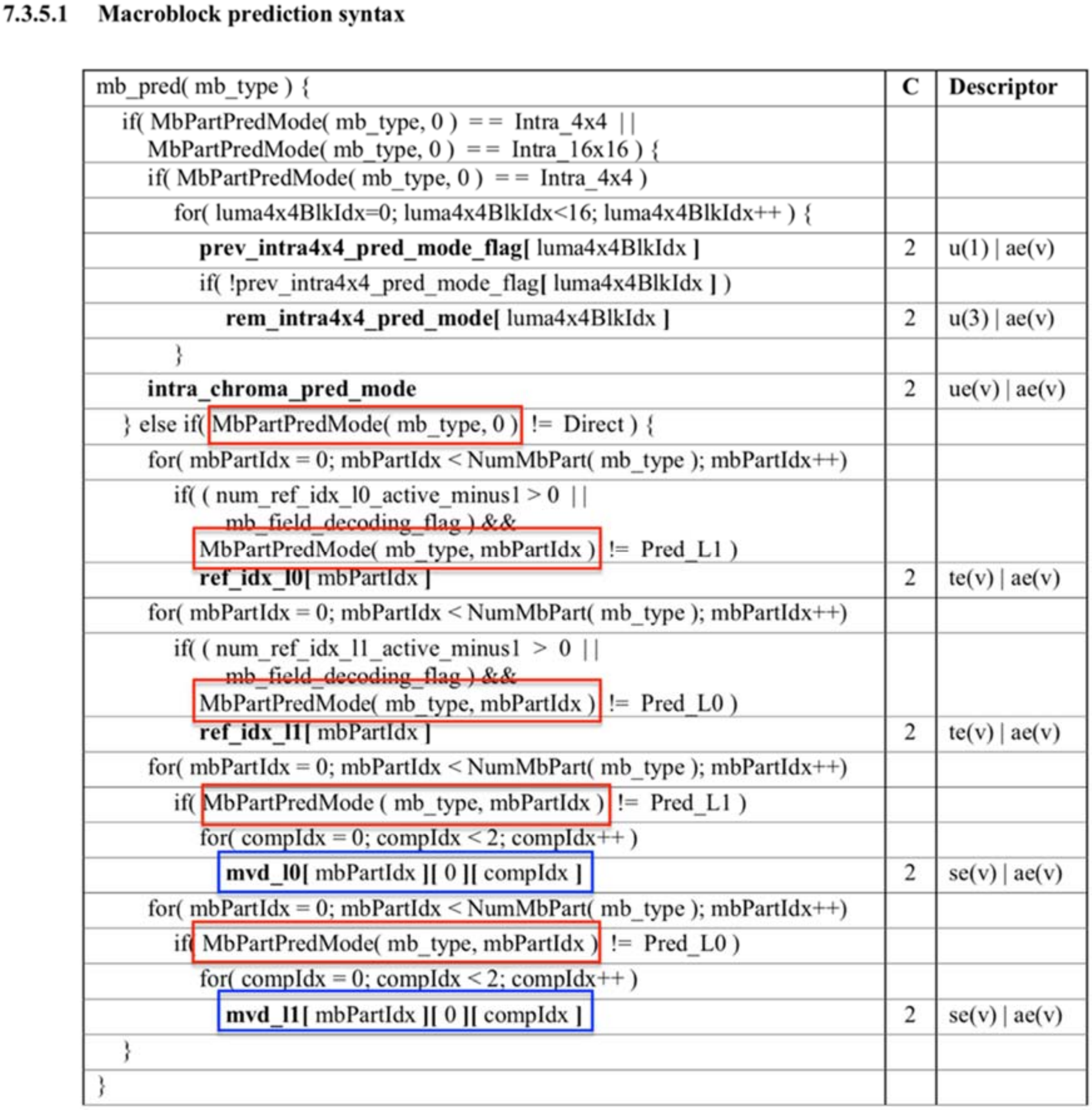
1 macroblock partition of 16*16 luma samples and associated chroma samples
2 macroblock partitions of 16*8 luma samples and associated chroma samples
2 macroblock partitions of 8*16 luma samples and associated chroma samples
4 sub-macroblocks of 8*8 luma samples and associated chroma samples
Macroblock partitions
0
0
1
0
1
0
1
2
3
1 sub-macroblock partition of 8*8 luma samples and associated chroma samples
2 sub-macroblock partitions of 8*4 luma samples and associated chroma samples
2 sub-macroblock partitions of 4*8 luma samples and associated chroma samples
4 sub-macroblock partitions of 4*4 luma samples and associated chroma samples
Sub-macroblock partitions
0
0
1
0
1
0
1
2
3
H.264(09)_F6-9

**Figure 6-9 – Macroblock partitions, sub-macroblock partitions, macroblock partition scans, and sub-macroblock partition scans**

**Source:** H.264 Standard (03-2010) at p. 26

87. The optimum prediction mode as chosen for the current macroblock is embedded in the compressed bit stream of H.264, as shown in the following two syntaxes.

**Macroblock prediction syntax in H.264**

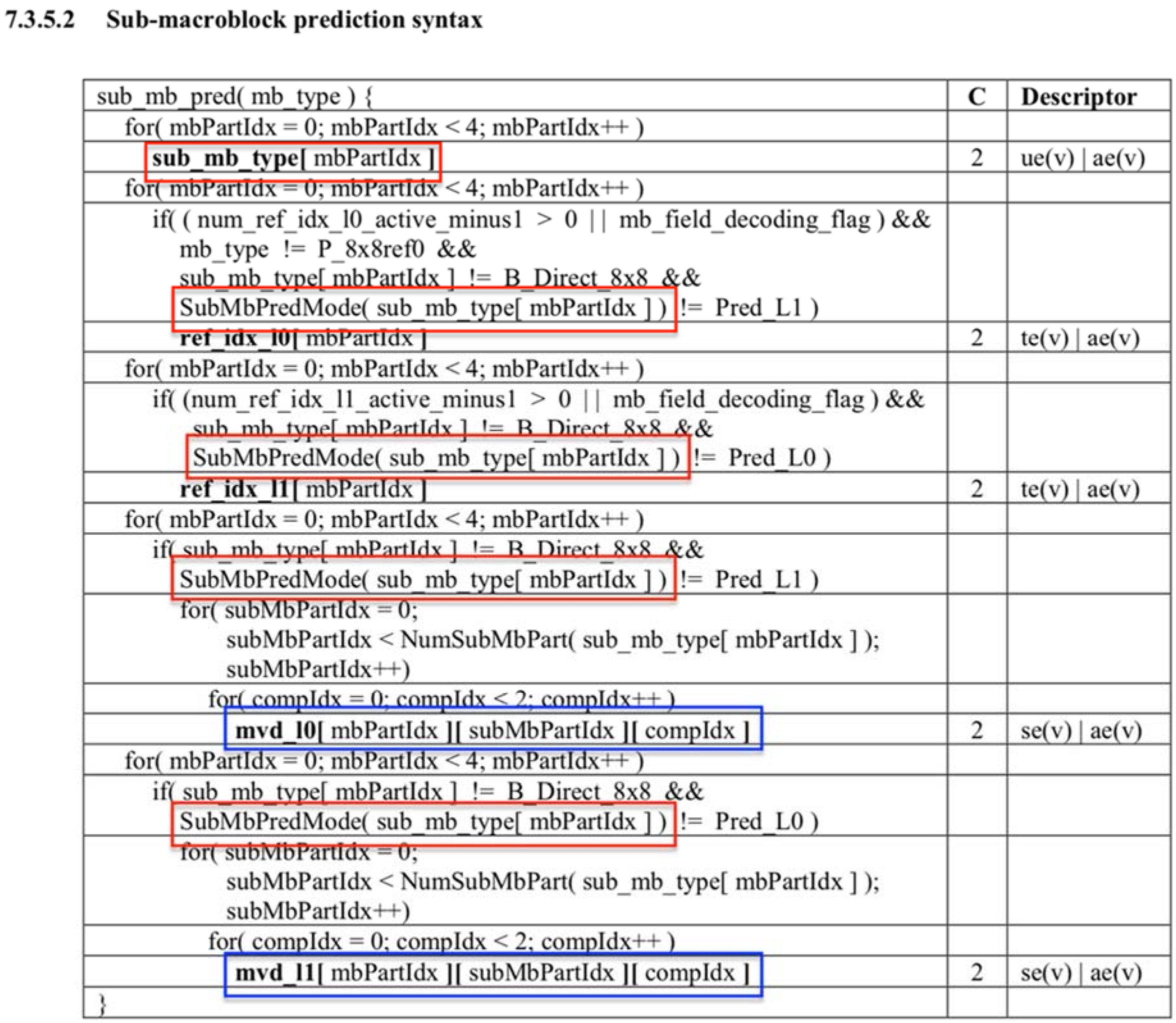


**7.3.5.1 Macroblock prediction syntax**

| mb_pred( mb_type ) { | C | Descriptor |
|---|---|---|
|   if( MbPartPredMode( mb_type, 0 ) = = Intra_4x4 \|\|<br>    MbPartPredMode( mb_type, 0 ) = = Intra_16x16 ) { | | |
|     if( MbPartPredMode( mb_type, 0 ) = = Intra_4x4 ) | | |
|       for( luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++ ) { | | |
|         **prev_intra4x4_pred_mode_flag**[ luma4x4BlkIdx ] | 2 | u(1) \| ae(v) |
|         if( !prev_intra4x4_pred_mode_flag[ luma4x4BlkIdx ] ) | | |
|           **rem_intra4x4_pred_mode**[ luma4x4BlkIdx ] | 2 | u(3) \| ae(v) |
|       } | | |
|     **intra_chroma_pred_mode** | 2 | ue(v) \| ae(v) |
|   } else if( MbPartPredMode( mb_type, 0 ) != Direct ) { | | |
|     for( mbPartIdx = 0; mbPartIdx < NumMbPart( mb_type ); mbPartIdx++) | | |
|       if( ( num_ref_idx_l0_active_minus1 > 0 \|\|<br>          mb_field_decoding_flag ) &&<br>        MbPartPredMode( mb_type, mbPartIdx ) != Pred_L1 ) | | |
|         **ref_idx_l0**[ mbPartIdx ] | 2 | te(v) \| ae(v) |
|     for( mbPartIdx = 0; mbPartIdx < NumMbPart( mb_type ); mbPartIdx++) | | |
|       if( ( num_ref_idx_l1_active_minus1 > 0 \|\|<br>          mb_field_decoding_flag ) &&<br>        MbPartPredMode( mb_type, mbPartIdx ) != Pred_L0 ) | | |
|         **ref_idx_l1**[ mbPartIdx ] | 2 | te(v) \| ae(v) |
|     for( mbPartIdx = 0; mbPartIdx < NumMbPart( mb_type ); mbPartIdx++) | | |
|       if( MbPartPredMode ( mb_type, mbPartIdx ) != Pred_L1 ) | | |
|         for( compIdx = 0; compIdx < 2; compIdx++ ) | | |
|           **mvd_l0**[ mbPartIdx ][ 0 ][ compIdx ] | 2 | se(v) \| ae(v) |
|     for( mbPartIdx = 0; mbPartIdx < NumMbPart( mb_type ); mbPartIdx++) | | |
|       if( MbPartPredMode( mb_type, mbPartIdx ) != Pred_L0 ) | | |
|         for( compIdx = 0; compIdx < 2; compIdx++ ) | | |
|           **mvd_l1**[ mbPartIdx ][ 0 ][ compIdx ] | 2 | se(v) \| ae(v) |
|   } | | |
| } | | |

**Source:** H.264 Standard (03-2010) at p. 57

## Sub-macroblock prediction syntax in H.264



### 7.3.5.2 Sub-macroblock prediction syntax

| sub_mb_pred( mb_type ) { | C | Descriptor |
|---|---|---|
| for( mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++ ) | | |
| **sub_mb_type**[ mbPartIdx ] | 2 | ue(v) \| ae(v) |
| for( mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++ ) | | |
| if( ( num_ref_idx_l0_active_minus1 > 0 \|\| mb_field_decoding_flag ) && mb_type != P_8x8ref0 && sub_mb_type[ mbPartIdx ] != B_Direct_8x8 && SubMbPredMode( sub_mb_type[ mbPartIdx ] ) != Pred_L1 ) | | |
| **ref_idx_l0**[ mbPartIdx ] | 2 | te(v) \| ae(v) |
| for( mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++ ) | | |
| if( (num_ref_idx_l1_active_minus1 > 0 \|\| mb_field_decoding_flag ) && sub_mb_type[ mbPartIdx ] != B_Direct_8x8 && SubMbPredMode( sub_mb_type[ mbPartIdx ] ) != Pred_L0 ) | | |
| **ref_idx_l1**[ mbPartIdx ] | 2 | te(v) \| ae(v) |
| for( mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++ ) | | |
| if( sub_mb_type[ mbPartIdx ] != B_Direct_8x8 && SubMbPredMode( sub_mb_type[ mbPartIdx ] ) != Pred_L1 ) | | |
| for( subMbPartIdx = 0; subMbPartIdx < NumSubMbPart( sub_mb_type[ mbPartIdx ] ); subMbPartIdx++) | | |
| for( compIdx = 0; compIdx < 2; compIdx++ ) | | |
| **mvd_l0**[ mbPartIdx ][ subMbPartIdx ][ compIdx ] | 2 | se(v) \| ae(v) |
| for( mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++ ) | | |
| if( sub_mb_type[ mbPartIdx ] != B_Direct_8x8 && SubMbPredMode( sub_mb_type[ mbPartIdx ] ) != Pred_L0 ) | | |
| for( subMbPartIdx = 0; subMbPartIdx < NumSubMbPart( sub_mb_type[ mbPartIdx ] ); subMbPartIdx++) | | |
| for( compIdx = 0; compIdx < 2; compIdx++ ) | | |
| **mvd_l1**[ mbPartIdx ][ subMbPartIdx ][ compIdx ] | 2 | se(v) \| ae(v) |
| } | | |

**Source:** H.264 Standard (03-2010) at p. 58

88. The ’005 Accused Infringing Devices provide a method for determining which of the second pixel arrays (e.g., macroblock) constitutes a best match with respect to the first pixel array (e.g., macroblock) for the optimum prediction mode.

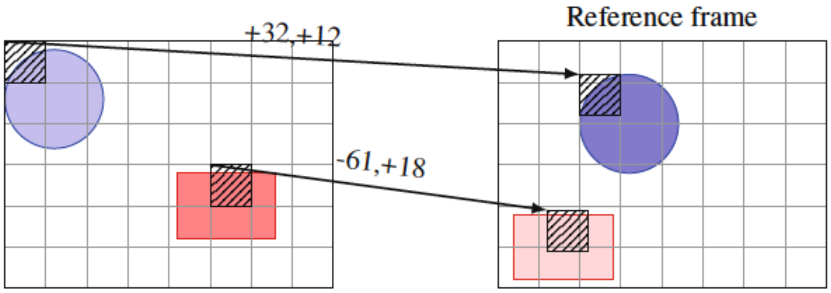
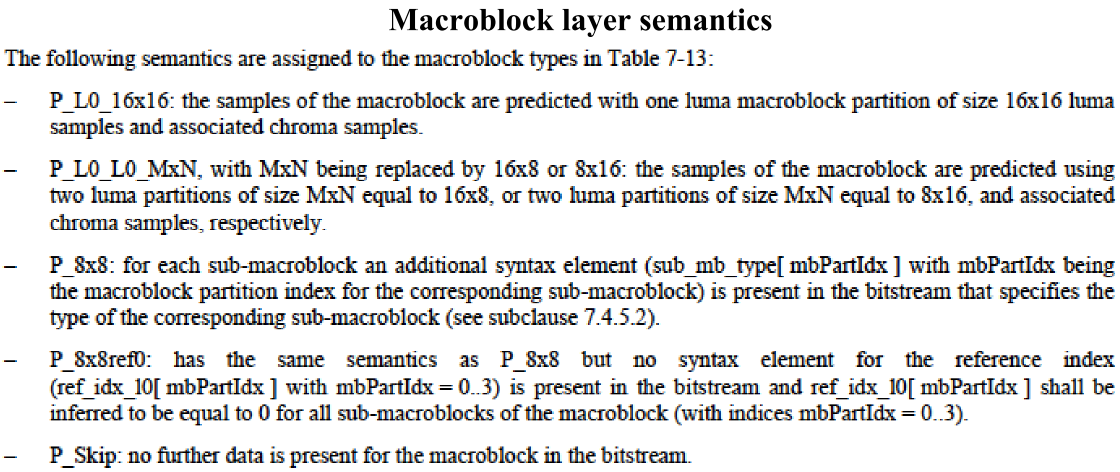


Fig. 2.4: Motion estimation. For each MB the best matching block in the reference frame is found. The encoder codes the differences (errors) between the MBs and their best matching blocks. Arrows indicate motion vectors and are labeled by the vector coordinates. In this example the shapes are identical but their colors are slightly larger/darker.

**Source:** B. Juurlink et al., Scalable Parallel Programming Applied to H.264, Chapter 2: Understanding the Application: An Overview of the H.264 Standard, p. 12

89. For example, the encoder performs mode decision to select the most optimum prediction mode with least rate distortion cost.



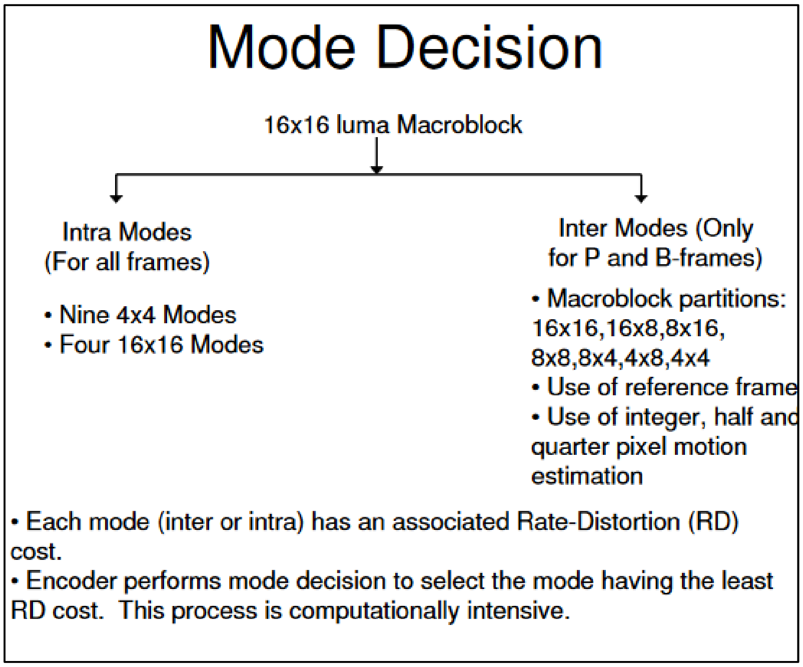
### Macroblock layer semantics

The following semantics are assigned to the macroblock types in Table 7-13:

- P_L0_16x16: the samples of the macroblock are predicted with one luma macroblock partition of size 16x16 luma samples and associated chroma samples.
- P_L0_L0_MxN, with MxN being replaced by 16x8 or 8x16: the samples of the macroblock are predicted using two luma partitions of size MxN equal to 16x8, or two luma partitions of size MxN equal to 8x16, and associated chroma samples, respectively.
- P_8x8: for each sub-macroblock an additional syntax element (sub_mb_type[ mbPartIdx ] with mbPartIdx being the macroblock partition index for the corresponding sub-macroblock) is present in the bitstream that specifies the type of the corresponding sub-macroblock (see subclause 7.4.5.2).
- P_8x8ref0: has the same semantics as P_8x8 but no syntax element for the reference index (ref_idx_l0[ mbPartIdx ] with mbPartIdx = 0..3) is present in the bitstream and ref_idx_l0[ mbPartIdx ] shall be inferred to be equal to 0 for all sub-macroblocks of the macroblock (with indices mbPartIdx = 0..3).
- P_Skip: no further data is present for the macroblock in the bitstream.

**Source:** H.264 Standard (03-2010), p. 100

**Mode Decision**



Mode Decision

16x16 luma Macroblock

Intra Modes
(For all frames)

- Nine 4x4 Modes
- Four 16x16 Modes

Inter Modes (Only
for P and B-frames)

- Macroblock partitions:
16x16,16x8,8x16,
8x8,8x4,4x8,4x4
- Use of reference frame
- Use of integer, half and
quarter pixel motion
estimation

- Each mode (inter or intra) has an associated Rate-Distortion (RD) cost.
- Encoder performs mode decision to select the mode having the least RD cost. This process is computationally intensive.

**Source:** https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 30

90. The '005 Accused Infringing Devices provide a method for generating a motion vector for the first pixel array in response to the determining step. The encoder calculates the appropriate motion vectors and other data elements represented in the video data stream.

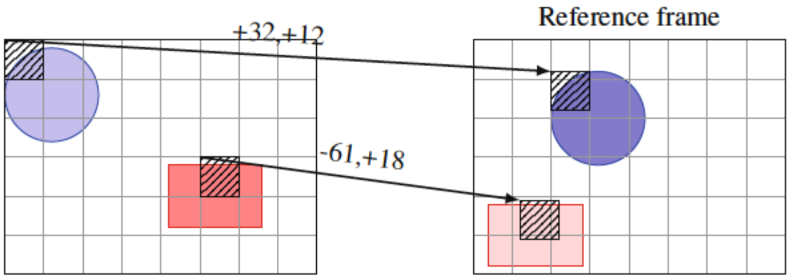
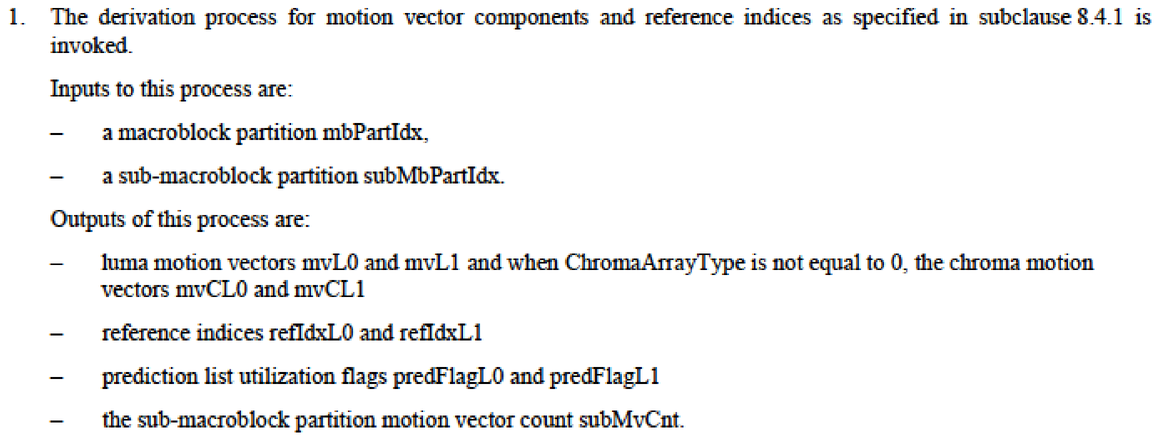


Fig. 2.4: Motion estimation. For each MB the best matching block in the reference frame is found. The encoder codes the differences (errors) between the MBs and their best matching blocks. Arrows indicate motion vectors and are labeled by the vector coordinates. In this example the shapes are identical but their colors are slightly larger/darker.

Source: B. Juurlink et al., Scalable Parallel Programming Applied to H.264, Chapter 2: Understanding the Application: An Overview of the H.264 Standard, p. 12

**Motion Vector Derivation is described below**



1. The derivation process for motion vector components and reference indices as specified in subclause 8.4.1 is invoked.

   Inputs to this process are:

   – a macroblock partition mbPartIdx,

   – a sub-macroblock partition subMbPartIdx.

   Outputs of this process are:

   – luma motion vectors mvL0 and mvL1 and when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1

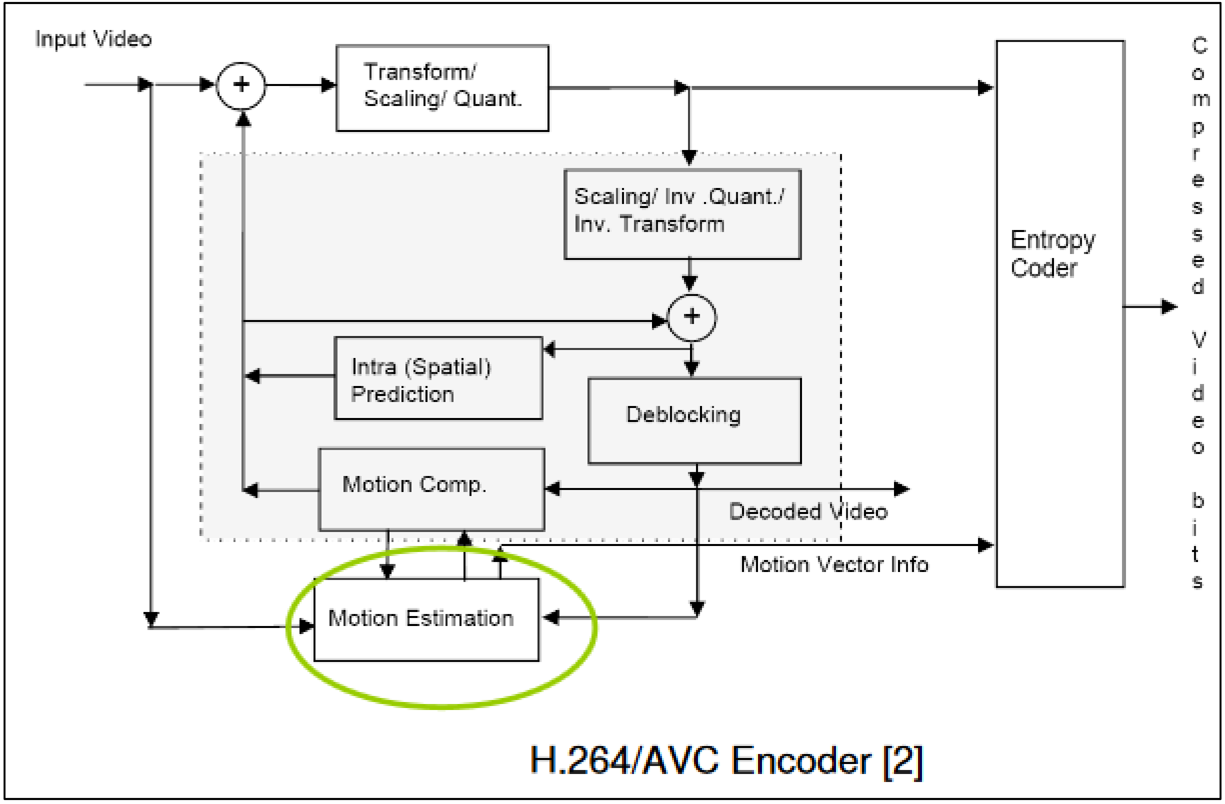
   – reference indices refIdxL0 and refIdxL1

   – prediction list utilization flags predFlagL0 and predFlagL1

   – the sub-macroblock partition motion vector count subMvCnt.

**Source:** H.264 Standard (03-2010), p. 151

**H.264 Encoder Block Diagram**



**Source:** https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 2

91. Bitmovin has thus infringed at least claim 1 of the ’005 patent by making, using, testing, selling, offering for sale, importing and/or licensing the ’005 Accused Infringing Devices, and operating them such that all steps of at least claim 1 are performed.

92. Bitmovin’s acts of direct infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin’s wrongful acts in an amount subject to proof at trial.

## COUNT IV: INFRINGEMENT OF THE ’345 PATENT

93. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

94. Uniloc owns by assignment the entire right, title, and interest in the ’345 patent.

95. The ’345 patent, is titled “Replacement of Substrings in File/Directory Pathnames With Numeric Tokens.” issued on October 22, 2002. A copy of the ’345 patent is attached as Exhibit D. The priority date for the ’345 patent is January 4, 2000. The inventions of the ’345 patent were developed by IBM.

96. Pursuant to 35 U.S.C. § 282, the ’345 patent is presumed valid.

97. Claim 1 of the ’345 patent addresses a technological problem indigenous to data processing systems and file systems in a networked environment—specifically in the computer science field of canonicalization. https://en.wikipedia.org/wiki/Canonicalization.

98. Claim 1 of the ’345 patent reads as follows:

> 1. A method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, comprising the acts of:
>
> reading a name string to be converted into a list of tokens;
>
> canonicalizing a current working directory and the name string to form a pathname containing a plurality of substrings;

> parsing the pathname and replacing each substring with an associated token; and
>
> validating the parsed pathname containing the list of tokens.

99. The invention of claim 1 of the ’345 patent concerns a novel method for canonicalization where substrings are replaced in file and directory pathnames with tokens in the computer-implemented file system.

100. At the time of invention of the ’345 patent, in the field of data processing systems and file systems in a networked environment, canonicalization was a task used in file systems to identify file system resources, such as files, directories or other types of resources. ’345 patent at 1:11-19. Another important task at the time is the semantic validation of a path, made up of the root, intermediate directories, and file or directory specification. *Id.* at 1:20-22. All intermediate directories must be valid for a pathname to refer to a valid file system resource. *Id.* at 1:22-27. Canonicalization and validation are often intertwined in a single function or set of functions. *Id.* at 1:28-29. The combination of these two functions can effect some savings by being more efficient. *Id.* at 1:34-35. If the current working directory for a given process is taken to be always valid, then validation of a path can start with the partial information specified by the user of the file system. *Id.* at 1:35-40. However useful this method of combining these two functions can be, the two tasks must always be considered separately, or severe penalties could occur. *Id.* at 1:41-44.

101. As demonstrated below, the claimed invention of claim 1 of the ’345 patent provides a technological solution to the problem faced by the inventors—replacing substrings in file and directory pathnames with tokens in a computer-implemented file system by parsing pathnames and replacing each substring with an associated token and validating the parsed pathname containing a list of tokens. This technological solution resulted in a significant and

substantial improvement in the performance of storage of strings as well as in the performance of comparing substrings and savings in the amount of storage needed to implement a file system as only one copy need be kept of any substring. ’345 patent at 2:24-41.

102. As detailed in the specification, in designing a file system that is structured on a client/server split, where the client portion keeps track of a current working directory and therefore has to perform the canonicalization, the path validation can often only be efficiently done by the server. *Id.* at 1:52-56. The inventors discovered that in most cases even where there is no client/server split, it is advantageous to separate canonicalization from validation and perform these two operations in a close sequence, but not interleaving validation of intermediate path information with a forming of a canonical name. *Id.* at 1:56-62. This results in a simpler implementation and superior performance, especially in a network environment. *Id.* at 1:62-63.

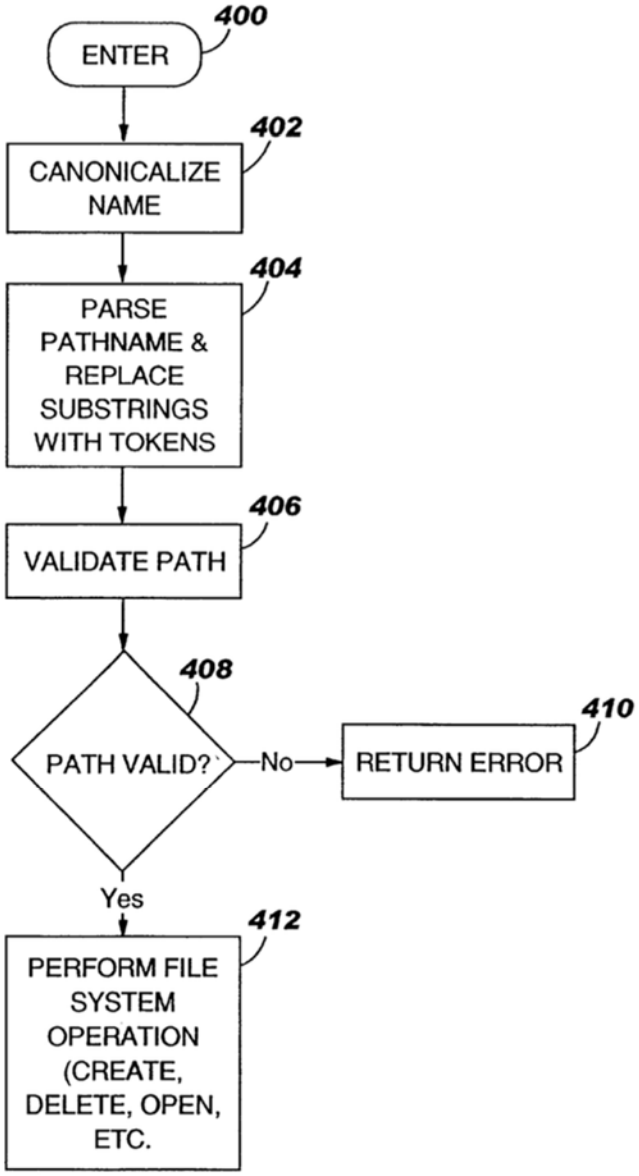
103. In dealing with file/directory pathnames, the number of sometimes quite lengthy strings poses a significant problem, especially when these are broken into substrings which then are constantly compared to other substrings. *Id.* at 2:24-27. According to the invention of the ’345 patent, parsing the strings into their semantically correct substrings and replacing those substrings with unique numeric tokens provides a significant improvement in the storage of the strings as well as better performance in comparing those substrings. *Id.* at 2:27-31. Since each substring (e.g., a subdirectory, filename or extension) is replaced with a numeric value, these numeric values can be arithmetically compared (e.g., is a ==b) instead of string compared (i.e., are all characters the same, what about uppercase vs. lowercase, etc.). *Id.* at 2:32-36. This represents a substantial improvement in performance. *Id.* at 2:36-38. In addition, by keeping a string dictionary, which the token uniquely indexes, only one copy is kept of any substring. *Id.*

at 2:38-39. This too can represent a substantial savings in the amount of storage needed to implement a file system. *Id.* at 2:40-41.

104. The foregoing is set forth in Figures 4-7 and the accompanying text:

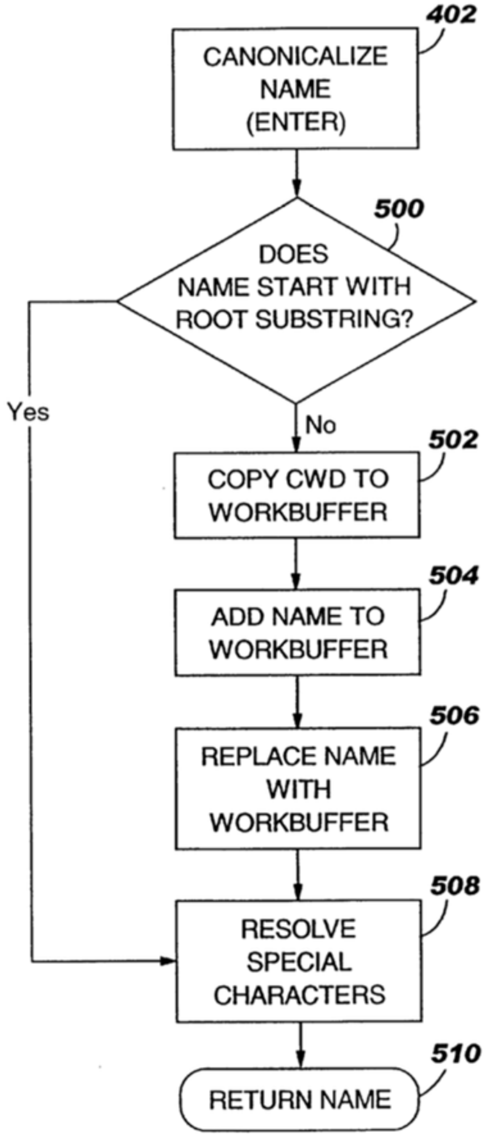
> FIG. 4 illustrates a high-level flowchart of the token replacement process of the present invention. The process starts in entry block 400 in which the current working directory and filename (e.g., current-work-dir=.backslash.dir1.backslash.dir2; name=filename) are input to the canonicalization process as indicated by logic block 402. This action results in the canonical form such as pathname=.backslash.dir1.backslash.dir2.backslash.filename. This is followed in logic block 404 with parsing of the pathname and replacement of substrings with tokens. The substrings in this small example are "dir1", "dir2", and "filename". The result of this action are tokens t1, t2, and t3. The validation of the path is the next act in the process as indicated by logic block 406. From this act the process continues in decision block 408 with a determination of the validity of the path. If the path is found to be invalid an error is returned as indicated by termination block 410. Otherwise, the path is found to be valid and a file system operation is performed as indicated in logic block 412.

FIG. 4


400
ENTER
402
CANONICALIZE NAME
404
PARSE PATHNAME & REPLACE SUBSTRINGS WITH TOKENS
406
VALIDATE PATH
408
PATH VALID?
No
410
RETURN ERROR
Yes
412
PERFORM FILE SYSTEM OPERATION (CREATE, DELETE, OPEN, ETC.

’345 patent at 10:58-65, Fig. 4.

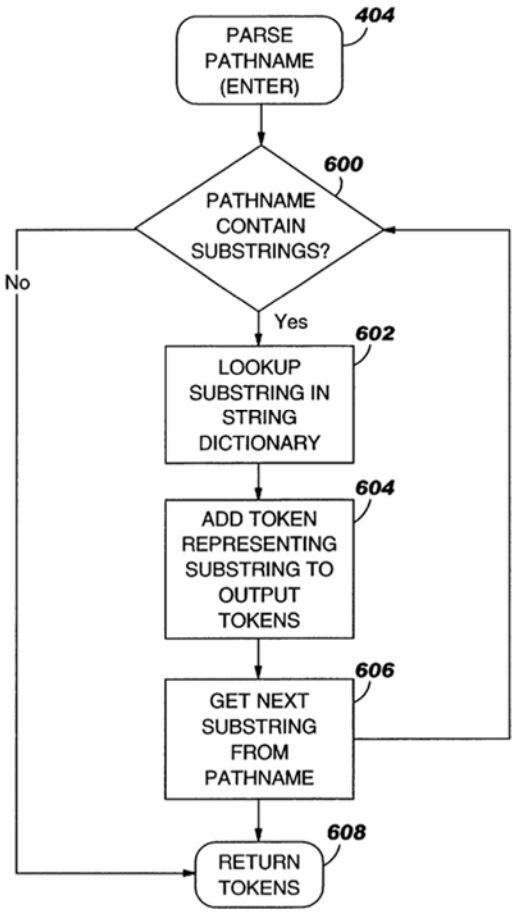
> FIG. 5 illustrates the specific acts of the canonicalization process 402 of FIG. 4. It begins in decision block 500 with a determination if the name starts with a root substring. If it does, then processing jumps to logic block 508 for resolution of special characters in the name. If the name does not start with a root substring, then in logic block 502 the current working directory is copied to a work buffer. The content of the work buffer at this point in the process is .backslash.dir1.backslash.dir2. Next, the name (i.e., filename) is added to the work buffer as indicated in logic block 504. The content of the work buffer at this point

> is .backslash.dir1.backslash.dir2.backslash.filename. In logic block 506, the name is replaced with the work buffer contents. The process concludes in logic block 508 with the resolution of special characters such as ".." or ".". The canonicalization process exits back to the many processing logic in termination block 510.
>
> FIG. 5
>
> 
>

402
> CANONICALIZE NAME (ENTER)
> 500
> DOES NAME START WITH ROOT SUBSTRING?
> Yes
> No
> 502
> COPY CWD TO WORKBUFFER
> 504
> ADD NAME TO WORKBUFFER
> 506
> REPLACE NAME WITH WORKBUFFER
> 508
> RESOLVE SPECIAL CHARACTERS
> 510
> RETURN NAME

’345 patent at 10:66-11:14, Fig. 5.

> FIG. 6 illustrates a flowchart of the parsing process 404 of the present invention. It commences with the entry of decision block 600 which initiates an iterative routine to perform as long as the pathname contains substrings. The iterative routine begins in logic block 602 in which a substring is looked up in the string dictionary. If the

> substring does not exist then a new token is created to represent that substring. In logic block 604, the token representing the substring is added to a list of output tokens for the pathname. The next act is to get the next substring from the pathname as indicated in logic block 606. The iterative routine loops back to decision block 600. After the entire pathname has been parsed into substrings and replaced with tokens (DONE indication out of decision block 600), the parsing process retuns the tokens found as indicated in termination block 608.
>
> FIG. 6
>
> 
>

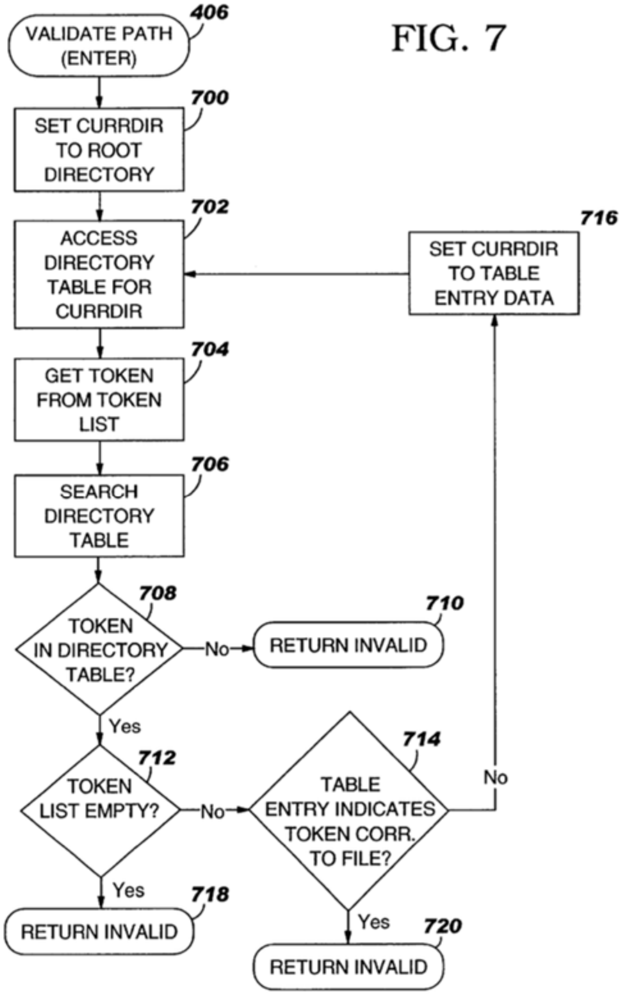
404
> PARSE PATHNAME (ENTER)
> 600
> PATHNAME CONTAIN SUBSTRINGS?
> No
> Yes
> 602
> LOOKUP SUBSTRING IN STRING DICTIONARY
> 604
> ADD TOKEN REPRESENTING SUBSTRING TO OUTPUT TOKENS
> 606
> GET NEXT SUBSTRING FROM PATHNAME
> 608
> RETURN TOKENS

’345 patent at 11:15-11:30, Fig. 6.

> FIG. 7 illustrates a flowchart of the validation process 406 of the present invention. The token list is input to logic block 700 in which the current directory is set to the root directory. In logic block 702, the directory table is accessed for the current directory. This is followed in logic block 704 with the act of getting a token from the token list. Next, in logic block 706, a search is performed to locate the token in the directory table. In decision block 708, a test is made to determine if the token was found in the directory table. If the search failed, then an invalid pathname

indication is returned to the main processing logic via termination block 710. If the search was successful, processing continues in decision block 712, in which a test is made to determine if the token list is empty. If not, the processing continues in decision block 714 in which a determination is made as to whether or not the directory table entry found is for a file (rather than for a directory). If the directory table entry is for a directory, then processing continues in logic block 716 in which the current directory is set to the table entry data; processing then returns to logic block 702. If the directory table entry found in decision block 714 is for a file, then processing ends in termination block 720 with an invalid pathname indication. If, in decision block 712, the token list was found to be empty (i.e., all tokens have been processed) then processing exits in termination block 718 with the return of an valid pathname.

FIG. 7


406
VALIDATE PATH (ENTER)
700
SET CURRDIR TO ROOT DIRECTORY
702
ACCESS DIRECTORY TABLE FOR CURRDIR
716
SET CURRDIR TO TABLE ENTRY DATA
704
GET TOKEN FROM TOKEN LIST
706
SEARCH DIRECTORY TABLE
708
TOKEN IN DIRECTORY TABLE?
No
710
RETURN INVALID
Yes
712
TOKEN LIST EMPTY?
No
714
TABLE ENTRY INDICATES TOKEN CORR. TO FILE?
No
Yes
718
RETURN INVALID
Yes
720
RETURN INVALID

’345 patent at 11:31-11:56, Fig. 7.

105. Figures 8A and 8B contrast the prior art with the inventions of the ’345 patent and the accompanying text explains the advantages of the inventions of the ’345 patent over the prior art:

> FIGS. 8A-8B indicate both the prior art and the inventive method of storing directory and file names on a storage device, such as a disk. FIG. 8A shows a linked list structure with dir1 stored in root block memory location 80, dir2 stored in subdirectory block memory location 82, the filename stored in subdirectory block memory location 84, and the actual file stored at memory location 86. FIG. 8B indicates the method of storing directory and pathnames according to the present invention. Token t1 is stored in root block memory location 90, token t2 is stored in subdirectory block memory location 92, token t3 is stored in subdirectory block memory location 94 which contains a pointer to the file stored at memory location 96. Also shown in FIG. 8B is the string dictionary 98 corresponding to this simple example.

FIG. 8A

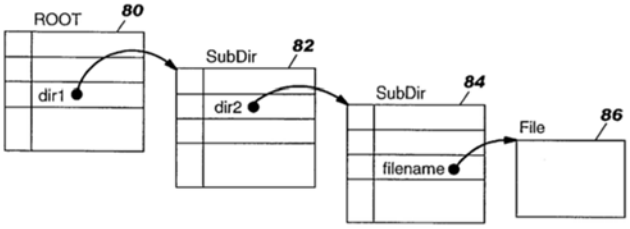
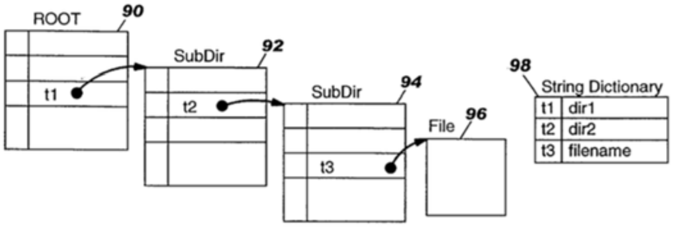
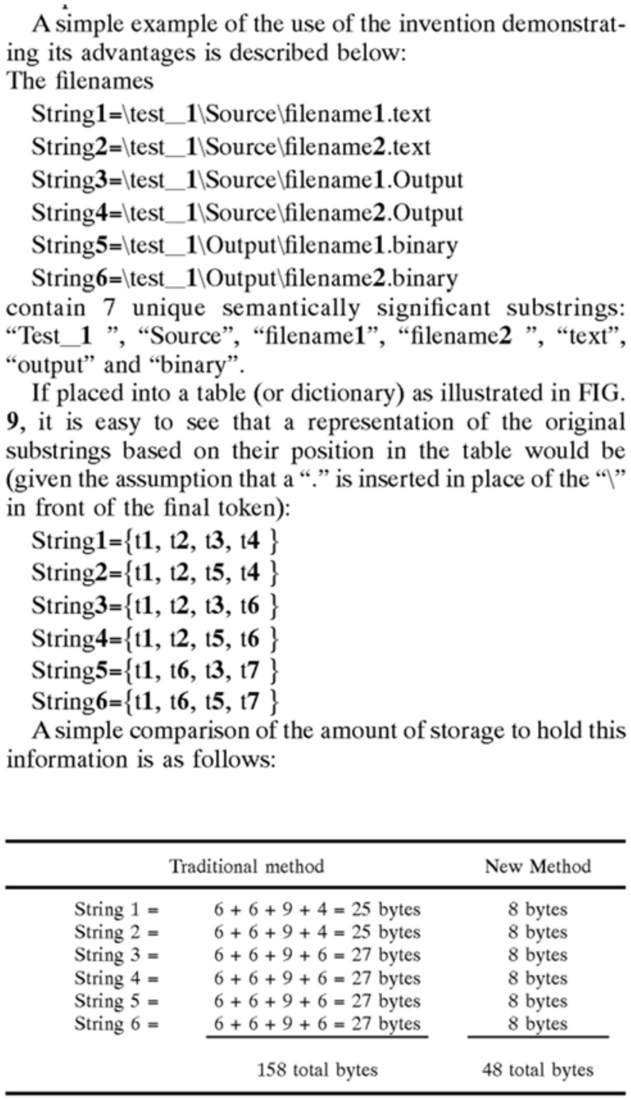


FIG. 8B





A simple example of the use of the invention demonstrating its advantages is described below:

The filenames

String**1**=\test__**1**\Source\filename**1**.text

String**2**=\test__**1**\Source\filename**2**.text

String**3**=\test__**1**\Source\filename**1**.Output

String**4**=\test__**1**\Source\filename**2**.Output

String**5**=\test__**1**\Output\filename**1**.binary

String**6**=\test__**1**\Output\filename**2**.binary

contain 7 unique semantically significant substrings: "Test__**1** ", "Source", "filename**1**", "filename**2** ", "text", "output" and "binary".

If placed into a table (or dictionary) as illustrated in FIG. **9**, it is easy to see that a representation of the original substrings based on their position in the table would be (given the assumption that a "." is inserted in place of the "\" in front of the final token):

String**1**={t**1**, t**2**, t**3**, t**4** }

String**2**={t**1**, t**2**, t**5**, t**4** }

String**3**={t**1**, t**2**, t**3**, t**6** }

String**4**={t**1**, t**2**, t**5**, t**6** }

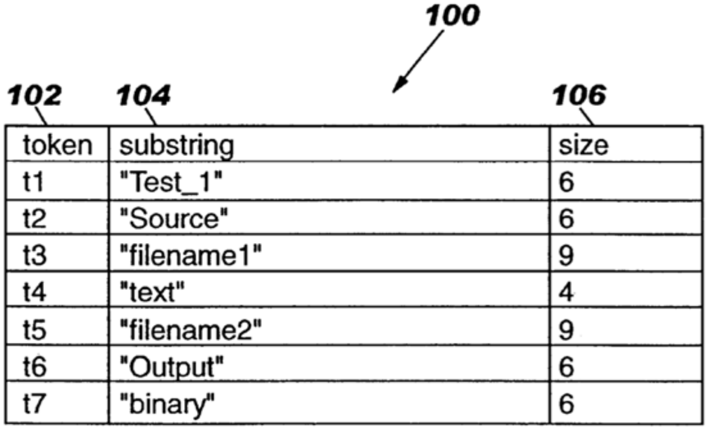
String**5**={t**1**, t**6**, t**3**, t**7** }

String**6**={t**1**, t**6**, t**5**, t**7** }

A simple comparison of the amount of storage to hold this information is as follows:

| | Traditional method | New Method |
|---|---|---|
| String 1 = | 6 + 6 + 9 + 4 = 25 bytes | 8 bytes |
| String 2 = | 6 + 6 + 9 + 4 = 25 bytes | 8 bytes |
| String 3 = | 6 + 6 + 9 + 6 = 27 bytes | 8 bytes |
| String 4 = | 6 + 6 + 9 + 6 = 27 bytes | 8 bytes |
| String 5 = | 6 + 6 + 9 + 6 = 27 bytes | 8 bytes |
| String 6 = | 6 + 6 + 9 + 6 = 27 bytes | 8 bytes |
| | 158 total bytes | 48 total bytes |

FIG. 9



100
102
104
106

| token | substring | size |
|---|---|---|
| t1 | "Test_1" | 6 |
| t2 | "Source" | 6 |
| t3 | "filename1" | 9 |
| t4 | "text" | 4 |
| t5 | "filename2" | 9 |
| t6 | "Output" | 6 |
| t7 | "binary" | 6 |

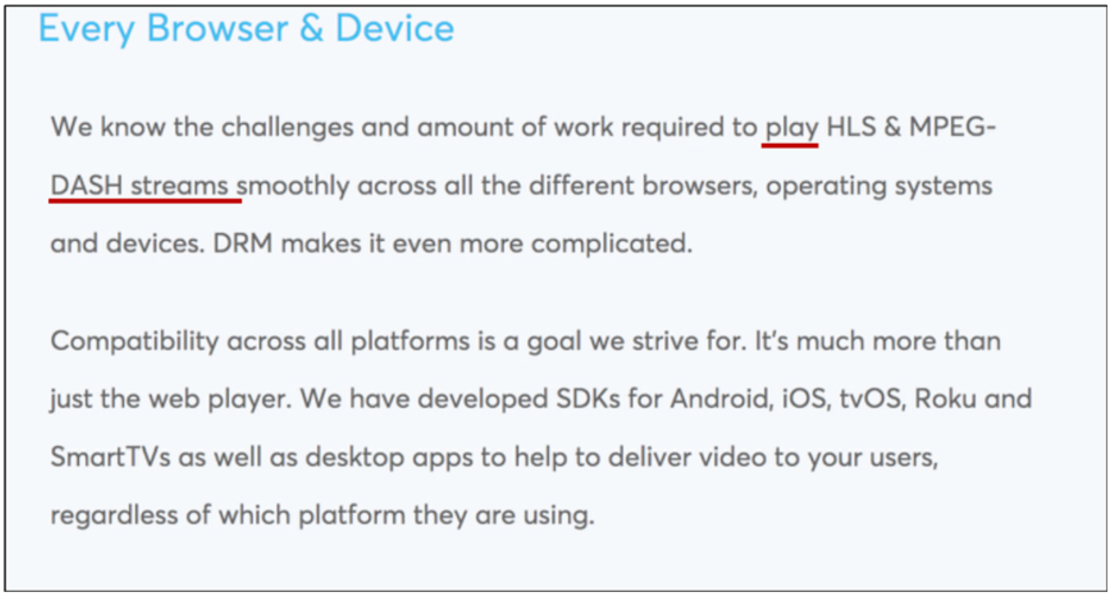
> However, this greater than 3 to 1 comparison ratio is not quite entirely complete in that there is an "overhead" of 81 bytes to store the substrings in a dictionary (as null-terminated strings) along with the pointers to locate them. This overhead, while not negligible, is not as significant as the savings in replacing substrings with 2-byte numeric tokens.
>
> The difference in speed of comparison is not quite so readily calculated. It is clear that comparing a new string:
> StringN=.backslash.Test_1.backslash.Output.backslash.filename2.binary.NEW
>
> with String6, character by character, would involve 32 comparisons of single bytes until a mismatch is found. A simple comparison of the two strings using the token-scheme would require four comparisons of 2-byte tokens.
>
> Again, this 8 to 1 ratio is not entirely complete in that the conversion of the strings into substrings and proper insertion into the table require some overhead, but in a file system where locating information is much more frequent than inserting, removing or renaming it, this overhead is not as significant as the savings in numeric comparisons verus string comparisons.
>
> A third advantage that is usually involved whenever data compression is present is the additional security for a file system that uses the new method. Several schemes could be easily applied to prevent the string dictionary from being accessed even though the file and directory names may be available. This is the "shared-secret" type of security and is the most difficult to decrypt. While the substrings themselves can also be encrypted, it would be easier to take advantage of the clean split between the semantic information embodied in the tokens and the human-readable form of the strings to deter someone from locating secure information in a file system.
>
> The fourth advantage is that of the additional flexibility that tokenizing the substrings provides. Since the actual substrings are stored in a separate place from the directory and file information in the native file system, limits on the length of a substring, overall length of a path (composed of many substrings) as well as the permissible characters in any substring can be much different than those imposed by the native file system. As long as the sequence of tokens can be uniquely mapped to a native file system resource practically any string can be accommodated. The tokens are used only to uniquely represent the substrings, wherever they may be used in a file system name. A clear example is the above use of "Output" as both a sub-directory name and as a file "extension" in String3 and String5 for instance.

’345 patent at 11:57-13:23, Figs. 8A, 8B and 9.

106. As set forth above, claim 1 of the ’345 patent presented an unconventional method for canonicalization for computers that led to better performance of computers and enhanced storage. In light of the foregoing, a person of ordinary skill in the art would understand that claim 1 of the ’345 patent is directed to a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system. Moreover, a person of ordinary skill in the art would understand that claim 1 of the ’345 patent contains that corresponding inventive concept of replacing substrings in file and directory pathnames with tokens in a computer-implemented file system by parsing pathnames and replacing each substring with an associated token and validating the parsed pathname containing a list of tokens.

107. Upon information and belief, Bitmovin has directly infringed at least claim 1 of the ’345 patent by making, using, testing, selling, offering for sale, importing and/or licensing in the United States without authority products and services that perform a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, including an MPEG-DASH compatible video player (collectively “the ’345 Accused Infringing Devices”) in an exemplary manner as described below.

108. The ’345 Accused Infringing Devices perform a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system. The ’345 Accused Infringing Devices include a MPEG-DASH compatible video player. DASH video streams include a media presentation description (MPD) which is a manifest of the media segments that make up the complete media presentation. The MPD contains file and directory pathnames to access these segments in the form of HTTP URLs.



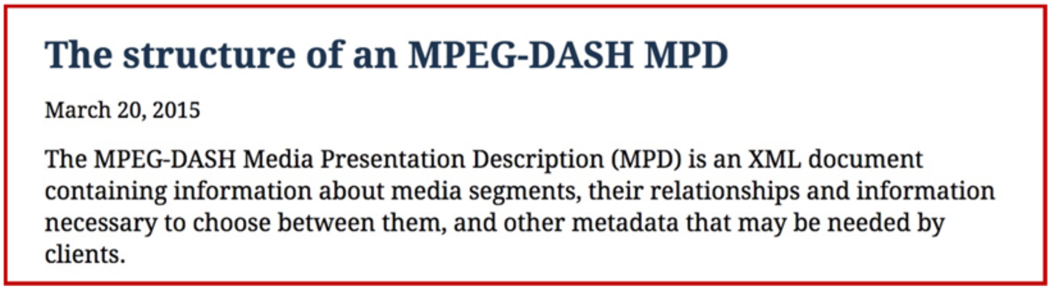
## Every Browser & Device

We know the challenges and amount of work required to play HLS & MPEG-DASH streams smoothly across all the different browsers, operating systems and devices. DRM makes it even more complicated.

Compatibility across all platforms is a goal we strive for. It's much more than just the web player. We have developed SDKs for Android, iOS, tvOS, Roku and SmartTVs as well as desktop apps to help to deliver video to your users, regardless of which platform they are using.

**Source:** https://bitmovin.com/video-player/

109. MPEG DASH in the '345 Accused Infringing Devices has a mechanism whereby URLs to access segment files can use a SegmentTemplate to specify file and pathnames. This mechanism allows DASH video players to replace specific substrings (identifiers) in the template with dynamic numbers (tokens) in a computer implemented file system (URLs).

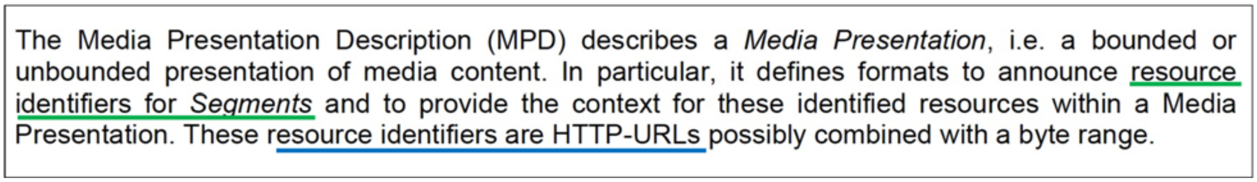


## The structure of an MPEG-DASH MPD

March 20, 2015

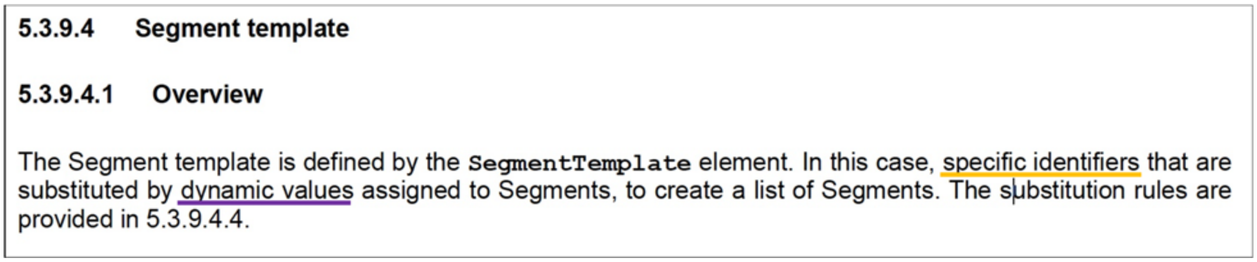
The MPEG-DASH Media Presentation Description (MPD) is an XML document containing information about media segments, their relationships and information necessary to choose between them, and other metadata that may be needed by clients.

**Source**: https://www.brendanlong.com/the-structure-of-an-mpeg-dash-mpd.html



The Media Presentation Description (MPD) describes a *Media Presentation*, i.e. a bounded or unbounded presentation of media content. In particular, it defines formats to announce resource identifiers for *Segments* and to provide the context for these identified resources within a Media Presentation. These resource identifiers are HTTP-URLs possibly combined with a byte range.

**Source**: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p7

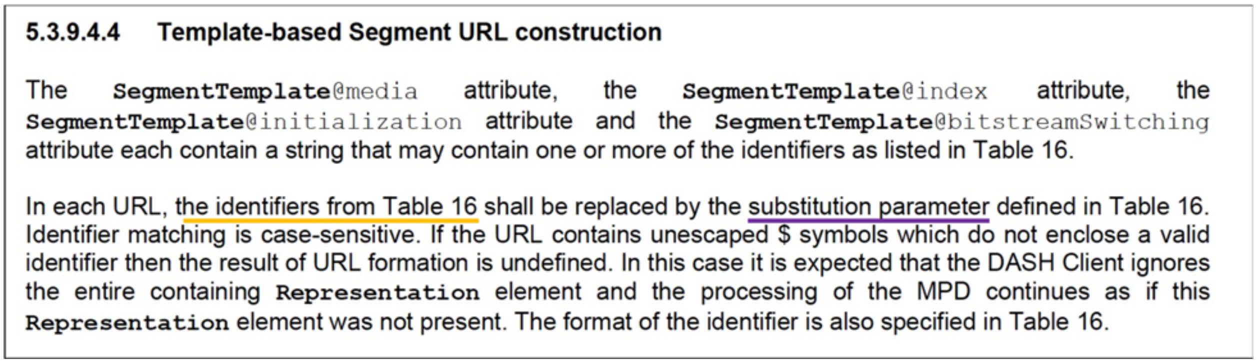


**5.3.9.4 Segment template**

**5.3.9.4.1 Overview**

The Segment template is defined by the **`SegmentTemplate`** element. In this case, specific identifiers that are substituted by dynamic values assigned to Segments, to create a list of Segments. The substitution rules are provided in 5.3.9.4.4.

**Source**: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p53



**5.3.9.4.4 Template-based Segment URL construction**

The **`SegmentTemplate`**`@media` attribute, the **`SegmentTemplate`**`@index` attribute, the **`SegmentTemplate`**`@initialization` attribute and the **`SegmentTemplate`**`@bitstreamSwitching` attribute each contain a string that may contain one or more of the identifiers as listed in Table 16.

In each URL, the identifiers from Table 16 shall be replaced by the substitution parameter defined in Table 16. Identifier matching is case-sensitive. If the URL contains unescaped $ symbols which do not enclose a valid identifier then the result of URL formation is undefined. In this case it is expected that the DASH Client ignores the entire containing **`Representation`** element and the processing of the MPD continues as if this **`Representation`** element was not present. The format of the identifier is also specified in Table 16.

**Source**: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p53

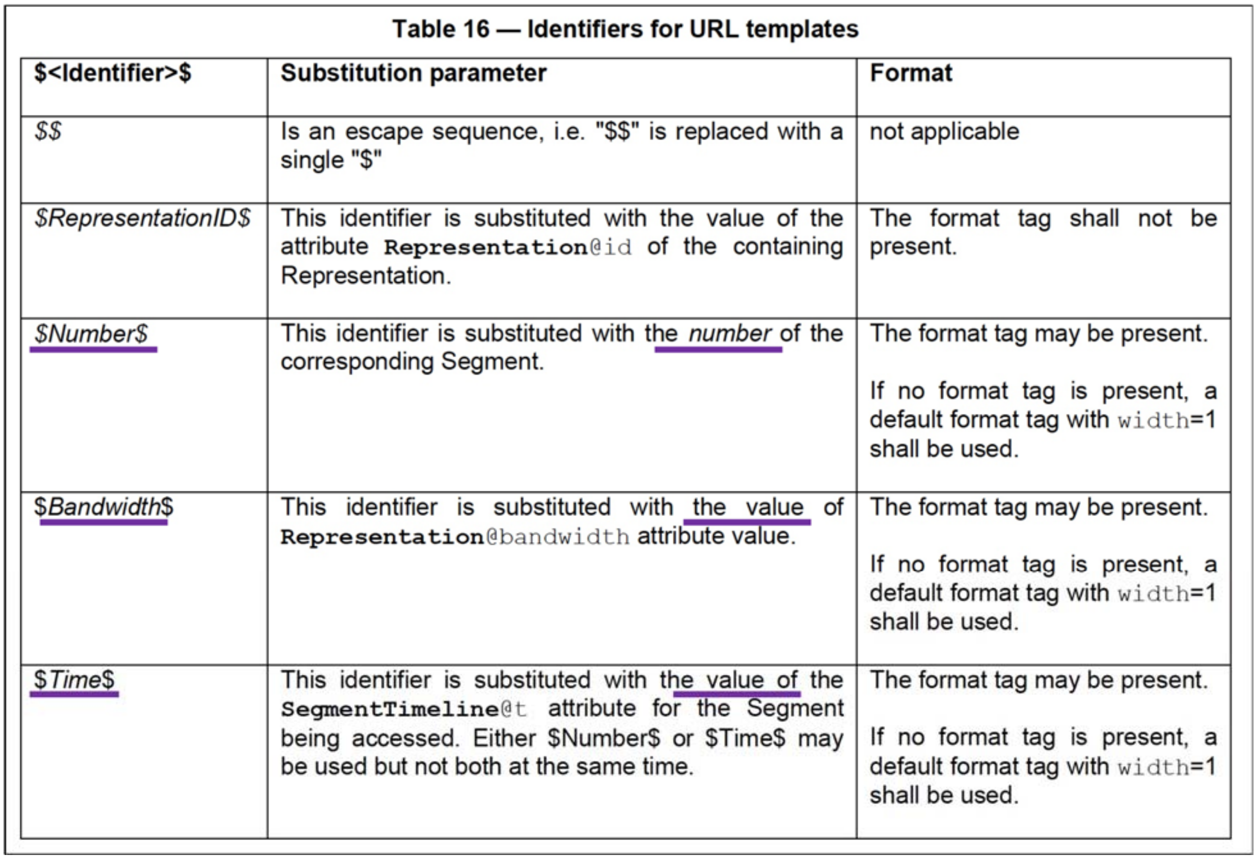


**Table 16 — Identifiers for URL templates**

| $<Identifier>$ | Substitution parameter | Format |
|---|---|---|
| *$$* | Is an escape sequence, i.e. "$$" is replaced with a single "$" | not applicable |
| *$RepresentationID$* | This identifier is substituted with the value of the attribute **Representation**@id of the containing Representation. | The format tag shall not be present. |
| *$Number$* | This identifier is substituted with the *number* of the corresponding Segment. | The format tag may be present.<br>If no format tag is present, a default format tag with width=1 shall be used. |
| $*Bandwidth*$ | This identifier is substituted with the value of **Representation**@bandwidth attribute value. | The format tag may be present.<br>If no format tag is present, a default format tag with width=1 shall be used. |
| $*Time*$ | This identifier is substituted with the value of the **SegmentTimeline**@t attribute for the Segment being accessed. Either $Number$ or $Time$ may be used but not both at the same time. | The format tag may be present.<br>If no format tag is present, a default format tag with width=1 shall be used. |

**Source**: ISO IEC 23009-1:2014, “Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats”, p55

110. The ’345 Accused Infringing Devices name a string to be converted into a list of tokens. For example, the ’345 Accused Infringing Devices read DASH MPD files to play media. MPD files can include SegmentTemplates with name strings according to the ISO IEC 23009-1 specification.

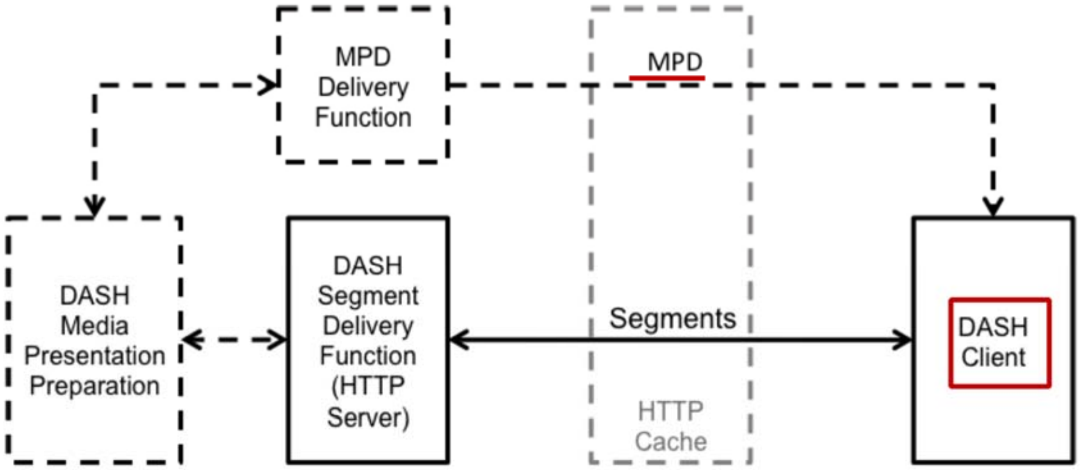
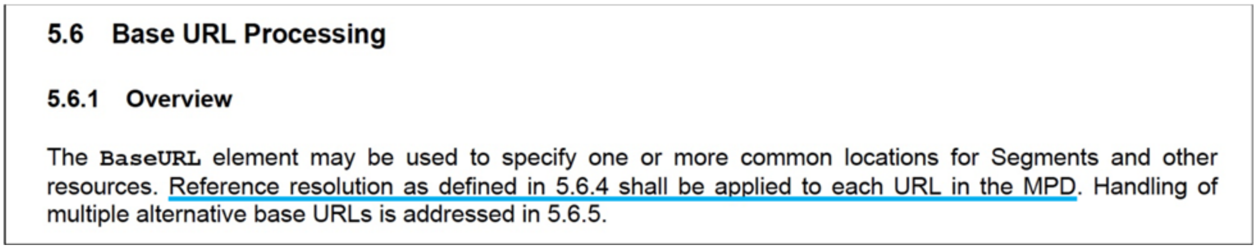

MPD Delivery Function
MPD
DASH Media Presentation Preparation
DASH Segment Delivery Function (HTTP Server)
Segments
HTTP Cache
DASH Client

Figure 1 — Example system for DASH formats

**Source**: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats," p8

111. The '345 Accused Infringing Devices canonicalize a current working directory and the name string to form a pathname containing a plurality of substrings. For example, the '345 Accused Infringing Devices use a canonicalization process which converts the partial path/file name in the template into a complete path/file name using the MPEG DASH BaseURL mechanism. The MPEG-DASH specification requires that URL references in an MPD use reference resolution (canonicalization) for each URL in the MPD, including those related to media segments.



### 5.6 Base URL Processing

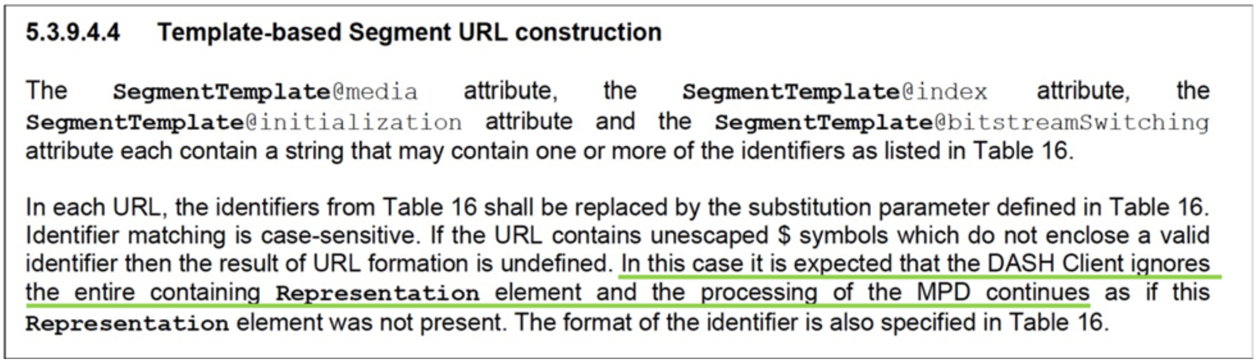
#### 5.6.1 Overview

The `BaseURL` element may be used to specify one or more common locations for Segments and other resources. Reference resolution as defined in 5.6.4 shall be applied to each URL in the MPD. Handling of multiple alternative base URLs is addressed in 5.6.5.

**Source**: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p64

112. The ’345 Accused Infringing Devices, according to the required behavior in the MPEG-DASH specification, parse the pathname and replace the substrings in Table 16 with the associated token.

113. The ’345 Accused Infringing Devices validate the parsed pathname and should ignore invalid pathnames within the context (Representation) in which they were defined.



**5.3.9.4.4 Template-based Segment URL construction**

The **SegmentTemplate**@media attribute, the **SegmentTemplate**@index attribute, the **SegmentTemplate**@initialization attribute and the **SegmentTemplate**@bitstreamSwitching attribute each contain a string that may contain one or more of the identifiers as listed in Table 16.

In each URL, the identifiers from Table 16 shall be replaced by the substitution parameter defined in Table 16. Identifier matching is case-sensitive. If the URL contains unescaped $ symbols which do not enclose a valid identifier then the result of URL formation is undefined. In this case it is expected that the DASH Client ignores the entire containing **Representation** element and the processing of the MPD continues as if this **Representation** element was not present. The format of the identifier is also specified in Table 16.

**Source**: ISO IEC 23009-1:2014, “Information technology — Dynamic adaptive streaming over HTTP (DASH) —Part 1: Media presentation description and segment formats,” p. 54

114. Bitmovin has thus infringed at least claim 1 of the ’345 patent by making, using, testing, selling, offering for sale, importing and/or licensing the ’345 Accused Infringing Devices, and operating them such that all steps of at least claim 1 are performed.

115. Bitmovin’s acts of direct infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin’s wrongful acts in an amount subject to proof at trial.

## PRAYER FOR RELIEF

WHEREFORE, Uniloc 2017 respectfully requests the following relief:

A. A judgment that Bitmovin has infringed the '712 patent;

B. A judgment that Bitmovin has infringed the '118 patent;

C. A judgment that Bitmovin has infringed the '005 patent;

D. A judgment that Bitmovin has infringed the '345 patent;

E. A judgment that Uniloc be awarded damages adequate to compensate it for Bitmovin's past infringement and any continuing or future infringement of the '712 patent, the '118 patent, the '005 patent and the '345 patent, including pre-judgment and post-judgment interest costs and disbursements as justified under 35 U.S.C. § 284 and an accounting;

F. That this be determined to be an exceptional case under 35 U.S.C. § 285;

G. That Uniloc be granted its reasonable attorneys' fees in this action;

H. That this Court award Uniloc its costs; and

I. That this Court award Uniloc such other and further relief as the Court deems proper.

## DEMAND FOR JURY TRIAL

Uniloc hereby demands trial by jury on all claims and issues so triable.

DATED: April 9, 2019

Respectfully submitted,

**FARNAN LLP**

*/s/ Michael J. Farnan*
Brian E. Farnan (Bar No. 4089)
Michael J. Farnan (Bar No. 5165)
919 North Market Street, 12th Floor
Wilmington, DE 19801
phone 302-777-0300
fax 302-777-0301
bfarnan@farnanlaw.com
mfarnan@farnanlaw.com

M. Elizabeth Day (admitted *pro hac vice*)
David Alberti (admitted *pro hac vice*)
Sal Lim (admitted *pro hac vice*)
Marc Belloli (admitted *pro hac vice*)
**FEINBERG DAY ALBERTI LIM & BELLOLI LLP**
1600 El Camino Real, Suite 280
Menlo Park, CA 94025
Tel: 650.618.4360
Fax: 650.618.4368
eday@feinday.com
dalberti@feinday.com
slim@feinday.com
mbelloli@feinday.com

*Attorneys for*
Uniloc 2017 LLC