Compliance Procedures for Dynamic Adaptive Streaming over HTTP (DASH)

KASHAF MAZHAR

Master's Degree Project
Stockholm, Sweden

XR-EE-LCN 2011:011
Compliance Procedures for Dynamic Adaptive Streaming over HTTP – (DASH)

Master of Science Thesis

KTH
The Royal Institute of Technology
School of Electrical Engineering

Date: 2011-07-30
Munich, Germany 2011
Nomor Research GmbH
Brecherspitzstrasse 8, Munich, Germany

www.nomor.de

Master Thesis

Conformance Procedures for Dynamic Adaptive Streaming over HTTP

By
Kashaf Mazhar

Supervisor
Dr.-Ing. Thomas Stockhammer, CEO and Founder of Nomor Research GmbH

Examiner
Viktoria Fodor, Associate Professor, Laboratory for Communication Networks, KTH, Stockholm
ABSTRACT

MPEG has been actively contributing towards the multimedia side in terms of storage and delivery of media content with flexible and advanced file formats like the MPEG-2 transport stream and lately the MPEG 4 formats. Adaptive Streaming over HTTP is relatively a new genre in the area of streaming where this diverse content is delivered to TV's, portable data units, smart phones, home set top boxes etc. Most common of these streaming in practice are the apple live streaming, Smooth Streaming by Microsoft, HTTP Adaptive streaming as per the 3GPP specifications and the MPEG specified Dynamic Adaptive streaming over HTTP (DASH). The technology is based on intelligence at clients end to cope up with variable network conditions and optimize the performance in an automated fashion.

As the DASH specification is being finalized measures are to be taken to define criteria that will ensure the validation of a DASH process. Rules and conditions specified in the specification need to be followed by the client and server interaction as well as the content that will actually constitute the streaming process. This conformance is necessary to ensure the properties defined for a streaming process to provide the best user experience and performance efficiency. In light of this an initiative in this area of conformance testing is required so that the future deployment and implementation of DASH has a baseline to follow.

The initial outline lay out by MPEG and DASH is followed to formulate and later implement the rules for the interaction between the client and the server. Secondly the content used in streaming is also checked for properties and attributes mentioned in the MPD and the ones necessary for DASH process. These efforts will help define the criteria required for DASH validation as well as a ground work that can be extended for further work in this domain. In summary, the main tasks are

- Development of last stage of the integrated software for MPD conformance stages (Java Script)
- Formulation of Rules for Conformance from DASH specifications.
- Extension to the existing Qualcomm DASH Reference software to support conformance test on media content level

The thesis mainly concentrates on logs that are the result of conformance processing. The logs are analyzed to see whether the condition specified are fulfilled or not. The process can be regarded as a 'YES' or 'NO' result indicating the fault/error that occurred.
Acknowledgements

I would like to thank my supervisor Dr.-Ing. Thomas Stockhammer (CEO, Nomor Research GmbH) for providing me the opportunity to work with him and prepare my thesis. I would also like to thank Waqar Zia, M.Sc. who helped me through my thesis at Nomor and advised me during my work.

At KTH, I would thank Associate Professor Viktoria Fodor for supervising my thesis and all Professors who taught me during my master’s program at KTH.

I would also like to thank my parents for their support during my master’s degree.

Munich, Germany
Kashaf Mazhar
LIST OF ABBREVIATIONS

DASH- Dynamic Adaptive Streaming over HTTP
CDN- Content Distribution Network
ISO- International Standard Organization
3GPP- 3rd Generation Partnership Project
MPEG2-TS- MPEG-2 Transport Stream
HTTP- Hypertext transfer Protocol
URL- Uniform Resource Locator
MPD- Media Presentation description
SIDX- Segment Index
MOOF- Movie Fragment
FTYP- File Type
PdIn- Progressive Download Information
MOOV- Movie Box
TFAD- Track Fragment Adjustment
TFDT- Track Fragment Decode Time
STYP- Segment Type
MPD- Media Presentation Description
XML- Extensible Markup Language
XLink- XML Linking Language
URI- Uniform Resource Identifier
XSLT- Extensible Style Sheet Language
SVRL- Schematron validation Report Language
ISOBMFF- ISO Base Media File Format
MFHD- Movie Fragment Header
TRAF- Track fragment
TFHD- Track Fragment Header
TRUN- Track Fragment Run
EPT- Earliest Presentation Time
LPT- Latest Presentation Time
SSEG- Sub Segment
SEG- Segment
SAP- Stream Access Point
RAP- Representation Access Point
Trap- Presentation Time of RAP
Tsap- Presentation Type of SAP
# Contents

<table>
<thead>
<tr>
<th>CHAPTER 1</th>
<th>INTRODUCTION</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Overview</td>
<td>15</td>
</tr>
<tr>
<td>1.2</td>
<td>Goals for the Thesis</td>
<td>15</td>
</tr>
<tr>
<td>1.3</td>
<td>Basis of Thesis Work</td>
<td>16</td>
</tr>
<tr>
<td>1.4</td>
<td>Structure of Thesis</td>
<td>17</td>
</tr>
<tr>
<td>CHAPTER 2</td>
<td>DASH PRINCIPLES</td>
<td>18</td>
</tr>
<tr>
<td>2.1</td>
<td>HTTP Media Streaming</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Common Streaming Services</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>DASH DESIGN</td>
<td>18</td>
</tr>
<tr>
<td>2.3.2</td>
<td>DASH Data Model</td>
<td>19</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Media Presentation Description Element</td>
<td>20</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Period Element</td>
<td>21</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Groups</td>
<td>21</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Representation</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>MPD Structure</td>
<td>22</td>
</tr>
<tr>
<td>2.5.1</td>
<td>DASH Components</td>
<td>22</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Segment Format</td>
<td>22</td>
</tr>
<tr>
<td>2.5.2.1</td>
<td>Initialization Segment</td>
<td>22</td>
</tr>
<tr>
<td>2.5.2.2</td>
<td>Media segment</td>
<td>23</td>
</tr>
<tr>
<td>2.6</td>
<td>Segment Indexing</td>
<td>23</td>
</tr>
<tr>
<td>CHAPTER 3</td>
<td>DASH MPD CONFORMANCE PROCEDURES</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Conformance Procedure of Dynamic Adaptive Streaming</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Media Presentation Description Compliance</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1</td>
<td>MPD – AS XML Document</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2</td>
<td>XLink Support for XML Documents</td>
<td>27</td>
</tr>
<tr>
<td>3.2.3</td>
<td>XML Validation via XML Schema</td>
<td>27</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Schematron Validation</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>DASH Media presentation Description Conformance Implementation</td>
<td>31</td>
</tr>
<tr>
<td>3.3.1</td>
<td>XLINK Resolver</td>
<td>31</td>
</tr>
<tr>
<td>3.3.1.1</td>
<td>Check for Circular Referencing</td>
<td>32</td>
</tr>
<tr>
<td>3.3.1.2</td>
<td>Check Invalid Protocol Usage</td>
<td>33</td>
</tr>
<tr>
<td>3.3.1.3</td>
<td>Check Wrong Element Referencing</td>
<td>34</td>
</tr>
</tbody>
</table>
3.3.2 XML Schema Validation 
3.3.3 Schematron Validation 
3.3.3.1 Results for Schematron Rules Checking

CHAPTER 4 FILE FORMAT CONFORMANCE ................................................................. 46
4.1 File format conformance checks ....................................................................... 46
4.2 ISO Base Media File Format specified for DASH .............................................. 46
4.2.1 Initialization Segment 
4.2.1.1 FTYP (File Type Box) .............................................................................. 47
4.2.1.2 MOOV BOX (Movie Box) ......................................................................... 48
4.2.1.3 MVEX Box ............................................................................................... 49
4.2.1.4 PdIn Box .................................................................................................. 49
4.2.2 Media Segment for DASH 
4.2.2.1 STYP Box .................................................................................................. 50
4.2.2.2 SIDX Box .................................................................................................. 50
4.2.2.2 MOOF Box ............................................................................................... 51
4.2.2.4 MFHD ....................................................................................................... 51
4.2.2.5 TRAF ......................................................................................................... 52
4.2.2.6 TFHD ......................................................................................................... 52
4.2.2.7 TFDT ......................................................................................................... 52
4.2.2.8 TRUN ........................................................................................................ 52
4.3 Sub Indexed Media Segments .......................................................................... 53
4.4 File Format Conformance Checking ................................................................ 53
4.6 ISO Base media File Format Level ................................................................... 54
4.7 Bit stream Level .................................................................................................. 55
4.7.1 Segment Alignment Flag 
4.7.1.1 Implementation Details for Conformance .................................................. 56
4.7.1.2 Conformance Checks for Segment Alignment ........................................... 58
4.7.2 SUB SEGMENT ALIGNMENT FLAG ............................................................ 58
4.7.2.1 Conformance Checks for Segment Alignment ........................................... 60
4.7.3 “STARTWITHRAP” Conformance Check 
4.7.3.1 Case: With no Sub-segments (no SIDX) ..................................................... 61
4.7.3.2 Case: With Sub-segments (SIDX box present) ........................................... 62
4.7.3.3 Conformance Checks for STARTWITHRAP ........................................... 63
4.7.4 Bit Stream Switching Flag 

KTH, Royal Institute of Technology
4.7.5 MAX RAP PERIOD Attribute Check

4.8 Hierarchal Rules

CHAPTER 5 SOFTWARE IMPLEMENTATION DETAILS

5.1 Software Implementation of FILE FORMAT conformance checks

5.3 Results for Conformance Procedure

CHAPTER 6 FUTURE WORK & CONCLUSIONS

6.1 Future Work

6.2 CONCLUSION

References

APPENDIX A: SCHEMATRON SCHEMA

APPENDIX B: DASH SCHEMA
LIST OF FIGURES

FIGURE 1 - HTTP MEDIA STREAMING SCHEMATIC [1] 18
FIGURE 2 - DASH ARCHITECTURE [2] 19
FIGURE 3 - DASH DATA MODEL [2] 20
FIGURE 4 - DASH STRUCTURES & COMPONENTS [2] 22
FIGURE 5 - MPD CONFORMANCE SCHEMATIC [10] 26
FIGURE 6 - SAMPLE MPD 27
FIGURE 7 - DASH SCHEMA SAMPLE 28
FIGURE 8 - XSLT PROCESSING 29
FIGURE 9 - DASH SCHEMATRON SCHEMA SAMPLE 30
FIGURE 10 - SCHEMATRON PROCESSING STAGES [15] 30
FIGURE 11 – CIRCULAR REFERENCING EXAMPLE - "CIRCULAR_REF.XML" 32
FIGURE 12 - CIRCULAR REFERENCING EXAMPLE - "EX_INCLUDE.XML" 33
FIGURE 13 - CIRCULAR REFERENCING EXAMPLE - "EX_CIRCULAR.XML" 33
FIGURE 14 - INVALID PROTOCOL USE IN MPD 34
FIGURE 15 - WRONG REFERENCE - MAIN MPD (WRONGELEMENT.XML) 34
FIGURE 16 - WRONG REFERENCE - REFERENCED ELEMENT (REF-WRONGELEMENT.XML) 34
FIGURE 19 - SCHEMATRON VALIDATION SCHEMATIC FULL [14] 36
FIGURE 17 - MPD - EXTRACTED PERIOD ELEMENT 36
FIGURE 18 - EXTRACT FROM DASH SCHEMA 36
FIGURE 20 –CASE 1 - TEST MPD 43
FIGURE 21 – CASE 1: SCHEMATRON OUTPUT RESULT 43
FIGURE 22 - CASE 2: TEST MPD 44
FIGURE 23 - CASE 2: SCHEMATRON OUTPUT RESULT 44
FIGURE 24 - INITIALIZATION SEGMENT CONSTRUCTION 47
FIGURE 25 - FIELDS OF 'STSD' BOX INSIDE MOOV BOX 48
FIGURE 26 - CONSTRUCTION OF MEDIA SEGMENT 50
FIGURE 27 - FILE FORMAT CONFORMANCE SCHEMATIC 54
FIGURE 28 - SEGMENT ALIGNMENT SCHEMATIC 57
FIGURE 29 - EARLIEST PRESENTATION TIME OF SSEG 59
FIGURE 30 - LATEST PRESENTATION TIME OF SSEG 60
FIGURE 31 - DASH PROCESS LOGS 64
FIGURE 32 - DASH SW CLIENT MODEL 65
FIGURE 33 - SW IMPLEMENTATION SCHEMATIC 67
FIGURE 34 - CONTENT GENERATION PROCESS 68
FIGURE 35 - SUB SEGMENT REQUEST INFORMATION 69
FIGURE 36 - INITIALIZATION SEGMENT & SIDX CHECKING 70
FIGURE 37 – MEDIA SEGMENT CHECKS 70
FIGURE 38 - CONFORMANCE DATA BASE ENTRIES 73

LIST OF TABLES

TABLE 1 - COMPARISON OF STREAMING SERVICES [3] 24
TABLE 2 - DASH VALIDATION RULE COLLECTION 37
TABLE 3 - DASH SCHEMATRON CONFORMANCE RULES 39
TABLE 4 - SIDX BOX ENTRIES 51
TABLE 5 - ISOBMFF LEVEL CONFORMANCE CHECKS 54
TABLE 6 - CONFORMANCE CLASS PARAMETERS 66

KTH, Royal Institute of Technology 14
CHAPTER 1 INTRODUCTION

1.1 Overview

With the advent of time multimedia technology is growing very rapidly and several measures are taken to enhance the user experience. Since availability of high data rates is becoming more widespread, technologies are being developed to capitalize on this advantage. In this regard, DASH (Dynamic Adaptive Streaming over HTTP) comes as a 3GPP standardized model for streaming services. It incorporates features that enable the client server model of the existing infrastructure to work more efficiently. Main focus shifts to client side as the decision process is under the client’s control and hence the intelligence as well. Although the specification is quite new but work is under progress to finalize the standard.Alongside that implementation models for DASH are also in practice to verify the standard and develop platforms for commercial use as well.

DASH also addresses the shortcomings of the streaming technologies in market so making it the most robust and effective technology. At present there are numerous available options for streaming videos available but the restrictions and shortcomings associated with each are also present. We can encounter videos with accessibility problems by finding plug-in not available, bandwidth not sufficient or having the wrong format for play out. Then the quality of experience also plays as a vital part in video streaming by influencing factors like start up delay, frequent rebuffing of the content, degradation of playback quality and lip sync issues [1]. Expensive streaming service demanding more bandwidth and dedicated devices add up to the cost of the service. So as whole streaming over the web can pose out to be potential problem when all these aspects are taken into account. To eradicate the complexities, efforts in the direction of building confidence in an open standard is being encouraged so that the common format is followed to maximize the quality and to overcome all the previous shortcomings in this aspect.

1.2 Goals for the Thesis

The main objective of the thesis is to formulate compliance procedure of DASH. The need for conformance procedures focused in the thesis can be summarized as:

- It forms the outline for the interaction between the client and server for the exchange of information before the streaming begins. This interaction needs to follow the rules and regulations laid out by the DASH standard itself. So the main aim is to formulate the rules and then implement them so that common grounds for DASH interaction can be laid out. Services that conform to these grounds can be regarded as DASH. In this regard, the first part of conformance is already presented by DASH committee via a three stage process of MPD Conformance. Task at hand is to implement and integrate the entire process in a JAVA script. For software implementation, rules for XLink Validation are extracted from the DASH specification and the Schema Validation is aided by the DASH Schema already specified in the standard. The major task in this stage is the rule based validation i.e Schematron Validation. No prior work has been done regarding the rules formulation so it required that the specification be scrutinized to extract rules at each stage of the MPD for conformance. Later Schematron is used to implement the rules for the MPD conformance procedures. These rules formulation and its implementation in Java Script forms a major part of the thesis objectives.
Then the file format conformance is investigated to correlate the information exchanged and the media content itself that is used by the streaming service. Providing cross reference mechanisms confirms that the content that is available on the server for streaming is following DASH specification. This is important to check because the features and efficiency that DASH proclaims is possible only via the existence of the attributes that are being checked by conformance procedures.

This process is a new initiative and no prior work is done in this respect. The work is based on the scope of file format provided by figure 27. Firstly, the thesis focuses on software implementation at ISOBMFF Level by extending the QUALCOMM reference software to cater for conformance rules derived from the DASH specification for the streaming content to follow. This mainly concerns the rules and conditions for initialization and media segments to be formulated and then implemented. Secondly the main area of focus is the Bit Stream Level checks that requires clarifying definitions of attributes in the DASH standard by checking them for practical aspects. The vague definitions initially included for attributes at different stages of the MPD are checked in the software domain and then amendments are proposed if needed. The attribute that are related with the media content are selected to be checked and logs are generated via the conformance software developed so that any content that claims to be DASH compliant can be confirmed by passing it through the software conformance stage and analyzing the logs for results. Formation of steps for checking the attributes and then its software logic creation forms a major task for the thesis at this stage of conformance.

As the standardization process is underway it is important to clarify the implementation details for the process as well. In light of this a link needs to be created between the theoretical concepts mentioned and the practical implementation required for that. The conformance procedures will serve this purpose as the implementation part of it will test the concepts of DASH in practice and will provide grounds to make amendments in the specification where required.

The conformance procedure at this stage will lay out the initial ground work for the further development of conformance testing with added features.

This will provide the particular details to prospective DASH Software implementations that are needed to be followed and taken care of.

1.3 Basis of Thesis Work

The thesis is mainly based on the 3GPP Dynamic Adaptive streaming and MPEG system technology for dynamic adaptive streaming over HTTP. The conformance procedures follow these specifications and later are translated into practical implementation. As for the software implementation part, DASH content generation and DASH client server model is tested on the “Reference Software” provided by “QUALCOMM”. This software is extended to produce conformance procedures and the required results. Modification to the existing software is also done to cope up with the amendments in standard. Currently, a claimed DASH implementation by Paris Tech is also present in the form of GPAC tool name “MP4Box”. Conformance procedures defined in thesis will help distinguish the shortcomings in these implementations that can be rectified. As for the information exchanged between client and server the tools produced by University of Klagenfurt are also used to achieve compliance checks at that level.
1.4 Structure of Thesis

The thesis is segregated mainly into four parts. In chapter 2, a general overview of DASH is provided with the explanation for some of the key factors and features that DASH includes. The overview provided makes a foundation for the further discussion on DASH.

Chapter 3 takes a start on the conformance checks by concentrating on the first stage of compliance i.e the media presentation description exchanged between the client and server. The process encapsulates all the stages for MPD (Media Presentation Description) conformance and signifies the importance of each. Mainly the area of focus is the Schematron level checking that actually enables the rules application.

Then moving on to the major phase of file format conformance, Chapter 4 is based on the checks applied to structural components of media structure to ensure a synchronization between the information provided and the content that is available for the streaming process. This process is segregated into two stages that are investigated by first mentioning the rules applied and then later providing details on how to implement the checks at the bit stream level.

Lastly, Chapter 5 provides the overview of the software implementation part of thesis that provides means for the theoretical concepts in the standard to be tested at the software implementation level. The main model used and software details are provided in this section. Guidelines in creating DASH content are also provided that were necessary to test content for results. Results in the form of logs produced by the reference software are also provided to correlate the file format checks discussion with the actual results. Finally the future work required in this process is mentioned.
CHAPTER 2 DASH PRINCIPLES

2.1 HTTP Media Streaming

The streaming service comprises mainly of a client server architecture where the client requests content from the server as per demand. To elaborate the process and interaction between the two entities, there is division that forms the interaction levels between the client and server namely [1]. Figure 1 mainly lays out an outline of a streaming interaction with some generic elements like

- (MF) Manifest format: Manifest file exchange
- (DF) Delivery format: The media content is delivered in a special format depending on the type
- Media Content Base: ISOBMFF (ISO base media file format or MPEG2-TS that originally constitute the content are easily converted to the delivery format for the streaming service)

![Figure 1 - HTTP Media Streaming Schematic [1]](image)

2.2 Common Streaming Services

At present many proprietary services are in use commercially which follow the concept of HTTP streaming. Commonly renowned services include Apple http live streaming and adobe HTTP streaming. Although the services are currently in use but they are limited to certain factors which also contributed to the need to DASH standardization. Firstly, the services had proprietary underlying protocols and processes that limited the scope to investigate and make further improvement and in addition it still followed the pattern of a combined audio and video stream that again posed a limit to the adaptivity and flexibility of the streaming itself [1]. Furthermore, new streaming technologies are emerging that are incorporating the concept of separate audio video streams like Netflix and Microsoft Smooth streaming. In wake of these efforts the ISO is working to develop a standardized version of the dynamic adaptive streaming that can then be deployed at a larger scope. So far the advancement in this direction includes release of 3GPP Rel. 9 & 10 Adaptive HTTP Streaming which are following the MPEG DASH standardization.

2.3 DASH DESIGN

Perceiving DASH as a protocol or a system specification isn’t correct. DASH is more like a specification that provides format for better quality and efficient streaming. In order to design DASH, a lot of choices are on the table that makes its very adaptable and flexible with the existing technologies. DASH makes use of the existing file formats like container format for MPEG-4 and
MPEG-TS as well as existing codecs. It also provides the option of selection of content depending upon the network statistics and device capabilities. Users are given more options in term of Live, ‘On demand’ and ‘time shifted viewing’ support. So the design of DASH varies according to need and thus making it common in all aspects.

2.3.1 MAIN ASPECTS OF DASH

The most remarkable feature associated with DASH is that intelligence is concentrated towards the client unlike with the previous streaming process where the network and the server had the privileges for that. This enables the client to make the decision for seamless switching by gauging the network state and system capacity. This makes it more efficient as the content is already available on the HTTP CDN’s which is pre existing web infrastructure present and moving the intelligence on the client will induce a higher user experience as per demand. The architecture in figure 2 shows the HTTP content distribution network consists of HTTP caches and origin server that holds variable encoded media data of the same content that is produced by the media generation engine. This content is intimated to the client via a media presentation description through which the client decides the suitable content version according to the optimal condition at hand. The interaction relies on HTTP as it provides several advantages for smooth running. For instance, the existing web infrastructure can be used by DASH using HTTP GET and partial GET methods. Due to its large scale implementation, HTTP offers DASH ubiquitous service for both wired and wireless networks.

![Figure 2 - DASH Architecture [2]](image)

2.3.2 DASH Data Model

For a more specific insight on the data model upon which DASH resides, let us consider the figure 3 that shows a modular DASH process. From the left hand side Media presentation on the HTTP server is actually the media content available as different encoded version segments. This variety of content is conveyed to client via the Media Presentation Description. This exchange of the metadata establishes a sense of signaling between the client and server that relates to fact that all available options and details are shared. The metadata exchange can take place via HTTP or another protocol as the standard doesn’t define the interaction [2].

The streaming client is segregated into three divisions. HTTP streaming control acts as the brain upon which the intelligence of the process relies. Accounting for all the network factors the decision for the access of appropriate segments is made that form the basic content of the media description accessed by HTTP-URL’s. The access method for the media content is HTTP 1.1 since it gives the option of partial GET service that is required in case a byte range of the segment is requested. HTTP access client handles the request/response process for the media access as directed by the
streaming control engine. The segments received are then passed on to the media player for decoding.

The current implementation of the data model is done via reference software (not publically available) that follows the given schematic.

![Figure 3 - DASH Data Model [2]](image)

### 2.4 DASH Media Presentation description (MPD)

Media presentation description as stated in the data model serves as initial signaling establishment between the client and the server. The media presentation description presents the content type available at the server and its construction also specifies the intent of the content generating party. The MPD can be tailored in a way as to serve specific service. It provides the client with all the meta data needed to make an informed decision about the content, its type, codec, bit rate etc [5]. The most distinguishing factor of DASH is the MPD that has been standardized in contrast to the proprietary manifest files used by Apple Live streaming and adobe streaming. The structure of the MPD can be broken down into several elements each entailing attributes serving a specific purpose for Dynamic Streaming. Some of the major elements under focus are discussed further. [4]

- Media presentation description consist of a series of one or more periods
- Each period comprises of one or more groups.
- Each group contains one or more representations.
- Representation in turn consists of one or more segments. Segments are the data units that contain the media data or metadata to decode and present the media content.

#### 2.4.1 Media Presentation Description Element

This forms the root element of the MPD that in turn contains all the other major elements. MPD element provides the streaming client to choose representations. The attributes associated with this element provides the basic information about the content that enables the client to determine the initial parameters of the content. In light of the ‘On Demand’ and ‘Live’ services each attribute has its own importance and presence of those attributes define the profile of the media presentation.
2.4.2 Period Element

Each MPD element consists of one or more periods. Period boundary depicts a certain amount of information like a location of the content at a particular server, encoding parameters etc. Each period has a start time that is relative to start time of Media presentation. The significance of the period element varies from the ‘ON Demand’ to ‘Live’ profile where the period has more of an active role. In period as well the associated attributes describe significant aspects of the media content.

2.4.3 Groups

Groups are regarded as container for representations. Each group may consist of one or more representations associated with a particular set of attributes. Similarly inside a period, another group will contain representation with a distinct set of parameters. If no groups are defined then the representations are directly included inside the period and are assigned a group 0 by default. Generally another element ‘CommonAttributeElements’ defines some parameters that are common amongst groups and representation that they hold. Groups can be formed based on codecs, resolution, bandwidth, ranges for width, height, frame rate etc.

2.4.4 Representation

Representations provides with the alternative options for the media content that the group/period contain. The media choice differs in terms of bitrates, language, codecs. Each representation comprises of media segments that are presented in a media encoded content format for audio/video. These segments inside a representation together constitute the whole movie/media content. The representation generally contains at most an initialization segment followed by media segments. The initialization segment is present to provide information about the accessing the media data as well as for the media decoder to initiate decoding. The initialization segment doesn’t contain any media content. The media segment on the other end is the data unit that is accessed by HTTP-Url with a possibility of byte range use to access a particular part of the segment.

For DASH the media segment provides a wide range of functionality for the client so that it can optimally use the content. For this purpose certain characteristics may be associated with the media segments to fulfill certain aspects like [6]

- Media segments are assigned HTTP Url’s possibly restricted by byte ranges.
- A start time is explicitly or implicitly assigned relative to start of the representation. This enables to download the media segments in the proper play out mode or after seeking.
- Media segments should provide information like timing and byte index for accessing the segment in the media presentation.
- Media segments may provide random access information (RAP) i.e the presence and location of random access points [2].
- For seamless switching between the multiple representation, the media segments contain information to time accurately present each media component without accessing any previous media component in the representation provided that the segment starts with a RAP.
- Information may be included to access sub segments within the segment randomly by using HTTP partial GET method.
2.5 MPD Structure

Media presentation description is actually a well-formed XML document that is in line with the XML schema defined by the 3GPP Adaptive HTTP Streaming specifications. The MPD restructuring depends on the profile that is being followed. For ‘live’ services the MPD may be updated at particular instance in time that will require the MPD to restructure and update as the media content is being generated live and the access media is redefined with each update [7]. These update should be in line with the previous instance of the MPD. For ‘ON Demand’ services an update may occur as well but will not be as much common as on demand content is already available. An elaborate discussion on XML schema will be carried out in later section of MPD conformance testing.

2.5.1 DASH Components

2.5.2 Segment Format

The DASH specifies segment format according to 3GPP file format [6]. The format specified can be classified as:
- Initialization Segment with a sequence of media segments or a single media segment
- Self initialization media segment

At present, the combination of initialization segment and media segments are in practice in the current testing and implementation of DASH.

2.5.2.1 Initialization Segment

Initialization segment specifies the metadata that is needed to initialize the media content [5]. The format of initialization segment consists of a certain ISO box format that comprise of file type box ‘ftyp’, movie box ‘moov’ box containing no samples and optionally a progressive download information ‘pdin’ box. The details about the importance and structure of box will be provided in the file format conformance checking section later on.
2.5.2.2 Media segment

Media segment usually contain an optionally segment type box ‘styp’ used for file branding and may also contain a segment index box ‘sidx’ box. ‘Sidx’ box contains specific information about timing of tracks contained, byte offsets of the movie fragments and random access points. This information comes in handy for streaming client to gauge information about the content and making decision for requests to be passed to the HTTP access client. The ‘styp’ and ‘sidx’ is followed by one or more movie fragment boxes followed by the media data ‘mdat’ box. Additional, track fragment adjustment ‘tfad’ may also be specified to provide time alignment and random access to start of the segment. These distinct features make the DASH format a bit distinct from the standard ISO base media file format.

2.6 Segment Indexing

Segment indexing forms a very important aspect of dynamic adaptive streaming as it gives information about how to access sub segment structures from within the segment through byte ranges for random access and seamless switching purposes. The segment index box is present at the beginning of the segments documenting information about the sub segments that follows. SIDX placed in the start provides certain features that can be used by the streaming client. Particularly the information the index indicates is:

- Reference track ID of the media i.e the video track / audio track
- The timescale that is used for the reference tracks as the timing information is given in terms of specific timescale.
- Earliest presentation time provides with the Presentation of the reference track.
- First byte offset
- Reference count i.e the number of sub segments contained in the segment
- For each of the sub segments the ‘sidx’ provides with the following information:
  - Reference Size (Size of the sub segment – bytes)
  - Sub segment duration ( in timescale)
  - Random Access point information

The information above can be accessed by only downloading a small portion of the segment initially by partial GET request. The Segment index box thus obtained can be used to seek information for quick navigation to a later part in the segment. Timing information in the ‘SIDX’ box is crucial to ensure a synchronous play out of media within and across representation when switching takes place. Random access point’s location is also extracted from the segment index box information without the need to download any prior media data.

A simple depiction of the segment index box is provided below. The box label S1 is the ‘SIDX’ that provides timing, byte offset and random access point information about the movie fragments F1, F2 and F3 that follow. Although it can be a case that a segment comprised of may segments that may cause the composition of a large ‘sidx’ box. For this the information can be provided in a nested manner, like more than one sidx box may be present in the segment that can reference the other. Subject to this condition a variety nested ‘sidx’ structures are present namely hierarchical, daisy chain and hybrids. The details about the structure are not necessary as single sidx box is presently implemented and will be our area of focus as well.
2.8 Adaptive Switching

Adaptive switching makes DASH in particular, extremely viable for enhanced user experience. The switching process can be viewed as gradual tuning of the media streaming. Depending on the adaptive switching algorithm the client is already aware of the type of media content prepared and available at the server via the MPD. Parameters such as bandwidth, bitrates, duration, in the MPD aid the client to compare statistics that are available at that moment. That is, it may be the case that initially the client starts off with the lowest possible representation media content having a low bit rate and bandwidth associated to it. Soon after accessing a couple of segments an evaluation is made at the streaming client to determine whether or not there is capacity to switch to a better and higher statistical representation content. If the network condition allows, a switch takes place which should be completely transparent to the user. Same would be case in case network condition deteriorated and a switch down to a lower statistical representation is to take place.

For this switching to be seamless it is necessary to ensure that the switching points are aligned across the streams or representations so that there is no apparent disruption when the switch takes place. The alignment for DASH is not only confined to the segment level boundaries as in the case of Apple live streaming but the switching is even possible on the sub segment level by accessing the appropriate RAP. This granularity in switching makes the process more efficient and switch point does have to wait till the end of segments [2]. The process of adaptive streaming can be streamlined for ‘On demand ‘and ‘Live’ services separately. Each of the service has distinguished features associated with segment sizes (keeping in view the pros and cons), byte range request usage, and separate audio and video files, MPD updates.

The 3 GPP DASH specifications in comparison with other streaming services have the ability to optimize the process at the content generation side as well. Like for instance by generating content of different segment size and duration the HTTP request handling is enhanced by minimization of service time. In comparison to the other proprietary streaming, 3GPP HTTP streaming standard is more robust and reliable for media streaming. A general comparison of the streaming service is listed in table 1.

Table 1 - Comparison of Streaming Services [3]

<table>
<thead>
<tr>
<th>Stats</th>
<th>Adobe Dynamic HTTP Streaming</th>
<th>Apple HTTP Live Streaming</th>
<th>Microsoft Smooth Streaming</th>
<th>3GPP- DASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming modes</td>
<td>On demand / Live</td>
<td>On demand / Live</td>
<td>On demand / Live</td>
<td>On demand / Live</td>
</tr>
<tr>
<td>Adaptive bitrates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Segmenter Software</td>
<td>Free</td>
<td>Included in OS X</td>
<td>$199, includes encoder</td>
<td>So far not publically available</td>
</tr>
<tr>
<td>Media Container</td>
<td>MPEG 4 – Part 14, FLV</td>
<td>MPEG-2 TS</td>
<td>MPEG 4 – Part 14</td>
<td>3GPP, MPEG4</td>
</tr>
<tr>
<td>Video codecs</td>
<td>H.264</td>
<td>H.264 Baseline Level</td>
<td>Agnostic</td>
<td>Agnostic</td>
</tr>
<tr>
<td>Default fragment length</td>
<td>4 seconds</td>
<td>10 seconds</td>
<td>2 seconds</td>
<td>flexible</td>
</tr>
<tr>
<td>Server file type</td>
<td>Contiguous</td>
<td>Segmented</td>
<td>Contiguous</td>
<td>both</td>
</tr>
<tr>
<td>Maximum Bitrate</td>
<td>Unlimited</td>
<td>1.6 Mbps</td>
<td>Unlimited</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Caching Support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>End-to-End Latency</td>
<td>6sec</td>
<td>30sec</td>
<td>&gt;1.5sec</td>
<td>flexible</td>
</tr>
</tbody>
</table>
CHAPTER 3  DASH MPD CONFORMANCE PROCEDURES

3.1 Conformance Procedure of Dynamic Adaptive Streaming

Since a number of HTTP streaming protocols are currently in use each of them has propriety features associated with it. Apple live streaming is a predecessor of DASH. It is relatively simple and limited streaming protocol. Then streaming solution by Microsoft (smooth streaming) and Netflix have their own version of adaptive streaming clients. So with all these Adaptive streaming solution around, the need for a one concrete standard was essential to merge the features of the existing technology and bring in more features that can optimize the performance of streaming keeping in view the shortcomings associated with the existing streaming solution. There is need for a standardized model that governs the format of interaction between the server and client, features essential for the client to make an informed decision, the level of intelligence inclination towards the client side, the type of content to be used for streaming and many others. These aspects can be regarded as the building blocks for the entire process as the working of the streaming process heavily relies on such factors.

With the advent of Dynamic Adaptive Streaming as a standard, the features that should be associated with a streaming client and server need to be in compliance with the implementation. It is important that the media content that the server holds abides by the type of content that is approved by the standardization. Also, the information in the form of Media Presentation description that is provided to the client must have certain attributes that enables the client the leverage to impart intelligence in the process. The rules and regulation specified by the standard are focused in making the process simpler and more effective.

Since the standardization process is relatively recent, there is a need to see how the standard reflects itself in term of a practical model implementation. Mainly the additions required or any unnecessary aspects that don’t hold primary importance, the working performance, the media content type, aspects of MPD interaction need to be checked so that the new evolving standard can be updated and reformed accordingly. This effort to check the compliance of DASH in a working environment provides a particular dimension to see what is necessary for a streaming process to be called DASH compliant. It is also important to distinguish any streaming process from DASH so that the standard can be set as reference to all other processes. The efforts in this direction will make DASH more efficient and will help in developing the standard further for more expansion.

Particularly in the area of compliance procedure, the focus is mainly on the following aspects presently

- Media Presentation Description Compliance is necessary as the manifest file should incorporate DASH specification. As many versions of manifest files are in practice by streaming processes but it is essential for DASH that the interaction between the client and the server takes place according to rules and regulation of DASH.
- File Format Conformance ensures that the media content that is specified by the MPD is in accordance with the DASH features mentioned in the MPD as well as the standard itself. The file format level conformance ensures that the content when used by the client will provide it with the flexibility that DASH offers.
- While implementing checks for DASH features, contributions in the DASH standard can be suggested to clarify concepts and relate to the theoretical features in term of practical implementation.
3.2 Media Presentation Description Compliance

In the area of compliance testing, procedures at the MPD level present an important aspect. A
generic compliance procedure at this level is provided in figure 5. This compliance procedure is
presented by Markus Waltl and Christian Timmerer from University of Klagenfurt [10] which is
divided in three distinct stages of testing. Before exploring each stage is greater details a few of the
basic terminologies that are focused ahead are discussed to have a better understanding of the
subject at hand.

3.2.1 MPD – AS XML Document

MPD is generally presented in the form of an XML document. XML (Extensible Markup Language) is
designed to store and transport data for web applications. The reason for MPD as being a XML file is
associated with XML properties like simplified data sharing in plain text, data transportation among
non compatible systems and the option to introduce user defined tags [11]. To understand XML file
format consider an example MPD i.e an XML file format in figure 6.

It can be seen from the figure that an XML document has a tree like structure with a root node
(element) and child node (elements). The syntax of a XML dictates that it can have user defined tags
like <object> and each tag once opened must have a closing tag like </object>. Then XML
incorporates the concept of elements and attributes where each element can be linked as a node in
the XML tree and everything between the opening and closing tags of the element constitutes it as a
whole. Attributes add features to the elements providing additional information. Attributes can be
distinguished by text in quotes "attribute "or 'attribute'.

Applying the above discussion to the MPD presented, the root element of the MPD can be
designated as MPD element, followed by period, representation and segment info elements. Each of
the nested elements has its own associated attributes shown in quotes. The end of the XML
document is marked by the closing tags of each of the element opened.
XLink (XML linking language) is used to create hyperlinks inside an XML file. Xlink provides the function to point to a reference or to a particular part in a XML file. Any element in the XML structure can behave as a link referencing to parts outside the existing files as well. Xlink has certain attributes that govern features associated with the hyperlinks that are used. For that matter, while using XLink, its namespace is declared first. Namespaces in XML documents define the association of elements and its attributes to a particular namespace so that they can be unique. Usually the name spacing convention is xmlns:prefix="URI". Notable attributes of XLink that are used in DASH standardization are:

- **Xlink : type** ="simple" (All links are simple links in DASH)
- **Xlink : href** ="URL" (The URL to links shall exclusively be http-URLs)
- **Xlink : show** = “embed” (Where to open the link. ‘Embed’ means on the same page)
- **Xlink : actuate** ="On Load” (Defines when the linked resource is read and shown. By default the value is set to”onRequest”)

In DASH specification any element can be referenced by Xlink to remote DASH by specifying the appropriate URI [9]. This referencing should be according to the rules specified so as to set boundaries for a valid referenced element.

### 3.2.3 XML Validation via XML Schema

XML validation is a process that ensures that the XML is well formed and is according to a correct syntax. Generally in terms of XML, correct syntax is primarily concerned with the presence of root element, proper opening and closing of tags, elements and attributes are nested properly. To check whether an XML file conforms to a valid pattern an XML Schema is declared that describes the structure and properties that a valid XML document should have. A Schema can be considered as reference document that specifies the name, type, minimum and maximum occurrence and other
properties that should be associated with an element or an attribute that is in the XML. The schema thus describes the allowed content, data types for elements and attributes and the overall correctness of the XML document.

To have a better understanding of the Schema, the structure of the schema is described a bit in detail. It consists of a “Schema” element as the root element. As indicated in the extract from the DASH Schema specified by ISO-IEC FCD 23001-6 in figure 7 the elements and the data types used in the schema are taken from the name space “http://www.w3.org/2001/XMLSchema” [13] and are to be prefixed with ‘xs’. As can be seen further that the element MPD and it sub elements and attributes are prefixed with ‘xs’. The extract of the DASH schema provides an idea of how the schema document looks like. So, when reference with a XML file the properties and aspects mentioned in the schema like ‘type’, ‘name’ etc are matched so that a correct and valid XML document can be declared. In summary the Schema is an abstract of the metadata (Schema components).

The schema can be referenced from within the XML document as can be seen in figure 6 where ‘xs:schemaLocation="urn:mpeg:mpegB:schema:DASH:MPD:2011 dash.xsd” provides an instance to the schema location via a URN that can be referenced for schema validation. Hence, the XML instance is compared against the schema location.

### 3.2.4 Schematron Validation

Previously discussed schema validation was focused more on the constraints of the structure and the content of the XML. The concern with schema validation lies mainly on the grammatical rules of the document and the structural rules are defined in the XSD schema file to assert the wellformedness. Schematron validation on the other hand is rule based language that makes assertion on the presence and absence of paths in XML document [14]. Schematron schema apply

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
    attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"/>

<xs:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
 <!-- MPD: main element -->
<xs:element name="MPD" type="MPDtype"/>

<!-- MPD Type -->
<xs:complexType name="MPDtype">
    <xs:sequence>
    <xs:element name="ProgramInformation" type="ProgramInformationType"
        minOccurs="0"/>
    <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
    <xs:element name="BaseURL" type="BaseURLType" minOccurs="0"
        maxOccurs="unbounded"/>
    <xs:any namespace="#other" processContents="lax" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

Figure 7 - DASH Schema Sample
rules on a specific content in the XML document and checks if it abides by it, by generating a pass or fail message to display the results. Advantage of Schematron validation is that constraint written in plain English can be referred to as Schematron rules. Then, after defining the rules, assertions are applied to specific context to validate rules [15]. In contrast to simple schema the Schematron complements the validation process to provide checking of constraint at a different conceptual level. It along with the grammar also validates user defined rules.

Schematron is build upon a foundation of Xpath and XSLT. Xpath provides with the path expressions used in Schematron rules and XSLT processors are used to process Schematron Rules.

Xpath is a query Language used for selecting nodes in a XML Document. It is generally used to compute Values like strings, number, Boolean values from XML documents. Xpath works by navigating the XML tree structure and selecting nodes by specific criteria [17].

As for XSLT (Extensible Style sheet language Transformation) it is a declarative XML language that is used primarily for transformation of XML documents to different types or conversion between different XML Schemas. Usually the output for XSLT is in the form of plain text, HTML or XML file. The principle of working is that the template rules in style sheet define the handling of the nodes that match a particular X-path pattern [16]. It does so by building a source tree i.e from input XML starting from the tree root node and then finds the templates that best match that node so that its content can be evaluated. XSLT relies on XPath for identifying the sub sets and nodes in the XML file and performing the required calculations. An example demonstration of the process can be inferred from figure 8 where an XML document along with a XSLT code specifying rules templates are fed to an XSLT processors that produces the output in the desired form.

![Figure 8 - XSLT Processing](image)

The Schematron has a four layer hierarchy that defines the scope of the rule based validation [15]:

- Phases
- Patterns
- Rules
- Assertion

To explain it further, consider an extract from a DASH Schematron schema used in the Conformance process. It’s important that all namespaces that are used in the XML instance document are declared in Schematron as well. These name spaces element ‘ns’ are aided by two attributes i.e ‘prefix’ and the ‘uri’. Starting with schema as the root element the first level is ‘pattern’ having ‘name’ as an attribute [14]. This ‘name’ is displayed every time a Schemtron check is run. ‘Pattern’ are used to group together different rules. In the example from figure 9, MPD Element forms a pattern that will cater for all the rules pertaining to the MPD element in media presentation description. ‘Rule’ comes next, nested as children of the pattern since a pattern is a collection of ‘rules’. ‘Rule’ element has an attribute ‘context’ that defines the scope of the rule. This context should be in line with an Xpath pattern that is used to select a node from the document. The context “dash:MPD” specifies the node “MPD” in the namespace specified by “dash” at the start. Lastly the ‘assert’ element is the main
check that is applied to a specific context of the XML document. Having an attribute ‘test’, the condition is specified that’s will be applied against the XML. From the example the condition in ‘test’ states “If the attribute ‘type’ is “Live” and the attribute “availabilityStartTime” is not present then generate an error”.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema
xmlns="http://purl.oclc.org/dsdl/schematron"
xmlns:dash="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
queryBinding="xslt"
schemaVersion="ISO19757-3">
  <ns prefix="dash" uri="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"/>
  <ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>
  <title>Schema for validating DASH descriptions</title>
  <pattern name="MPD element">
    <!-- R1.*: Check the conformance of MPD -->
    <rule context="dash:MPD">
      <!-- R1.0 -->
      <assert test="if (@type = 'Live' and not(@availabilityStartTime)) then false() else true()">If MPD is of type "Live" availabilityStartTime shall be defined.</assert>
    </rule>
  </pattern>
</schema>
```

**Figure 9 - DASH Schematron Schema Sample**

The generic process of the Schematron validation is depicted in the figure 10. It comprises of the following

**First Stage:**
- Schematron Schema that has all the rules in the hierarchal order as stated earlier (Schematron.xsd)
- “Schematron_Basic.xsl” is a type of style sheet which will transform the schema and provide you with an output in text form. Other style sheet may used if a different output is desired of the first stage.

**Second Stage:**
- After obtaining a validating XSLT the next step is to process it with the XML document that in case of DASH is the MPD.
- The XSLT processor produces the desired result.

![Figure 10 - Schematron Processing Stages [15]](image-url)
3.3 DASH Media presentation Description Conformance Implementation

Now that the basics concepts have been clarified, conformance procedure in light of DASH can be discussed. The conformance is split in the following three stages [10]:

- **Xlink Resolver**: A MPD is firstly checked for resolved Xlink attributes. If in case of error, the error message is intimated else the resolved MPD is produced.
- **XML Validation**: The Xlink resolved MPD is provided along with the MPD schema to check for the wellformedness of the XML by checking for the grammatical validation. Again if any error occurs that should be displayed or else the validated XML document is produced for further processing.
- **Schematron Validation**: Last stage of the process involves the rules checking as specified by the DASH specification to check if the MPD abides by the standard requirements. Here, the XML validated and Xlink resolved MPD is checked against the Schematron schema to provide a final validated MPD in case not errors occur.

For each stage different approach and tools are available to attain the conformance goals desired. But here the concept by Markus Waltl and Christian Timmerer of University of Klagenfurt [10] is kept in focus to discuss the details of the implementation of this conformance procedure.

3.3.1 XLINK Resolver

As per the standard ISO/IEC-23001-6 the following rules for Xlink need to be checked: [4]

```
"The following rules apply to the processing of URI references within @xlink:href:

- **URI references to remote DASH elements that cannot be resolved shall be treated as invalid references.**
- **URI references to remote DASH elements that are inappropriate targets for the given reference shall be treated as invalid references (see list below for the appropriate targets).**
- **URI references that directly or indirectly reference themselves are treated as invalid circular references.**

The remote DASH elements referenced from within an MPD (referred to as appropriate targets) shall be embedded into the MPD by applying the following rules:

1. **Attributes shall be added to the element of the MPD that contains @xlink:href merged with existing attributes. If the same attributes are present in both MPD and remote DASH element, the attribute values should be the same. If they are not identical, then the value of the attribute of the MPD takes precedence over the value of the attribute in the remote DASH element.**
2. **The remote DASH element referenced by the @xlink:href shall conform to the type definition of the element in the MPD that contains @xlink:href.**
3. **All XLINK attributes shall be removed after dereferencing is completed.**

Only a single element shall be included in a remote DASH element."
```
In light of conformance the rules regarding unresolved references, inappropriate targets and circular reference form solid conformance test cases to be checked in a DASH MPD. It should be ensured that the Xlink in the MPD abides by these rules so that the client is provided with references that are valid and it doesn’t cause any ambiguity on the client’s end while accessing the segment for streaming. The test conditions for Xlink are explained as follows.

### 3.3.1.1 Check for Circular Referencing

URI are checked if they reference directly or indirectly to themselves which should cause an error. This case is illustrated by an example comprises of three XML files [9].

Consider a sample MPD “circular_ref.xml” shown in figure 11. In the Period element “xlink:href” points to a URI that in turn relates to a file “ex_include.xml” which means that the URI references a remote DASH element at this point. So the first period element in the MPD is referenced while the other period has all its sub elements and attributes present. So now in order to resolve the XLink reference the file “ex_include.xml” is checked which is depicted in figure 12.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
     xsi:schemaLocation="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011 dash.xsd"
     minBufferTime="PT10.00S" type="Live"
     availabilityStartTime="2001-12-17T09:40:57Z">
  <Period xlink:href="http://www.DASH-Example.com/files/ex_include.xml"
         xlink:actuate="onRequest" start="PT06H"/>
  <Period start="PT08H" id="10" segmentAlignmentFlag="true">
    <SegmentInfoDefault duration="PT10S" sourceURLTemplatePeriod="http://www.example.com/Period-06/rep-$RepresentationID$/seg-$Index$.3gp" />
    <Representation id="QVGA-LQ" mimeType="video/3gpp; codecs="avc1.42E00C, mp4a.40.2"
                    bandwidth="192000" width="320" height="240">
      <SegmentInfo>
        <InitialisationSegmentURL sourceURL="http://www.example.com/rep-QVGA-LQ/seg-init.3gp" />
      </SegmentInfo>
      </Representation>
    <Representation id="QVGA-HQ" mimeType="video/3gpp; codecs="avc1.42E00C, mp4a.40.2"
                    bandwidth="384000" width="320" height="240">
      <SegmentInfo>
        <InitialisationSegmentURL sourceURL="http://www.example.com/rep-QVGA-HQ/seg-init.3gp" />
      </SegmentInfo>
      </Representation>
  </Period>
</MPD>
```

**Figure 11 – Circular Referencing Example - "circular_ref.xml"**

As the remote DASH “Period” element is seen to have two representations inside, the first two representations are independent of any further referencing but the third representation in turn hold a "xlink:href" to a remote representation element. So the remote DASH element is referenced to “ex-circular.xml”. Now this reference is to be checked.
The error simply catches the eye as the last reference is again reference to a URI that is itself. That means that will be a loop when this representation is again referenced and will caused a circular referencing problem. This needs to be checked every time a "xlink:href" is present in an MPD.

This Xlink resolver is integrated in the ANT Java script tool developed for MPD conformance for DASH [10]. It will produce and exceptions in case an error occurs:

```
XLinkException: Circular referencing detected!
```

### 3.3.1.2 Check Invalid Protocol Usage

As for now only HTTP references are allowed for DASH MPD. Since all the dynamic adaptive streaming works on HTTP 1.1 so the references are also supposed to be HTTP based. So in case the MPD is of type as shown in figure 14, the conformance check should produce an exception since any other protocol except for HTTP doesn’t comply with DASH standardization at the moment.
3.3.1.3 Check Wrong Element Referencing

An important facet to look for is that even if the referencing is valid it doesn’t mean that it is correct as well. The referenced element should match the remote element. Like in a case that if a Period element is referenced from the MPD to external XML file then it should be made sure that the external XML file contains a period element and itself is a valid Period element. An error scenario of this kind is shown in figure 15 where a period element is referenced but the remote element is a representation element that forms an invalid reference as well.

Figure 14 - Invalid Protocol Use in MPD

Figure 15 - Wrong Reference - main MPD (wrongelement.xml)

Here the MPD shown in “wrongelement.xml” provides a reference of period element to another xml file “ref-wrongelement.xml”. Now the reference is supposed to have a period element but in turn it is a representation element that is present in the remote file. So this also forms a case of invalid Xlink referencing via MPD.

Figure 16 - Wrong Reference - Referenced Element (ref-wrongelement.xml)
3.3.2 XML Schema Validation

After resolving for the XLink attributes, the MPD under consideration is passed to the next stage of conformance i.e. the schema validation. The XML Schema defined in the ISO-IEC 23001-6 [4] provides the baseline for this stage of the conformance test procedures. The MPD generated for the DASH content must be in line with the MPD Schema. This process governs the checks for grammatical rules laid out by the abstract Schema metadata. The DASH Schema in full is present in Appendix B.

There are many ways to implement this stage of validation. Online resources available to validate generally simple schema files against XML documents are:

1. Xml Validation  http://www.xmlvalidation.com/

Stand alone validation can also be done by using Java Xml Validation package available at:

- http://download.oracle.com/javase/1.5.0/docs/api/javax/xml/validation/package-summary.html

Another convenient resource for this purpose is xmllint package in libxml2 library on Linux platform. Generally simple Schema Documents can be validated via the command line tool by:

- $ xmllint --schema MPDSchema.xsd content-mpd.xml

The method under consideration here is the Java XML package. This Java library provides methods to deal with the schema document and the xml to be checked. The ANT java script used for conformance also incorporates a Java Xml validator that will verify the grammatical validity of the xml document.

Consider an extract from a sample XML MPD in figure 17. In order to verify that the Period element specified here is in line with schema for DASH, the Schema extract for Period element is shown in comparison as well in figure 18.

Looking at the period element from the MPD a few of the attributes of the period element are present. Now let’s do a comparison with the extract from the DASH schema of the Period element there to see if the XML document conforms to the specification. Starting with the default value of the “xlink:actuate” is “onRequest” that is present in the MPD. Then the “start” attribute has value “PT08H” that’s coincides with the type “duration” as mentioned in the schema. The “id” attribute has a value “10” that is of type “string” and is also valid in this case. Lastly the segment alignment flag is of type “Boolean” in the schema and has a value “true” in the MPD. Apart from the types, the names of the attributes also match the names specified in the schema. In case that an attribute name is not specified in the schema or an existing attribute that doesn’t have a valid type then an error notification is generated by the Java XML validator.
Rule based validation makes up the last stage of this compliance process. Presence and absence of patterns in the MPD can be made by defining assertion rules at different hierarchical levels. A schematic of the rule based validation process for schematron software implementation is shown in figure 19.

![Figure 19 - Schematron Validation Schematic Full](image)

### 3.3.3 Schematron Validation

Rule based validation makes up the last stage of this compliance process. Presence and absence of patterns in the MPD can be made by defining assertion rules at different hierarchical levels. A schematic of the rule based validation process for schematron software implementation is shown in figure 19.

![Figure 19 - Schematron Validation Schematic Full](image)
The implementation of this process is according to skeleton implementation at [www.schematron.com][14] which being an open source is quite widespread. The implementation uses XSLT1 or XSLT2 to provide a variety of outputs. The entire procedure can be split in a four stage XSLT processing described as follows:

1. Resolve/Process Inclusions (via iso_dsd_include.xsl)
2. Resolve Abstractions (Patterns)(via iso_dsd_expand.xsl)
3. Compiling Schema - Produce the Validating XSLT StyleSheet to be applied to XML Description - MPD.xml in this case. (via iso_svrl_for_xslt2.xsl)
4. Validation Output by processing XSLT stylesheet (Validation XSLT) with MPD.xml

The validation rules that constitute the schematron are derived from the ISO-IEC 23001-6[4]. These rules in accordance with dash content authenticate the processes like MPD updates, bit stream switching, segment alignment, time lines etc. Schematron makes it possible to govern the specific combinations of elements and attributes so that a valid content description can be provided to the client that hold all the intelligence in case of DASH to make decisions based on the information provided.

The main utilities required for this process on Linux platform are:

- Java based XSLT Processor : Saxon 9 (saxon9.jar)
- ISO_Schematron distribution (XSLT2 is being used currently).
- ISO Schematron SVRL (Schematron Validation Report Language) also used in this version.

The main aspect of this stage is the extraction of valid rules from the standardization to be applied to every media presentation description. The standard does specify certain requirement pertaining to the organization of elements and attributes in combination as well as separately. The task is mainly to formulate Schematron rules and form a Schematron Schema like the Dash Schema for XML validation that is a universally accepted rules book for DASH MPD content structure. In this regard, work was done by scrutinizing the details of MPD elements and attributes to extract conditions that need to be satisfied for valid DASH description. So initially all possible rules and conditions are noted down and then later are sorted according to its significance level. A collection of the major rules is provided in table 2.

### Table 2 - DASH Validation Rule Collection

<table>
<thead>
<tr>
<th>COLLECTION OF ALL VALIDATION RULES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key: “@” signifies the attribute of the element under consideration</td>
</tr>
<tr>
<td>RULES FOR MPD ELEMENT:</td>
</tr>
<tr>
<td>- Only a single MPD element should exist. [Sect 5.3.3 - ISO/IEC FCD 23001-6]</td>
</tr>
<tr>
<td>- If MPD is of type &quot;Live&quot; availabilityStartTime shall be defined. [Sect 5.4.1.2 - ISO/IEC FCD 23001-6]</td>
</tr>
<tr>
<td>- If MPD is of type &quot;OnDemand&quot; timeShiftBufferDepth shall not be defined. [Sec 5.4.1.2 - ISO/IEC FCD 23001-6]</td>
</tr>
<tr>
<td>- Timeline of MPD can be checked by concatenation of timeline of each period present i.e @mediaPresentationDuration</td>
</tr>
</tbody>
</table>

---

[www.schematron.com]: https://www.schematron.com
MPD basic Profile for ‘OnDemand’ requires:

- Single period per MP
- @type = ‘On Demand’
- @mediaPresentationDuration shall be present
- @minBufferTime shall be present

**RULES FOR PERIOD ELEMENT:**

- For MPD type ‘Live’, @start should be present. [Sec 5.4.2.1 - ISO/IEC FCD 23001-6]
- For MPD type ‘OnDemand’, @start if present = 0 or by default is zero. [Sec 5.4.2.1 - ISO/IEC FCD 23001-6]
- In an Update MPD the Period ID should not change. [Sec 5.4.2.1 - ISO/IEC FCD 23001-6]
- If profile is ‘On Demand’ then if @minBufferTime is not defined at MPD level, Period@minBufferTime should be present. [Sec 5.4.2.1 - ISO/IEC FCD 23001-6]
- Media Presentation time should conform to sum of last period start time and Period duration.
- MPD@mediaPresentationDuration == Period@PeriodStartTime(last period)+@duration
- BitstreamSwitchingFlag shall not be set to true if segmentAlignmentFlag is set to false. [Sect 5.4.2.2 - ISO/IEC FCD 23001-6]

**RULES FOR REPRESENTATION ELEMENT:**

- Group and Representation attributes should abide by the following limits: [Sec 5.4.3.3 - ISO/IEC FCD 23001-6]
- Group@minBW < Representation@BW < Group@maxBW
- Group@minWidth < Representation@Width < Group@maxWidth
- Group@minHeight < Representation@Height < Group@maxHeight
- Group@minFrameRate < Representation@FrameRate < Group@maxFrameRate
- Representation ID should have valid characters as specified by RFC 1738 for URL construction. [Sec 5.4.3.4.2 - ISO/IEC FCD 23001-6]
- Representation@mimetype or Group@mimetype is ‘video/3gp’ for Initialization segment
  - [Sec 12.4.2 – 3GPP TS 26.234 V9.2.0 (2010-03)]
- Mime type for media segments accessed through URLs without byte ranges is ‘video/vnd.3gp.segment’ otherwise with byte ranges it is ‘video/3gpp’. [Sec 12.4.2 – 3GPP TS 26.234 V9.2.0 (2010-03)]
- Each Representation has at most one SegmentInfo Element.

**RULES FOR SEGMENT ELEMENT:**

- If no @duration is present and no Segment TimeLine then Representation will contain only 1 media segment at start time 0. [Sec 5.4.4.3.3 - ISO/IEC FCD 23001-6 ]
- SegmentInfo shall have a duration attribute if Period/Group/Representation has no SegmentInfo@duration. [Sec 12.2.4.2 – 3GPP TS 26.234 V9.2.0 (2010-03)]
- If MPD is of type “OnDemand” StartIndex shall not be present or set to 1. [Sec 5.4.4.5]
- InitialisationSegmentURL or URLTemplate@sourceURL must be present for initialization segment.
  - [otherwise self initialization segments] [Sec 12.2.4.2 – 3GPP TS 26.234 V9.2.0 (2010-03)]
- Approximate start time of a media segment is (index-1) * @duration. [Sec 5.4.4.3.3-ISO/IEC FCD 23001-6]
- URLTemplate@sourceURL should have index set to 0 to obtain initialization segment.
- SegmentInfo shall not have a duration attribute if SegmentTimeLine is present. [Sec 5.4.4.5 - ISO/IEC FCD 23001-6]
- If sourceURL is defined Representation id shall not be defined. [Sec 5.4.4.5 - ISO/IEC FCD 23001-6]
- Each SegmentInfo element shall contain either a URLTemplate Element or one or more Uri Elements. [Sec 12.2.4.2 - 3GPP TS 26.234 V9.2.0 (2010-03)]
As a part of my work after compiling all possible rules for checking of media presentation description, now the Schematron schema was formed. For this the rules that are viable in the scope of Schematron checks are selected and accordingly assertion ‘rule’ attributes are formulated for the schema. The Schematron schema for DASH formulated for conformance procedures is shown in Appendix A. The rules, their description and explanation laid out by me as a primary task at this stage are provided in the form of table 3 to have a deeper understanding of the process.

The main area of focus at the moment is the “On Demand” profile for DASH since its implementation is currently in practice. Although LIVE profile holds equally high importance but presently “On demand” is discussed primarily in terms of the rules for schematron schema.

**Table 3 - DASH Schematron Conformance Rules**

<table>
<thead>
<tr>
<th>RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>If (@type=’On Demand’) then true else false</td>
<td>type attribute should be present and set to ON Demand</td>
</tr>
<tr>
<td><strong>Explanation:</strong></td>
<td>It is important to notify the client the type of content. This specifies which attributes can be used from within the MPD for decision process. For that matter “On Demand” specifies that there is set of information that is viable while making decision.</td>
</tr>
<tr>
<td>If (@mediaPresentationDuration ) then true else false</td>
<td>mediaPresentationDuration attribute should be present</td>
</tr>
<tr>
<td><strong>Explanation:</strong></td>
<td>Media presentation duration should be present in the MPD to intimate client the entire duration of the content like a movie/video. Media presentation duration is helpful in checking the time alignment of the content.</td>
</tr>
<tr>
<td>if (not(@minBufferTime) and not(child::dash:Period/@minBufferTime)) then false() else true()</td>
<td>If MPD has no attribute minBufferTime each Period element shall have this attribute.</td>
</tr>
<tr>
<td><strong>Explanation:</strong></td>
<td>Minimum buffer time is also vital for the client to know so that it can buffer that amount of video before starting the play back. This buffer time aides the uninterrupted video stream experience for the client. The buffer time also bridges the Streaming client engine and Media decoder for DASH client as the buffer in between form as bridge between the two processes aiding a continuous HTTP request process to smoothly work on.</td>
</tr>
<tr>
<td>if (@type = ‘OnDemand’ and @timeShiftBufferDepth) then false() else true()</td>
<td>If MPD is of type “OnDemand” timeShiftBufferDepth shall not be defined.</td>
</tr>
<tr>
<td><strong>Explanation:</strong></td>
<td>If the type of media content is “On demand” then the attribute “time shift buffer depth” seems irrelevant to be present as the attribute is specific to the LIVE services. Time shift buffer is applicable in LIVE content where MPD updates take place and new content is defined after some interval of time. This time shift buffer determines the amount of previous content that is available in a LIVE stream that has been buffered. Since for “On demand” all of the content is already available on the server, presence of this attribute doesn’t make sense.</td>
</tr>
<tr>
<td>Other Rules (Not specific to ‘ON Demand’)</td>
<td></td>
</tr>
<tr>
<td>if (@type = ‘Live’ and availabilityStartTime)</td>
<td>If MPD is of type “Live” availabilityStartTime</td>
</tr>
</tbody>
</table>
**Explanation:** For LIVE services availability start time forms great significance. As for the ON demand since the content is already available the start time is zero. But in case of “LIVE” type the stream may begin at a later time like in case of a football match, so the start time of stream is not predetermined and availability start time then defines further the start time for the representation and the media segments it contains.

<table>
<thead>
<tr>
<th><strong>If (not(@availabilityStartTime)) then false() else true()</strong></th>
<th><strong>shall be defined.</strong></th>
</tr>
</thead>
</table>

**Explanation:** Availability end time specifies the duration of the entire media content by presenting the finish time of the media. So if media end time is present then minimum update period should not be present since it provides an intimation that updates on the media content are expected that will affect the media end time as well. So if updates are expected the final media content cannot be determined beforehand.

<table>
<thead>
<tr>
<th><strong>If (@availabilityEndTime and @minimumUpdatePeriodMPD) then false else true</strong></th>
<th><strong>If @availabilityEndTime attribute is present then @minimumUpdatePeriodMPD shall not be present</strong></th>
</tr>
</thead>
</table>

**Explanation:** Similar logic applies in this case as well as media presentation duration if present define the entire length of the media content. But at the same time minimum update Period of an MPD should not be present as it intimates that an update to the MPD is possible which in turn point to a new media duration attribute. So these two conflicting attributes are not supposed to be present at the same time.

<table>
<thead>
<tr>
<th><strong>If @minimumUpdatePeriodMPD and @mediaPresentationDuration then false else true</strong></th>
<th><strong>If @minimumUpdatePeriodMPD is present then @mediaPresentationDuration shall not be present</strong></th>
</tr>
</thead>
</table>

**ON DEMAND Profile Specification for ‘Period’**

<table>
<thead>
<tr>
<th><strong>RULE</strong></th>
<th><strong>DESCRIPTION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If( @start = 0) then true else false OR if (not(@start) and not(preceding-sibling::dash:Period[1]/@duration)) then false() else true()</strong></td>
<td>For ‘ON Demand’ The @start attribute is zero if present or by default is zero if the start attribute is not defined for a Period the previous Period element shall contain the duration attribute.</td>
</tr>
</tbody>
</table>

**Explanation:** Since for the “On Demand” profile the media content is already available so the start time of period is supposed to be zero without any delay. By default this value is considered as zero in case the attribute is not mentioned and if not present in period element it should be made sure that the previous period element has “duration” attribute. This is important so that the start of period can be estimated by adding the duration of period to a start of zero i.e for the first period.

| **If (not(@start) and not(@duration)) then false else true** | **To prevent inefficient seeking without requiring unnecessary de-referencing of Period referenced by XLink, Period@start or Period@duration should be present** |

**Explanation:** This condition is the same as the above which states that if a period doesn’t have a start attribute it should have a duration attribute so that the start of the next period can be determined as the start of the very first period in the MPD is considered to be zero in case of “ON Demand” profile.
if (string(@bitStreamSwitchingFlag) = 'true' and string(@segmentAlignmentFlag) = 'false') then false() else true()

**BitstreamSwitchingFlag shall not be set to true if segmentAlignmentFlag is set to false.**

**Explanation:** Segment flag dictates that the segments across the representation don’t overlap in presentation times. Segment alignment ensures that the segment boundaries are concealed in a sense that it doesn’t violate the boundaries of other segment. This is very important when switching takes place from one representation to another inside a period. Bit Stream Switching emphasizes this point that when we concatenate an initialization segment with segment across different representation this results in a valid bit stream which is only possible when there is no overlapping amongst the segments intimated by the segment alignment flag.

### ON DEMAND Profile Specification for ‘Representation’

<table>
<thead>
<tr>
<th>RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>If (not(@id) then false else true OR if (('@id = preceding::dash: Representation/@id)) then false() else true())</td>
<td>@id attribute shall be present</td>
</tr>
</tbody>
</table>

**Explanation:** Every representation inside a period should have distinguished “id”. Firstly the presence of the “id” is mandatory so that while the client chooses a representation it is able to distinguish it precisely by its id tag. Secondly the “id” tag is distinct for every representation residing inside a period and no two representations have the same id. This consideration is important for the client as it chooses to switch up and down the representation order.

<table>
<thead>
<tr>
<th>RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>If(not(@Bandwidth) then false else true</td>
<td>@Bandwidth attribute shall be present</td>
</tr>
</tbody>
</table>

**Explanation:** Bandwidth at the representation level is also a mandatory part of DASH specification as the Bandwidth attribute is used for the adaptive switching purposes. The client upon gauging the network capacity looks at the bandwidth parameter of the representation to see which is the most appropriate representation to get segments from, which will provide the user with the best quality under the given condition. Bandwidth parameter is also used in forming adaptive algorithms for Dynamic streaming purposes at the clients end.

<table>
<thead>
<tr>
<th>RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>If (not(@mimeType) and not(parent::dash:Group/@mimeType) and not(child::dash:SubRepresentation/@MimeType) then false else true)</td>
<td>@mimeType attribute should be present at least at one level of hierarchy</td>
</tr>
</tbody>
</table>

**Explanation:** According to latest specification “mimetype” is a member of “CommonAttributeElements” which consist of attributes common to GROUP, REPRESENTATIONS and SUB-REPRESENTATIONS. Since for simplicity sake we only deal with representations at the moment by keeping GROUP =0, it is necessary to define the mime type which provides the mime type of the initialization segment or in absence of it the first media segment of the representation. In general the Mime type should be included in any of the three levels of “CommonAttributeElement” members

<table>
<thead>
<tr>
<th>RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>If (not(@codecs) and not(parent::dash:Group/@codecs) and</td>
<td>@codecs attribute should be present at least at one level of hierarchy</td>
</tr>
</tbody>
</table>

**Explanation:**
not(child::dash:SubRepresentation/@codecs)
then false else true

**Explanation:** A similar explanation to mimetype is applied to codecs attributes as well since it also belong to the group of “CommonAttributeElements” and the codecs parameters for the media content also influence the decision process at the streaming client engine. Depending of the type of codec mentioned at the representation level provides extra depth in categorizing the content and then based on the system capabilities and network capacity a more well informed decision making advantage can be achieved.

## ON DEMAND Profile Specification for ‘Group’

<table>
<thead>
<tr>
<th>RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>If( @subsegmentAlignment =’true’)then true else false</td>
<td>@ subsegmentAlignment flag should be set to true for ‘On demand’</td>
</tr>
</tbody>
</table>

**Explanation:** For “ON Demand” profile sub segment Alignment flag should be true. Sub segment alignment relates to the same principle as that of segment alignment with its scope and focus extended to sub segments. The sub segments boundaries should not overlap across the representation so that if a switch takes place at a sub segment boundary the presentation time should not overlap for a smooth transition to take place. This condition should hold for “ON demand” profile media segments according to the specifications.

## ON DEMAND Profile Specification for ‘SegmentInfo’

<table>
<thead>
<tr>
<th>RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not yet Implemented</td>
<td>If @sourceURL is relative and BaseURL is not established any level then generate Error</td>
</tr>
</tbody>
</table>

**Explanation:** The source URL for the segment can either be absolute or relative. In case its absolute the entire URL is independent and the segment can be accessed by the URL directly but in case of relative it relies in the base URL attribute. If base URL is not present in the MPD then the media segment access is not possible which will yield an error.

if (not(@duration) and not(child::dash:SegmentTimeline)) then if (count(child::dash:Url) > 1) then false() else true()]

**Explanation:** The segment duration attribute provides the approximate duration of every segment in the representation except for the last one that might be different. Duration information in the form of “duration” attribute of Segment Time Line sub element is to be present.

if (ancestor::dash:MPD/@type = 'OnDemand' and @startIndex != 1) then false() else true()]

**Explanation:** Start index of the segments in the representation start from 1 in case the profile is “On Demand”. The start index start from one and continues to last segment in the same representation or a different representation subject to switching taking place.
3.3.3.1 Results for Schematron Rules Checking

A couple of results are produced to provide the practical implementation of rules. Consider the Case No. 1 for instance a MPD as depicted in figure 20.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:mpgB:schema:DASH:MPD:DIS2011"
     xsi:schemaLocation="urn:mpeg:mpgB:schema:DASH:MPD:DIS2011 dash.xsd" type="Live"
     availabilityStartTime="2001-12-17T09:40:57Z">
  <Period start="PT0S">
    <Representation mimeType="video/mp4; codecs=avc1.644028, svc1" group="1" width="320" height="240" frameRate="15" id="tag0" bandwidth="128000">
      <SegmentInfo duration="PT10.00S">
        <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
        <Url sourceURL="seg-s1-128k-1.mp4"/>
        <Url sourceURL="seg-s1-128k-2.mp4"/>
        <Url sourceURL="seg-s1-128k-3.mp4"/>
      </SegmentInfo>
    </Representation>
  </Period>
</MPD>
```

Figure 20 – Case 1 - Test MPD

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- <schematron-output xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="urn:mpeg:mpgB:schema:DASH:MPD:DIS2011"
 xsi:schemaLocation="urn:mpeg:mpgB:schema:DASH:MPD:DIS2011 dash.xsd" type="Live"
 availabilityStartTime="2001-12-17T09:40:57Z">
  <Period start="PT0S">
    <Representation mimeType="video/mp4; codecs=avc1.644028, svc1" group="1" width="320" height="240" frameRate="15" id="tag0" bandwidth="128000">
      <SegmentInfo duration="PT10.00S">
        <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
        <Url sourceURL="seg-s1-128k-1.mp4"/>
        <Url sourceURL="seg-s1-128k-2.mp4"/>
        <Url sourceURL="seg-s1-128k-3.mp4"/>
      </SegmentInfo>
    </Representation>
  </Period>
</MPD> -->
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svrl:schematron-output xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="urn:mpeg:mpgB:schema:DASH:MPD:DIS2011"
 xsi:schemaLocation="urn:mpeg:mpgB:schema:DASH:MPD:DIS2011 dash.xsd" type="Live"
 availabilityStartTime="2001-12-17T09:40:57Z">
  <Period start="PT0S">
    <Representation mimeType="video/mp4; codecs=avc1.644028, svc1" group="1" width="320" height="240" frameRate="15" id="tag0" bandwidth="128000">
      <SegmentInfo duration="PT10.00S">
        <InitialisationSegmentURL sourceURL="seg-s-init.mp4"/>
        <Url sourceURL="seg-s1-128k-1.mp4"/>
        <Url sourceURL="seg-s1-128k-2.mp4"/>
        <Url sourceURL="seg-s1-128k-3.mp4"/>
      </SegmentInfo>
    </Representation>
  </Period>
</MPD>
```

Figure 21 – Case 1: Schematron Output Result
Case No 2:

The MPD apparently seems fine but in accordance to conformance it violates the schematron rule pertaining to MPD level. This when seen in result is confirmed as the rules fired at every level assert the rules applied to MPD. The minBufferTime attribute is not found at MPD level or at its child i.e “period” level. The semantics of schematron rules follow SVRL (Schematron Validation report language) active pattern and if a deviation is found the rule that doesn’t comply is displayed in text in the form of an XML file as shown in figure 21.

Figure 22 - Case 2: Test MPD

Figure 23 - Case 2: Schematron Output Result
In case 2 again, the working of Schematron rules is shown where this time the violation takes place at the period level as the mutual relationship between the segment alignment and Bit stream switching is not maintained. As seen from the figure 22 the values for the theses attribute conflict with what is defined in the DASH schematron schema so when the rules are fired for checks the schematron immediately displays the error message that can be seen in text from figure 23.
CHAPTER 4 FILE FORMAT CONFORMANCE

4.1 File format conformance checks

In the previous section the conformance checking was limited to the MPD exchanged between the client and server for the streaming to take place. If the exchange of information is viable according to rules and regulation laid out by the standard don’t necessarily dictates that the content that is used for the streaming purpose also conforms. The information included in the MPD should coincide with the structure and format of the media content.

This preposition carries significant importance since the content for streaming is in abundance over the internet. For instance a single server farm may have different copies of the same media content used for a particular stream from various resources. This content variability is subject to different requirement as per the media players, network constraints, type of codec support etc. It may be the case that for the streaming the server may generate content which might not conform totally to the features mentioned in the MPD. In case the DASH streaming might be disrupted and performance will be affected.

In this stage of conformance, the checks are extended to the file format or media content level. Going deeper to media content level clarifies the minute details specified by the standard and gives a clear picture of what features are supported by DASH and the importance they carry. Although DASH specifies mainly the ISO base media file format or MPEG-2 Transport stream (TS) as the DASH content type, but in this discussion only ISOBMFF is being discussed for conformance purposes. Before hand, the details of the file format structure and the fields involved are discussed so a basic knowledge of physical entities can be established.

4.2 ISO Base Media File Format specified for DASH

According to standard specification of DASH the media content can be accessed by using HTTP 1.1. This offers the leverage to have persistent connection that allows multiple requests to be queued under the same process. This resolves the problem of creation of new connection stages of TCP and drop in window size each time the process takes place. Apart from this, the byte range request feature allows partial requests for the media content that is very useful while accessing sub segments defined by a particular byte range.

Generally three type of media content are specified for DASH [4]:
- Initialization segment that contains the initialization information to gain access to the media segment containing the desired content.
- Media Segments, The actually media content itself provided in the form of segments (pieces)
- Self Initialized media segments, are media segment that don’t need a separate initialization segment and they provide authentic DASH content. Although up till this point the self initialization segment have not been tested or used in practical scenarios. So it’s not included as well for thesis work.
4.2.1 Initialization Segment

As the name suggests it provides the Meta data and information required by a media decoder to play the media content. Now the information that is required for initialization purposes as per the ISOBMFF consists of a range of boxes and sub fields. The idea here is to provide an overview of these boxes and structures to an extent that a ground level understanding of the fields and their function can be comprehended. A skeleton structure of the main boxes included in initialization segment is presented as follows:

4.2.1.1 FTYP (File Type Box)

Being an optional box in the ISOBMFF, FTYP box makes itself a mandatory aspect in DASH. Ftyp box comes before any other box in a file. It is positioned first in the order containing fields of “major brand”, “minor version” and “compatible brands”. These fields provide the information for the “best use” of the file, the minor versions associated with the file and other compatible specifications that it supports [18]. Each brand is four character code that is registered with ISO which defines a particular specification. The FTYP field definition as per the ISOBMFF standard [18]:

- **Major Brand** — Brand Identifier (For DASH SW implementation has value of “3gh9”)
- **Minor Version** — Informative integer for minor version of major brand (For DASH SW implementation has value of “512”)
- **Compatible brand** — list of brands (For DASH SW implementation has value of “DASH” specified by standard ISO/IEC 23001-6)

An extract of FTYP box using MP4v2 Linux tool is shown in figure 24 depicting the major and minor Version fields for the DASH content in use.

![Figure 24 - Initialization Segment Construction](image-url)
4.2.1.2 MOOV BOX (Movie Box)

Movie box makes the backbone of the initialization segment as it contains the major meta data information required for the initialization of media player to play out the content. It contains the movie header box that provides information like creation time, modification time, timescale and duration pertaining to media file. MOOV box also contains the track information that actually constitutes the media. Tracks can be audio or video and the Meta information pertaining to the tracks is found in TRACK box inside the MOOV box. Track header box will provide detailed information about that particular track aided by “mdia” media information in the track. The “minf” media information container inside the “mdia” box provides the information regarding details aspects like video and audio of track with the help of “vmhd” video media header and “smhd” sound media header.

While testing the media content on the commercially available media player “MPlayer” for Linux platform the sample description is of prime importance. Since the entire media presentation at the building block level is made up of sample bundled together to form segments and then these segments join together to form the entire presentation. “Stsd” – Sample Descriptions can be found inside the Sample table box and provides detailed information relating to the initialization content. A sample extract of the stsd box is shown in figure 25.

```plaintext
| type std (moov.trak.mdia.minf.stbl.stsd) |
| version = 0 (0x00) |
| flags = 0 (0x000000) |
| entryCount = 1 (0x00000001) |
| type mp4v (moov.trak.mdia.minf.stbl.stsd.mp4v) |
| reserved1 = <6 bytes> 00 00 00 00 00 00 | ...... |
| dataReferencelndex = 1 (0x0001) |
| reserved2 = <16 bytes> 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ........... |
| width = 640 (0x0280) |
| height = 480 (0x01e0) |
| reserved3 = <14 bytes> 00 48 00 00 00 48 00 00 00 00 00 00 00 00 01 | .H...H....... |
| compressorName = |
| reserved4 = <4 bytes> 00 18 ff ff | .... |
| type esds (moov.trak.mdia.minf.stbl.stsd.mp4.esds) |
| version = 0 (0x00) |
| flags = 0 (0x000000) |
| bufferSizeDB = 0 (0x000000) <24 bits> |
| maxBitrate = 1024000 (0x000fa000) |
| avgBitrate = 0 (0x00000000) |
| decSpecificInfo |
| info = <45 bytes> |
| 00000000 00 00 01 b0 01 00 00 01 b5 89 13 00 00 01 00 00 | .......... |
| 00000010 00 01 20 00 c4 8d 88 00 f5 14 04 3c 14 63 00 00 | .. ...<.c.. |
| 00000020 01 b2 4c 61 76 63 35 32 2c 37 32 2e 32 | Lavc52.72.2 |
| profileLevelIndicationIndexDescr |
| slConfigDescr |
| predefined = 2 (0x02) ................................. |
```

Figure 25 - Fields of ‘stsd’ Box inside MOOV box
The “stsd” specified above is for the test content. It is for the TRACK ID 1, which for this content is video. As can be seen that the “stsd.mp4v” box intimates that the video used is mpeg 4 with the resolution parameters of 640x480, maximum bit rate of the video 1024Kbit/s and Decoder Specific Information that provides the decoder type of “Lavc 52.72.2” for this content. These set of information is required by the “MPlayer” to play out the content and any tampering or missing parameter yields in an error produced by the player. Another similar “stsd” box is also present in the initialization segment that is for the TRACK ID 2 i.e the audio track. The audio tack has information like the audio is mp4a, number of channels = 2, maximum bit rate of 64Kbits/s etc. Every tack will provide these details in the initialization segment.

Another point stated by the DASH standard ISO/IEC 23001-6 is that “the tracks in MOOV box will not contain any samples). These samples are mainly concerning to the following boxes:

- “stts” decoding time to sample box (Contains the decoding delta times)
- “stsc” sample to chunk box (sample are grouped into chunks of different sizes)
- “stco” chunk offset table (gives offset of each chunk)

Since these and few other boxes of similar types contains fields of “entry count” or “sample count”, so according to DASH standardization that all sample count field should be equal to zero so that the size of the initialization segment could kept to a minimum and the amount of time to download the initialization segment (start time) is as minimum as possible before the media segment can be accessed [4]. So minimizing the initialization data is essential to make sure that media segment is accessed as soon as possible because the actual media sample reside in the Media segments.

### 4.2.1.3 MVEX Box

MVEX – Movie Extends box intimates the user that there may be more movie fragments that are expected. The mvex box has TREX – Track Extend box for each track present which specifies the default values of parameters like sample size, sample duration, sample flags etc. TREX must be present with default values or else “MPlayer” generates an error of “TREX” box missing while decoding the media content.

### 4.2.1.4 Pdin Box

Progressive Download Information is an optional box in DASH specification. It has not yet been seen in practice so far in the implementation done.

### 4.2.2 Media Segment for DASH

As seen by now that the initialization provides the major details to prepare and intimate the player about the content itself. The client uses the information in the “moov” box to identify the available media components and their characteristics [4]. Now moving to the media segments that actually contain the samples for the media the media segments have the formation as shown in figure 26.
The format of the media segment for DASH specifies a few new boxes that are to be included in this case primarily “styp” - Segment Type Box, “sidx” - Segment Index Box and “ssidx” - Sub Segment Index Box.

### 4.2.2.1 STYP Box

Segment type box is an optional box for DASH that can be included in the beginning of a media segment just like FTYP box in the initialization segment. The role of STYP is a replica of that of FTYP. The change is the scope to which it applies i.e that segment level. STYP box also carry the same fields of major Brand, minor Version and compatible brands. This box is new inclusion to ISOBMFF for 3GPP DASH specific purposes as mentioned in [6]. The compatible brand filed in STYP box is governed by the Media Segments types that are categorized as:

- Simple Media Segments (Compatible Brand : “msdh”)
- Indexed Media Segment (Compatible Brand : “msix”)
- Sub Indexed Media Segments (Compatible Brand : “sims”)

### 4.2.2.2 SIDX Box

SIDX box is DASH specific box added for the Indexed media type segments. The placement of SIDX box is always before the start of MOOF box and after the STYP box if present. The number of SIDX boxes depends on the number of tracks contained in the media segment since each track has a corresponding SIDX box at the start of the segment. Primarily the SIDX documents the information regarding the sub segments contained inside so that the client even before requesting the whole segment gets aware of the specific information about the content of the media segments in terms of the sub segments it contains.

A SIDX box plays a vital role to determine the composition details of sub segments that finally compose the entire segment. The SIDX box is consulted by the client to access a specific sub segment within a segment or to seek to later presentation time inside the segment. The information that is included inside a SIDX is shown in table 4.
Table 4- SIDX Box Entries

<table>
<thead>
<tr>
<th>General Information About the Segment as a Whole</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference ID</td>
</tr>
<tr>
<td>Timescale</td>
</tr>
<tr>
<td>Earliest presentation Time</td>
</tr>
<tr>
<td>First offset</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information Included for Each Sub Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Type</td>
</tr>
<tr>
<td>Reference Size</td>
</tr>
<tr>
<td>Sub Segment Duration</td>
</tr>
<tr>
<td>Contains SAP</td>
</tr>
<tr>
<td>SAP delta Time</td>
</tr>
</tbody>
</table>

This information enables the client to just download a small portion in the start of segment (SIDX boxes) and then make an informed decision as to which part of the media segment to hop to for access. The concept for SAP will be discussed later in the section of conformance procedures for media files.

4.2.2.2 MOOF Box

Media segment are supposed to have self contained movie fragment boxes (MOOF). The self contained aspect point to a “moof” box and “mdat” box that contain the sample of the media content that do not use external data references in Track runs (TRUN boxes) in the movie fragment [4]. MOOF box can be regarded as the box that links details to samples actually contained in MDAT box. Every MOOF box corresponds to an MDAT box. For the purposes of DASH it becomes a necessity that a MOOF box contains at least one track fragment box (TRAF) because otherwise an empty MOOF box doesn’t serve the purpose and make no sense. So at least one track of audio, video or any other type is a must to be in a MOOF box. As can be seen from the media segment format in figure 26, MOOF box primarily comprises of the “MFHD”- Movie fragment header box and one or more “TRAF”- Track fragment box.

4.2.2.4 MFHD

Movie fragment header box contains the “sequence number” attribute that intimates the moof box of the numbering in the segment. Sequence number provides the count of the total number of MOOF boxes as this information can be associated with determining the sub segments that the Media file contains.

Although there is no limit to the number of MOOF boxes it takes to compose a single sub segment but in our test case with the DASH content generated it was observed that a single MOOF box corresponds to a single sub segment. So the sequence number count in that case intimated the number of sub segments. But it could be that a single sub segment is made up of more than one MOOF box as well.
4.2.2.5 TRAF

Track fragment box/boxes are present in MOOF box to correspond to the tracks present in the media segment. TRAF box in turn consist of a number of Track Run boxes (TRUN) that present a continuous run of samples for a particular track [18]. Mainly sub boxes for TRAF are “TFHD” - Track fragment Header box, “TFDT” – track fragment decode time and “TRUN”- Track Fragment Run Box.

4.2.2.6 TFHD

This box tends to provide the specific details about contiguous run of samples concerning a specific TRACK in the media file. The box consists of main “TRACK ID” field and a number of optional fields like

- Base Data Offset
- sample_description_index
- default_sample_duration
- default_sample_size
- default_sample_flags

For any type of segment i.e., simple, indexed or sub indexed segments the flag “default_base_is_moof” is set that means that the “Base Data Offset” in TFHD is set to the data offset field extracted from the MOOF box as part of DASH specifications. The rest if default information is present can be used to estimate the statistics of the samples that comprise the TRACK like its size, duration and flags (indicating the presence or absence of fields in the TFHD box).

4.2.2.7 TFDT

Track fragment decode time is a DASH specific box that is mandatory aspect of the TRAF box. This box documents the Base Media Decode Time for a particular Track that the TRAF box refers to in the segment. The Base media decode time is in timescales in TFDT box.

4.2.2.8 TRUN

Track fragment run box makes the most essential part of the media segment. Although it should be noted that the actual media sample reside in the MDAT box but every minute information concerning the samples lie in the TRUN box. The box plays a significant role in the file format conformance procedures as the data acquired and formulated from the TRUN box forms the basis of the conformance checks. In order to understand the conformance at this level the field and attribute of the TRUN box are to be observed more deeply.

TRUN box consists of “Sample count” field showing the length if the contiguous samples for a particular track. It also then has a “Data Offset” field that can intimate an offset that should be added to Base Data Offset (Offset of MOOF box in case of DASH) to get the Start offset of the first sample in the TRUN box.

A number of optional field in TRUN box also play an essential part in the Conformance procedure:
- sample_duration
- sample_size
- sample_flags
- sample_composition_time_offset
A brief overview of the calculation and use of the fields in TRUN to determine particular stats is presented as follows:

- The composition time that are denoted by presentation time in DASH specification are calculated by addition of Base Media Decode time from the “TFDT” box to the “Sample Composition Time Offset” that can be checked by the presence of its corresponding flag.
- Every time a new sample turns up its start time is determined by the Cumulative Base Media Decode time from the previous sample that is achieved by adding up the “Sample Duration” attribute of each sample.
- “Sample size” parameter for each sample is added up to the “Data offset” attribute of TRUN box to determine the offset for next sample in line.
- “Sample flags” play a vital role in terms of DASH to check if the sample is of type “Sample_is_difference_sample” that relates to the presence or absence of a Sync Sample. These sample form important conditions to the type of SAP (Stream Access point) presence. The details about SAP will be discussed later.

TRUN box make different length of samples groups for each TRACK. The number of TRUN produced by the DASH content generated depends upon the amount of Interleaving specified. With very low values of interleaving a large number of TRUN boxes are produced with a relatively small number of samples in each of them. Where as if the Interleaving size is kept equal or close to the sub segment duration a minimum number of TRUN boxes are produced containing a large sum of samples. Although the effect of number of TRUN box on media decoding could not be apparently seen from the testing that was conducted on DASH content presently available.

### 4.3 Sub Indexed Media Segments

These media segments relate to the presence of sub segment index box “ssidx” that follows the “sidx” box. The “ssidx” box documents the sub segment information. The preceding “sidx” box will only index the sub segments [6]. So far the sub indexed media segment have not been tested in the DASH scenarios but might be included in future implementations.

### 4.4 File Format Conformance Checking

The previous discussion on the media segments types and structures has laid down the ground work for the discussion on the file format checks. The first Stage of the conformance checking involved dealing with the MPD alone and the element and attribute it contained. The further point of concern now is that all the information presented by the MPD in turn relates to the media content. So, it is essential to cross reference this information by moving to the media file level and confirming the features and attribute that are intimated to the client via the MPD.

Generally analyzing the elements and attribute of an MPD, there are a few areas that particular fall in the category of file format level checks. It is very essential to define the scope that can encompass the areas that should be checked if they are intimated by an MPD.

Another challenge is that the DASH standard ISO/IEC 23001-6 [4] present attributes and elements by providing a theoretical description for them and as well as the rules concerning them. The task here is to check that the definitions are clear and any ambiguity concerning the abstract nature of these attributes should be solved. So, when conformance procedures are laid out it is vital to have a clear understanding of the conceptual definitions so that when implementation of conformance involving these attributes start at the software level all concepts can be translated to a practical concept. So
the task at hand requires that the scope of the file format should be defined by listing candidate checks that forms a practical scenario to be applied to media segments.

4.5 Defining Scope for Checks

The scope of the checks can be segregated in mainly two areas:
- ISO Base media File Format Level
- Bit Stream level

Now, checks on both levels are according to the conditions and rules specified by the DASH Standard itself. Firstly, the ISOBMFF level is explained by further splitting the process into two levels namely Initialization segment and Media segment. Then Bit stream Level is explored by addressing the MPD flags confirmation and Hierarchical rules implementation.

Figure 27 - File Format Conformance Schematic

4.6 ISO Base media File Format Level

This stage mainly focuses on the box structures and the content of the fields that they hold. The rules that are laid out are extracted from the standard and can been seen in table 5.

Table 5 - ISOBMFF Level Conformance Checks

<table>
<thead>
<tr>
<th>INITIALIZATION SEGMENT LEVEL</th>
<th>Checks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>‘FTYP’ box is present</td>
<td>Provides information about the compatible brands that should specified as “dash”</td>
</tr>
<tr>
<td></td>
<td>‘MOOV’ box is present</td>
<td>This is essential to provide initialization details to the media player to initialize playing out of content</td>
</tr>
<tr>
<td></td>
<td>‘MOOV’ box with empty field for ‘stts’, ‘stsc’ ‘stco’</td>
<td>The entry/sample count for these boxes inside MOOV boxes is set to zero to reduce the download time.</td>
</tr>
<tr>
<td></td>
<td>‘MVEX’ box is present</td>
<td>Intimates the client about movie fragment boxes and sets defaults values of certain parameters</td>
</tr>
<tr>
<td></td>
<td>No “moof” or “mdat” boxes</td>
<td>The initialization segment will contain no samples, hence no “MOOF” or “MDAT” is to be included in “MOOV” box</td>
</tr>
</tbody>
</table>
### MEDIA SEGMENT LEVEL

<table>
<thead>
<tr>
<th>Checks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOOF is present</td>
<td>MOOF box should be present as indicates the presence of media samples</td>
</tr>
<tr>
<td>Self Contained Structure</td>
<td>MOOF box is followed by MDAT box to form a self contained structure</td>
</tr>
<tr>
<td>At least one track fragment box</td>
<td>One or more TRAF boxes so that at least samples of a particular track are available as media content</td>
</tr>
<tr>
<td>TRAF should have TFDT</td>
<td>TFDT is mandatory to provide the Base Media Decode Time for the segment</td>
</tr>
<tr>
<td>Compatible brand for Last DASH segment</td>
<td>STYP box of last segment may carry 'lmsg' as compatible brand [6].</td>
</tr>
</tbody>
</table>

#### Conditional to Availability

<table>
<thead>
<tr>
<th>Checks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIDX is present</td>
<td>Indexed media segment will carry “SIDX” box and SIDX box conforms to the ISO/IEC TC1/SC29/WG11 MPEG2010/M20339 (Amendment) [19]</td>
</tr>
<tr>
<td>‘STYP’ is present</td>
<td>In case STYP is present, for Indexed Media Segments it will have ‘msix’ and for Sub Indexed media segments it will have ‘sims’ as compatible brand.</td>
</tr>
<tr>
<td>‘SSIX’ is present</td>
<td>Sub Indexed media segment will carry “SSIX” box and SIDX box conforms to the (Amendment 3) [20].</td>
</tr>
<tr>
<td>‘SIDX’ box should come before any ‘moof’ box and ‘mdat’ box</td>
<td>SIDX should always be placed before MOOF and MDAT for client to know information without downloading the segment</td>
</tr>
</tbody>
</table>

### 4.7 Bit stream Level

The conformance procedure at bit stream level tries to evaluate checks that are governed by statistics at the MPD level exchange. The most viable checks at this stage are the confirmation of the flags and attributes that are specified in MPD. These checks mainly comprise of the following:

- Segment Alignment
- Sub Segment Alignment
- StartWithRAP
- MaxRAPPPeriod
- BitStreamSwitching

**Challenge Poised At this Level**

These flags and attributes previously mentioned in the DASH standardization were theoretically stated and the implementation scope to cross reference them was a bit abstract and vague. Steps
4.7.1 Segment Alignment Flag

This flag principally resides at the Period level element of MPD and is included as a period attribute. The previous definition of the “segmentAlignmentFlag” states [4]:

“ When set to ‘true’, indicates that all presentation start and end times of media components of any particular media type are temporally aligned in all Segments, except possibly the last one, across all Representations with the same value of the @duration attribute on Representation level in this Period.”If the @segmentAlignmentFlag attribute is set to ‘false’, the @bitstreamSwitchingFlag shall be set to ‘false’. “

Now, here the temporal alignment of segments when interpreted in term of the segment’s physical parameter becomes a bit vague to understand. Segment Alignment basically relates to the fact that the segment should not overlap in presentation times. This should hold between segments across the representations in a single period. The @duration parameter should be same on the Representation level. To make the flag more transparent it is now interpreted according to [22]:

“When set to ‘true’, then this specifies that for any two representations, A and B, with the same value of the @duration attribute, the i-th segment of A and the j-th segment of B are non-overlapping whenever i is not equal to j.”

This last definition clarifies the flags in terms of implementation as it categorically states that overlapping of segment’s presentation time having different indexes, between two representations is not allowed when the flag is set to ‘true’. This overlapping is avoided to ensure smooth switching and a valid bit stream for the video play out.

4.7.1.1 Implementation Details for Conformance

The conformance of this flag depends upon the value of the flag when it appears in the MPD. For instance when set to ‘true’ it relates that there is no overlapping taking place and to be checked at the media segment level the following information is required:

- Earliest Presentation Time of Media Segment (EPT)
- Latest Presentation Time of Media Segment (LPT)
- Segment Duration

To explain further consider the two representations A & B in figure 28. According to the Flag’s definition the EPT of segment 2 in Representation B should be greater than the LPT of segment 1 in representation A to ensure that no overlapping of presentation time take place.

From figure 28, the Segment Alignment check schematic can be visualized. The point of interest is to find the presentation time of samples within a segment that have the minimum and maximum values so that they can be assigned to earliest presentation time and Latest presentation times respectively. The vertical bars on the segment depict the sample it contains. The red double arrow relate to checks on the EPT and LPT of segments in order to maintain segment alignment condition.
Before calculation the EPT and LPT for the segments an important concept about the difference between decode time, composition time and presentation time is to be understood to avoid any confusion later in the discussion.

The decode time of sample corresponds to the sum of sample duration that are accumulated over the run of samples. If the frames in a sample i.e. I/P/B frames are ordered properly then the decode time is the same as the presentation time. But it might happen if the frames are reordered and in this case an offset is required to convert the decode time to the presentation time. This offset that is added up to the decode time forms the composition time or presentation time. So, in case I/P/B frames are not ordered then composition time is calculated first that actually form the presentation times. In order to determine the information needed the things that need to be extracted from the media files are:

- ‘TFHD’ Box : Track ID
- ‘TFDT’ Box : BaseMediaDecodeTime
- ‘TRUN’ Box : Sample Count, Sample Duration, Composition Time Offset

The presentation time calculation will take place according to following steps:

1. ‘Base Media Decode Time’ is extracted from the ‘TFDT’ box and this gives the start time of the segment.
2. For the 1st sample in TRUN Box the Composition time will be:
   \[ \text{Composition Time of 1st Sample} = \text{Base Media Decode Time (TFDT)} + \text{Composition Time Offset} \]
   (From sample info in ‘TRUN’)
3. For next sample’s decode time and further, the sample duration is added from Sample Info in TRUN Box (+=Sample Duration)
   \[ \text{Composition Time of 2nd Sample} = \text{Base Media Decode Time (TFDT)} + \text{Sample Duration of 1st Sample} + \text{Composition Time Offset} \]
   (From sample info ‘TRUN’)
   \[ \text{Composition Time of 3rd Sample} = \text{Base Media Decode Time (TFDT)} + \text{Sample Duration of 1st \& 2nd Samples} + \text{Composition Time Offset} \]
   (From sample info ‘TRUN’)

Figure 28 - Segment Alignment Schematic
4. Hence, the general form is:

\[ \text{Composition Time of } i^{th} \text{ Sample} = \text{Base Media Decode Time (TFDT)} + \text{Sum of Sample Duration of last } (i-1) \text{ Samples} + \text{Composition Time Offset of } i^{th} \text{ sample (From sample info 'TRUN')}. \]

5. As the composition times of the sample are being calculated the minimum of these samples composition time will be the earliest presentation time and the maximum will be the latest presentation time.

6. The scope of samples under consideration is the entire segment (all 'MOOF' Boxes)

Along with the process of composition time evaluation the segment duration for each segment is also determined by the difference of EPT between successive media segments i.e.:

\[ \text{Duration of Segment 'i'} = \text{EPT of Segment '}(i+1)' - \text{EPT of Segment 'i'} \]

4.7.1.2 Conformance Checks for Segment Alignment

Since the above procedure provides the required presentation times and duration attributes, so to fulfill the segment alignment criteria the following condition are satisfied.

- Duration of segments across representations should be the same
- LPT of Segment 'K' < EPT of Segment 'K+1' (In the same Representation)
- LPT of Segment 'K' in Representation A < EPT of Segment 'K+1' in Representation B

Note: Here, an important aspect to be noted is that the segment alignment will take place for all the track id’s that are present in the media file i.e audio, video etc.

4.7.2 SUB SEGMENT ALIGNMENT FLAG

The sub alignment flag like the segment alignment flag also concentrates on the overlapping boundaries but the scope is now shifted to the sub segment level. Here as well a conceptual clarification of the check was needed so that it can have a physical shape for implementation. The sub segment alignment states [22]:

- Each Media Segment shall be indexed (i.e. either it contains a Segment Index or there is an Index Segment providing an index for the Media Segment)
- For every Sub-Segment with contains_SAP =1, the T_{RAP} value of the first RAP in the Sub-Segment shall be the earliest presentation time of any sample of the reference stream of the Sub-Segment in which it appears.
- For any two Representations, A and B, within the same Adaptation Set, the i-th subsegment of A and the j-th subsegment of B are non-overlapping whenever i is not equal to j.

From the above statements, the information required to verify the sub segment alignment includes:

- Earliest presentation Time (EPT) for each sub segment
- Latest presentation time (LPT) for each sub segment
- Presence of SIDX is essential as it documents the sub segment information of:
  - EPT of Segment
  - Sub Segment Durations
**Test Case for Checking**

In order to demonstrate the sub segment alignment checking procedures and the calculation done in this respect, let us consider a test case of a representation having segments with a maximum of two sub segments. So in this case the EPT and LPT will be demonstrated for two sub segments. The first step in sub segment alignment checking is to calculate the earliest presentation time of sub segments (SSEG’s).

**Earliest Presentation Time for SSEGs**

1. First Check that the EPT present in SIDX and the EPT calculated for the Segment (Segment Alignment) are the same.
2. If the SIDX has “contain_SAP” field set to 1 then the EPT of 1\(^{st}\) Sub segment will be:

   \[
   \text{EPT}_{1^{st}\text{SSEG}} = \text{EPT (SIDX) or EPT of the Segment (also equal to Tsap if type is 1 or 2)}
   \]

   (SAP delta time is the difference between the SAP presentation time and the EPT of the segment. Type of SAP will be discussed in “StartWithRAP” flag)

   \[
   \text{EPT}_{2^{nd}\text{SSEG}} = \text{EPT of 1\text{st} SEG} + \text{SSEG duration of SEG 1 (SIDX)} \quad (\text{also equal to Tsap if type is 1 or 2})
   \]

3. End of Segment = EPT of 2\(^{nd}\) SEG + SSEG duration of SEG 2 (SIDX)

So in general the EPT of a sub segment is calculated as:

\[
\text{EPT of sub segment } 'i' = \text{EPT of Segment} + \text{Sum of SSEG duration of (i-1) sub segments}
\]

A visual representation of the EPT concept for sub segments can be seen from figure 29.

![Figure 29 - Earliest presentation Time of SSEG](image)

**Latest Presentation Time Calculation for SSEGs**

The latest presentation time of sub segment will be just before the start of a new one. At present a single movie fragment is being considered as one sub segment. So, the samples in a single MOOF box constitute the EPT and LPT a particular sub segment. So the LPT of sub segments 1 and 2 in the present test case are computed as:

\[
\text{LPT}_{1^{st}\text{SSEG}} = \text{Presentation time of a sample that is just before the EPT (or Tsap if type is 1 or 2) of the next Sub segment (2\text{nd SEG}) – In timing order}
\]

\[
\text{LPT}_{2^{nd}\text{SSEG}} = \text{Presentation time of a sample that is just before the End of segment presentation time.}
\]
Point to note is the LPT of SSEG 2 and end of segment are different point as they have a difference of the last sample duration.

### 4.7.2.1 Conformance Checks for Segment Alignment

After determining the EPT and LPT of each of sub segments in the representations, the following conditions are checked to be valid:

- SIDX is present with Sub segment information
- EPT in SIDX is equal to EPT of segment calculated (By Segment Alignment)
- LPT of Sub Segment ‘K’ < EPT of Sub segment K+1 (In the same Representation)
- LPT of Sub Segment ‘K’ in Representation A < EPT of Sub Segment ‘K+1’ in Representation B
- Just to check the authenticity of the SIDX information check that the sub segment duration present in the SIDX follows the following logic:

  \[ \text{Sub segment Duration} = \text{Difference of earliest presentation time of the current sub segment and the one that follows} \]

*Note: At present only Sub segment alignment of the reference Track ID can be taken since there is only a single SIDX box present. Whereas the sub segment alignment should be applied to all track id’s present in the media segment*

### 4.7.3 “STARTWITHRAP” Conformance Check

Start with RAP attribute depicts that the segments present in the representation have start from a representation access point. In principle, a RAP (representation access point) can be regarded as a position in the representation from which the play back can be started with only the information from that point onwards. That means this point is not dependant on any previous information present in the representation and can start off the video play back ahead in time. Usually this dependence is corresponding to B or P frames that are dependent on other frames for decoding purpose. So RAP actually corresponds to the presence of an IDR frame (a special type of I frame in MPEG-4) that in turn relates to no referencing back to frames preceding the IDR frame [23]. So for an RAP to occur, every stream present in media content (more commonly known as the tracks (audio, video,)) should have separately its own SAP (stream access point) that in definition has the same principle as a RAP.
Some of main properties of SAP according to [19] are:

- $T_{SAP}$ is the earliest presentation time of any access unit of the media stream such that all access units with presentation time greater than or equal to $T_{SAP}$ can be correctly decoded using data in the Representation starting at $I_{SAP}$ and no data before $I_{SAP}$.
- $I_{SAP}$ is the greatest position in the Representation such that all access units of the media stream with presentation time greater than or equal to $T_{SAP}$ can be correctly decoded using Representation data starting at $I_{SAP}$ and no data before $I_{SAP}$.
- $T_{EPT}$ is the earliest presentation time of any access unit of the media stream starting at $I_{SAP}$ in the Representation.

In view of the above properties associated with SAP, they can be categorized in mainly three types [19]:

- **Type 1:** Can be regarded as “Closed GOP random access” point where the first unit in decoding order is also the first access unit in presentation order.
- **Type 2:** Can be regarded as “Closed GOP random access” point where the first unit in decoding order is not also the first access unit in presentation order.
- **Type 3:** Can be regarded as “Open GOP random access” point where there are some access unit after $I_{SAP}$ that cannot be correctly decoded having presentation time less than the $T_{SAP}$.

Now, with the preliminary concepts about a SAP being laid out the conformance procedure demand that if the “StartwithRAP” attribute is mentioned in the MPD it should be verified at the file format level as well. The attribute appears at the “CommonAttributeElement” level and bears a value amongst 1,2,3 wich means that any of type less than or equal to attribute value may be present. That is if “StartwithRAP” =2 it mean that either RAP type 1 or type 2 is present.

In order to verify the RAP, Presentation time of SAP for each of the track contained in the media segment is to determine as well the type to which it belongs. When all the SAP’s for each segment in all streams are known, the Trap will be the greatest $T_{SAP}$ amongst the multiple streams.

“StartWithRap” calculation can be presented in light of two cases i.e with or without subsegment.

### 4.7.3.1 Case: With no Sub-segments (no SIDX)

**Step 1: Defining $T_{SAP}$ for each segment and noting the position in decoding order**

- Pick up a track ID or stream in the media segment
- In the TRUN box look for the sample with Composition time zero and assign it as $T_{SAP}$ for Segment number 1.
- Note the sample index position in the decoding order to find the type of SAP.
  - $T_{SAP}$ (Segment 1) = Sample in TRUN box with Composition time zero
- To get the start time of next segment apply:
  - Start time of the next segment (Segment 2) = $T_{SAP}$ (Segment 1) + Segment Duration of Segment 1
- In the next Segment look for the sample with the composition time equal to the start Time of Segment 2 and assign it as $T_{SAP}$ (i.e. of Segment 2)
Note whether this sample is indexed first in the decoding order or note to find the type of SAP.

Follow the procedure for all segments in the representations.

Follow same procedure for all Track ID’s

**Step 2: To find the Trap and Type:**

- Trap = Greatest values of all $T_{sap}$’s for all tracks in a particular segment.
- After Identifying Trap, check for the position of Trap in decoding order.
- If it is the first in order, then the type is 1.
- If it is not first in order and all other samples in the TRUN box have Presentation time greater than the Trap than it is type 2.
- If there are samples in the TRUN box that have samples with presentation time less than Trap than type is 3.

4.7.3.2 Case: With Sub-segments (SIDX box present)

**Step 1: Defining $T_{sap}$ for each sub segment and noting the position in decoding order**

- Pick up a track ID or stream in the media segment
- In the TRUN box look for the sample with Composition time zero and it is the Earliest presentation time of the segment
- This EPT of the segment should be equal to EPT is SIDX
- $T_{sap}$ for sub segment (1) = Earliest presentation time in SIDX + SAP Delta Time (SSEG 1)
- $T_{sap}$ for next sub segment (2) = EPT+ sub segment duration (SSEG 1) + SAP Delta Time (2)
- $T_{sap}$ for next sub segment (3) = EPT+ sub segment duration (SSEG 1) + SAP Delta Time (2) + sub segment duration (SSEG 2) + SAP Delta Time (3)
- $T_{sap}$ of $i^{th}$ sub segment = $T_{sap}$ of $(i-1)^{th}$ sub segment + sub segment duration $(i-1)^{th}$ sub segment + SAP Delta Time of $i^{th}$ sub segment
- Note the indexing position of $T_{sap}$ in the order of decoding.

- Follow same procedure for all tracks

**Step 2: To find the Trap and Type**

- Trap = Greatest values of all $T_{sap}$’s for all tracks in a particular sub segment.
- After Identifying Trap, check for the position of Trap in decoding order.
- Check for Sync samples from “Sample_is_difference_sample” set to zero from the sample flags in TRUN box.
- If it is the first in order and the sample is a sync sample then the type is 1.
- If it is not first in order and all other samples in the TRUN box have Presentation time greater than the Trap and it is a sync sample than it is type 2.
- If there are samples in the TRUN box that have samples with presentation time less than Trap than type is 3.
4.7.3.3 Conformance Checks for STARTWITHRAP

- RAP is present at the start of the segment (Starts with a sync sample)
- RAP type can be verified

4.7.4 Bit Stream Switching Flag

Bit stream switching flag has close correlation with the segment alignment flag. This flag states that if it is set to true, it mean that the initialization segment in any representation of a particular period can be concatenated in sequence with segments from different representations of a period starting from segment number 1. This will result in a valid bit stream that when provided to a media decoder will yield valid results.

Segment Alignment is important to be true because if there are overlapping of segment presentation time then this valid bit stream creation from segment of different representation is not possible. This will be very evident in case of switching from one representation to another. Although the statement is very clear itself a demonstration of this flag via a working log analysis of at DASH client can shed more light in the validation of the flag.

Here, a DASH client accesses an MPD via a VLC player DASH plug in (by University of Klagenfurt). The URL for the MPD is provided in the Network Stream of the VLC to play out DASH content. The DASH content available is separated into four representations in the pattern of:

"Name – Duration – Codec – Resolution - Video Coding Rate - Audio Coding rate"

<table>
<thead>
<tr>
<th>Rep</th>
<th>Name</th>
<th>Duration</th>
<th>Codec</th>
<th>Resolution</th>
<th>Video Coding Rate</th>
<th>Audio Coding Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MtBike-5min-h264-hvga-v200-a48</td>
<td>5min</td>
<td>h264</td>
<td>hvga</td>
<td>200K</td>
<td>48k</td>
</tr>
<tr>
<td>2</td>
<td>MtBike-5min-h264-hvga-v400-a64</td>
<td>5min</td>
<td>h264</td>
<td>hvga</td>
<td>400K</td>
<td>64k</td>
</tr>
<tr>
<td>3</td>
<td>MtBike-5min-h264-hvga-v800-a128</td>
<td>5min</td>
<td>h264</td>
<td>hvga</td>
<td>800K</td>
<td>128k</td>
</tr>
<tr>
<td>3</td>
<td>MtBike-5min-h264-hvga-v1200-a128</td>
<td>5min</td>
<td>h264</td>
<td>hvga</td>
<td>1200K</td>
<td>128k</td>
</tr>
</tbody>
</table>

As the DASH stream is initiated the DASH client request for segments was logged via wire shark to see how the process enroll. The DASH process of segment request is provided in figure 31.

The process seems to start by the request of MPD (mtbike.mpd). The client then downloads the initialization segment of the very first representation that is having the lowest quality content (Video: 200K, Audio: 48K). But since the bandwidth available at the time totally permits higher quality content so the segments to follow are of the highest representation (Video: 1200K, Audio: 128K). But there are segments of another representation also seen i.e (video: 400K and audio: 64K).

This downgrade to a lower representation takes place when we manually restrict the bandwidth available at the client by use a bandwidth control tool “wonder shaper”. The tools restrict the available bandwidth to see how the client reacts. It is seen whenever the wonder shaper is used to restrict the bandwidth, the client switches to a lower representation and as soon as the bandwidth is set free is returns back to higher resolution representation. Initially for a small period of time the restriction to a lower representation is for a maximum segment duration which in this case is 10 sec. But in the end of log it can be seen that the bandwidth restriction for longer time duration. The switching in between segments is not possible in this case as there is no sub segments present.
The content available was non-fragmented and the segment alignment for the media content was set to ‘true’. The bit stream switching flag can be verified by the fact that only once in the beginning the initialization segment is downloaded. After that the segments switched between the two representations are concatenated to form a valid bit stream. So a sequence of media segments switching in between representation with only a single initialization segment provides a valid bit stream.

### 4.7.5 MAX RAP PERIOD Attribute Check

If present in the MPD, this attribute can be cross checked with the RAP calculation carried out in the “StartWithRAP” checks. The value should be in close proximity to each other.

### 4.8 Hierarchal Rules

Since new elements and attribute are being added with the passage of time, so this level takes into account the checks that can be applied to that particular level, like for instance

- For ‘On demand’ Representation Start should be equal to Period@start
- For ‘Live’ Representation start == Period@start+@availabilitystarttime
- if Segments are Self Initialized then one segment per representation
- MPD@mediaPresentationduration == Period@Periodstarttime(last period)+@duration
CHAPTER 5 SOFTWARE IMPLEMENTATION DETAILS

5.1 Software Implementation of FILE FORMAT conformance checks

As the main concepts of the file format conformance are explained, the software implementation abstract is provided here to have an overview of the implementation process in this regard. The main software used is the “DASH Reference Software” which is a proprietary tool of “QUALCOMM”. Although the semantics of the software code will not be provided here, but a general schematic of the process and the conformance addition will be highlighted.

This software model is only for the File format conformance checks which are actually the extension to the Qualcomm reference software. The integrated software discussed earlier chapter is for MPD conformance procedure that is mainly a Java Script developed to implement the three stage of MPD conformance. Both are separate softwares and are designed separate tasks.

The working process of the conformance procedure occurs parallel to the streaming process that is taking place. The two stages of conformance mainly (MPD and file format) are done by the two distinct softwares. As the streaming process initiates, firstly the MPD from the server is downloaded and is passed to the integrated software for MPD conformance. The Java script provides the results for the MPD and if it passes all the three stages of conformance the process is handed over to the DASH client. Here, as the processing of streaming continues with the downloading of segments and switching between representations, parallel to that file format conformance procedures are also executed. Any error in the process throws an exception putting a halt to the streaming process.

The DASH client is made up of mainly:

1. **HTTP access client**: Responsible for making requests and receiving the HTTP requested data as directed by the streaming client engine.
2. **Streaming Control Engine**: This is the main part that will determine the processing at client. It directs the client according to conditions about the segments to request and separates the processing depending on the type of segment that is requested. It can be regarded as the brain of the DASH client.
3. **Media Engine**: This part is given only the media segments so that it can decode them to extract the information required and then provide the media samples for play out to a media player.

![Figure 32 - DASH SW CLIENT MODEL](image)
The conformance procedure implementation is implemented on top of the existing architecture. The following steps are involved at each stage in the process:

- Defining a Data Base for Compliance procedure that can hold in the parameter and values of interest to be used in the checks.
- Proposed data base structure is a 3 dimensional array that have the following parameters:
  DATABASE [representation ID][Track ID][Segment number]
- Each element in the Conformance Data Base array should belong to a Compliance class. The compliance class is supposed to have the following parameters that are essential for checks.

  Table 6- Conformance Class Parameters

<table>
<thead>
<tr>
<th>CONFORMANCE CLASS PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>int SampleCount</td>
</tr>
<tr>
<td>double Rap_DeltaTime</td>
</tr>
<tr>
<td>double EPT_Segment</td>
</tr>
<tr>
<td>double LPT_Segment</td>
</tr>
<tr>
<td>vector&lt;double&gt; EPT_SSEG</td>
</tr>
<tr>
<td>vector&lt;double&gt; EPT_SSEG</td>
</tr>
<tr>
<td>double EPT_SIDX</td>
</tr>
<tr>
<td>vector&lt;double&gt; LPT_SSEG</td>
</tr>
</tbody>
</table>

- Segregating the processing of segments at the streaming client i.e the initialization segment and the SIDX box request should be catered for at the Streaming client engine where as the media segment request should be sent to the media player for processing.
- At this stage of conformance adaptive logic of the streaming client engine is not required. The streaming client engine should direct the HTTP access client to request segments in order of the representation. Like it should start with the initialization segment of first representation and carry on till the last segment of the last representation in the period.
- The implementation can be systematically seen from figure 33.
5.2 Guidelines for DASH media content Generation

DASH media content should be created according to specification of the DASH in ISO-IEC 23001-6 [4]. At present for DASH content, the proprietary “Reference Software Ingest Tool” from “QUALCOMM” is used. Another commercially available “MP4Box” tool from GPAC [21] is also present that produces DASH like content but it’s not fully compliant with the DASH specification. Like
mainly, it doesn’t have a “TFDT” box which is a requirement for DASH. Still more tools will eventually develop that will provide the option to create DASH content. In this development a few guidelines are important to be taken care of so that there is uniformity in content and the DASH client anywhere is able to play out the content. In doing so, the standard implementation will become more simple and ubiquitous. Principally, DASH content generation tool should have the following features:
- Option to make fragmented content
- At present, SIDX box should be included
- Option to specify segment duration
- Option to specify fragment duration
- Interleaving Interval (Optional but good to have)
- Formation of DASH Media Presentation Description file (.xml)

The tool for DASH creation should provide content with different video and audio bit rates, spatial resolution, codec so that a range of representation can be formed for the client to choose the content. This variety of content creation can be accomplished by using Linux based tool “FFMPEG” that will convert content in any desirable format that is needed. For the testing purposes of DASH operation video bit rate was change to rates ranging from 256Kb/s to 2048Kb/s for the test media content for an original bit rate of 3398Kb/s.

After the DASH content is generated, the fragmented/ non fragmented content can be provided to a LINUX based media player “MPlayer” version 1.0rc4-4.4.5. Presently only the latest release of Mplayer is known to play out fragmented content for DASH. The initialization segment has to be concatenated before providing the media segment to the Mplayer so that it has the initialization data as well.

![Content Generation Process](image)

Figure 34 - Content Generation Process

5.3 Results for Conformance Procedure

The results for the conformance procedures are not graphical or illustrative type. They are in the form of logs that are produced by the reference software to provide the details of compliance procedures. The main task is to parse the MPD and make relative relationship with the components to see if the rules defined apply as well. Then for the media stream, the reference software is used to investigate the bit stream to collect information needed to satisfy the file format conformance. In summary, the conformance procedures come down to a Yes or No result that are determined by analyzing the logs produced. For a better understanding of the logs produced the following figures
hold certain sample logs with captions and description of the information parsed and extracted as well as the conformance data base entries that are produced as a result.

<table>
<thead>
<tr>
<th>INFO:ConnectionManager:</th>
<th>Request: <a href="http://www.example.com/512K-5.3gp">http://www.example.com/512K-5.3gp</a> (649693-956774)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some Useful info :</td>
<td>Proposed _representation : 2 , Representation count :3 ,Total Segment count :21</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Received media data on stream 3</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Decoding track Id: 1, Base Data Offset: 0</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample Count: 119 Data Offset: 5136</td>
</tr>
<tr>
<td>THE REP NUMBER :</td>
<td>2 ,THE SEGMENT NUM :4 ,TRACK ID :1</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample[0]: Decode Time (secs) 73.0333 Composition Time (secs) 73.0333 Non Sync Flag = false [Sample Flags = 0x02000000] Sample Offset 5136 Sample Size 10037</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample[118]: Decode Time (secs) 77.0000 Composition Time (secs) 77.0000 Non Sync Flag = true [Sample Flags = 0x01010000] Sample Offset 271784 Sample Size 1246</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample Count: 1 Data Offset: 305353</td>
</tr>
<tr>
<td>THE REP NUMBER :</td>
<td>2 ,THE SEGMENT NUM :4 ,TRACK ID :1</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample[0]: Decode Time (secs) 77.0333 Composition Time (secs) 77.0333 Non Sync Flag = true [Sample Flags = 0x01010000] Sample Offset 305353 Sample Size 1426</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample Count: 1 Data Offset: 230530</td>
</tr>
<tr>
<td>THE REP NUMBER :</td>
<td>2 ,THE SEGMENT NUM :4 ,TRACK ID :2</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample[0]: Decode Time (secs) 73.0240 Composition Time (secs) 73.0240 Non Sync Flag = false [Sample Flags = 0x02000000] Sample Offset 273030 Sample Size 183</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample[186]: Decode Time (secs) 76.9920 Composition Time (secs) 76.9920 Non Sync Flag = false [Sample Flags = 0x02000000] Sample Offset 305124 Sample Size 229</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample Count: 2 Data Offset: 306779</td>
</tr>
<tr>
<td>THE REP NUMBER :</td>
<td>2 ,THE SEGMENT NUM :4 ,TRACK ID :2</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample[0]: Decode Time (secs) 77.0133 Composition Time (secs) 77.0133 Non Sync Flag = false [Sample Flags = 0x02000000] Sample Offset 306779 Sample Size 150</td>
</tr>
<tr>
<td>INFO:MediaPlayer:</td>
<td>Sample[1]: Decode Time (secs) 77.0347 Composition Time (secs) 77.0347 Non Sync Flag = false [Sample Flags = 0x02000000] Sample Offset 306929 Sample Size 153</td>
</tr>
</tbody>
</table>

**Figure 35 - Sub Segment Request Information**

From Figure 35, a request to a sub segment is made that is of Segment number 5 in representation number 2 of the MPD. The Media Player decoding of the TRUN box entries for Decode Time, Composition Time, Sync Samples and Sample Size can be seen from the figure.
Figure 36 – Initialization Segment & SIDX Checking

The initialization segment check and SIDX that belongs to ISOBMFF level is seen in figure 36. The Box structures are parsed to see if the required boxes are present and with the fields specified.

Figure 37 – Media Segment Checks

The initialization segment check and SIDX that belongs to ISOBMFF level is seen in figure 36. The Box structures are parsed to see if the required boxes are present and with the fields specified.
Media segments checks are performed in figure 37 by parsing the media segments. The combination of 'MOOF' and 'MDAT' is confirmed and the presence of 'TRAF' and 'TFDT' box is ensured for DASH processing. These checks will be applied to all media segments in the MPD.

| TRACK ID | Representation No. | Total Number of Segments | THE EPT of Segment No. | SEG EPT of SEG | SEG LPT of SEG | SEG EPT of SEG | SEG LPT of SEG | SEG EPT of SEG | SEG LPT of SEG | SEG EPT of SEG | SEG LPT of SEG | THE LPT of Segment No. | Tsap | SAP_TYPE |
|-----------|---------------------|--------------------------|------------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|-------------------------|------|----------|
| 0         | 1                   | 7                        | 0                      | 1             | 0             | 2             | 8             | 3             | 12.333        | 4             | 12.6667       | 16.2333                  | 16.2667 | 1        |
|           |                     |                          |                        |               |               |               |               |               |               |               |               |               |                         | 32.4333 | 1        |
|           |                     |                          |                        |               |               |               |               |               |               |               |               |               | 48.6667                  | 48.7  | 1        |
|           |                     |                          |                        |               |               |               |               |               |               |               |               |               | 64.9999                  | 64.9  | 1        |

KTH, Royal Institute of Technology 71
SSEG EPT of SSEG :1 ======= 64.9333
SSEG EPT of SSEG :2 ======= 69
SSEG EPT of SSEG :3 ======= 73.0333
SSEG EPT of SSEG :4 ======= 77.1
SSEG EPT of SSEG :5 ======= 81.1
THE LPT of Segment No. 5 : 81.1
Tsap : 64.9333 :: SAP_TYPE : 1

SSEG LPT of SSEG :1 ======= 68.9667
SSEG LPT of SSEG :2 ======= 73.0667
SSEG LPT of SSEG :3 ======= 77.1
SSEG LPT of SSEG :4 ======= 81.1
THE LPT of Segment No. 6 : 81.1
Tsap : 81.1333 :: SAP_TYPE : 1

SSEG EPT of SSEG :2 ======= 81.1333
SSEG EPT of SSEG :3 ======= 85.2
SSEG EPT of SSEG :4 ======= 89.2333
SSEG EPT of SSEG :5 ======= 89.2667
SSEG EPT of SSEG :6 ======= 93.3333
SSEG EPT of SSEG :7 ======= 97.3
THE LPT of Segment No. 6 : 97.3
Tsap : 97.3333 :: SAP_TYPE : 1

THE EPT of Segment No. 6: 81.1333
THE EPT of Segment No. 7: 97.3333
THE LPT of Segment No. 7 : 101.333
Tsap : 97.3333 :: SAP_TYPE : 1

TRACK ID :1 (Audio), Representation No. :1, Total Number of Segments : 7

THE EPT of Segment No. 1: 0
THE LPT of Segment No. 1 : 16.2347
Tsap : 0 :: SAP_TYPE : 1

THE EPT of Segment No. 2: 16.256
THE LPT of Segment No. 2 : 32.4267
Tsap : 0 :: SAP_TYPE : 1

THE EPT of Segment No. 3: 32.448
THE LPT of Segment No. 3 : 48.6613
Tsap : 0 :: SAP_TYPE : 1

THE EPT of Segment No. 4: 48.6827
THE LPT of Segment No. 4 : 64.896
Tsap : 0 :: SAP_TYPE : 1

THE EPT of Segment No. 5: 64.9173
THE LPT of Segment No. 5 : 81.1093
Tsap : 0 :: SAP_TYPE : 1

THE EPT of Segment No. 6: 81.1307
THE LPT of Segment No. 6 : 97.3013
Tsap : 0 :: SAP_TYPE : 1

THE EPT of Segment No. 7: 97.3227
THE LPT of Segment No. 7 : 101.333
Lastly the conformance data base entries are shown in figure 38. The point to be noted is that the Sub segment and Trap information is only present for Track ID 0. This is due to the fact that only the SIDX box for the reference track is available at this moment for only segment level information is present for other track ID’s. It can be seen earliest and latest presentation times of segment and sub segments are calculated as mentioned in the file format conformance section and then applied for implementation on the software. This yield the Data base values which are then compared to verify the compliance checks for DASH.
CHAPTER 6  FUTURE WORK & CONCLUSIONS

6.1 Future Work

Although the work documented here is an initiation to the conformance procedures to Dynamic Adaptive streaming over HTTP a lot of area needs to be covered as the standard develops and progresses. The following aspects for the future work can be marked as important:

- Presently the area of focus was the “On demand” profile for DASH. As the standard is improved with the passage of time rules and regulation are updated. Keeping this work as basic grounds further conformance processing can be built upon it. Then “Live” service pose a stern challenge in terms of conformance as the processing is much more complicated as compared to the “On demand” [7]. Additional factors and attributes associated with the “Live” service need to be incorporated in the DASH conformance procedures to see how a unified check system can be formulated.

- Since the first release of the DASH specification amendments have been made to the DASH structures in terms of addition of new elements and attribute and omission of few existing ones as well. So, the DASH processing needs to be in sync with the new specifications so that upgrades can be a part of the conformance procedure as well. Like the inclusion of
  - Adaptation Set
  - Sub Set (Sub- Element – Period)
  - Metrics Element
  - Content Descriptor Element,
  - Sub Representations (Sub- Element – Representation)
  - Accessibility, Rating (Sub- Element – Group)
  - maxPlayRate, codingDependency (Attributes – Group)

These new additions are to be included in the software implementation of DASH processing so that the conformance procedure that are formed on the corresponding levels can make use of these attributes.

- As the DASH is now being implemented widely at the software domain, the concepts in DASH standard need to be more open and clear. Conformance procedure for further DASH processing will provide a platform that will aid in clarifying DASH specification in term of practical implementations. Further refining of aspects and rules will take place with more amendments in time.

- Integration of the conformance process itself would also be included as a next step to the process. The MPD conformance and file format conformance should be integrated as one unit to provide a single entity for checks.

- At present, primarily the work is done on the ISOBMFF. Although some of the aspects of ISOBMFF can be translated to MPEG 2 TS as well but a proper compliance procedure of it should also be defined. Software implementation aspects for MPEG 2 TS should be supported by the integrated compliance software.
6.2 CONCLUSION

The compliance procedure provides a very significant aspect to the DASH working. As the DASH standard is being developed the compliance will provide the implementation guidelines. The interaction between the client and server can be made in a way that intelligence is shifted towards the client for more efficient working. Due to the existing technologies, DASH stands out as a solution that provides features for best user experience. To maintain this level, conformance of DASH client and server interaction as well as the media content used in streaming forms an essential part for the process. Without compliance the feature of DASH will get corrupted and every client server interaction will differ thus reverting back to the same shortcoming as before.

Compliance makes it possible to have a tight grip on the video streaming end to end process. That includes even the content generated at the server. The content should in turn validate the information provided in the manifest files. DASH content should have certain parameters that impart features for client to make informed and swift decisions. Clarification of the concepts involved in terms of implementation details is also essential so that the desired target can be accomplished.

My thesis work made contribution to a working draft of the DASH standardization that will be included in a committee draft of MPEG. This will aid in the upgrade and reformation of the DASH standard itself.

As work goes on in this direction, it will lead DASH to further development making it simple and wide spread.
References

[1]. Christian Timmerer and Christian Muller, Alpen–Adria Universitat Klagenfurt
Presentation on “Dynamic Adaptive Streaming over HTTP (DASH)”
http://www.slideshare.net/christian.timmerer/dynamic-adaptive-streaming-over-http-dash

[2]. Thomas Stockhammer, Qualcomm Incorporated c/o Nomor Research
Paper on “3GPP Dynamic Adaptive Streaming over HTTP – Standard and Design Principles”
Published in MMSys ’11 proceeding of second annual ACM conference on Multimedia
Systems, ACM New York NY, USA 2011, Pages 133-144

[3]. Velibor Adzic, Hari Kalva, Borko Furht Dept. of Electrical & Computer Engineering and
Computer Science Florida Atlantic University Boca Raton, FL,
Paper on “A Survey of multimédia content adaptation for mobile devices”
Published in Journal “Multimedia Tools and Applications”, Volume 51, issue 1, January 2011,
Pages: 379-396.

over HTTP (DASH)
ISO /IEC JTC 1/SC 29, Dated 2011-01-28
ISO /IEC FCD 23001-6

[5]. Truong Cong Thang, Jin Young Lee, Jung Won Kang, Seong-Jun Bae, Sang Taick Park
Broadcasting & Telecom Convergence Research Laboratory, Electronics and
Telecommunications Research Institute (ETRI), Korea
Paper on “Signaling Metadata for Adaptive HTTP Streaming”
Systems 2011 February 23-25, 2011 San Jose, California

Aspects.
Transparent end-to-end Packet-switched Streaming Service (PSS);
Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) (Release 10)

[7]. Thorsten Lohmar Markus & Kampmann - Ericsson GmbH; Per Fröjdh & Torbjörn Einarsson -
Ericsson AB; Frédéric Gabin - Ericsson France
Paper on “Dynamic adaptive HTTP Streaming of Live Content”
Presented at 12th IEEE International Symposium on a World of Wireless, Mobile and
Multimedia Networks, WoWMoM 2011 at the IMT Institute for Advanced Studies Lucca,
central Italy, on June 20-23, 2011.

[8]. Information Technology- Coding of audio-visual Objects – Part 12 : ISO base media file
format, Amendment 3 : DASH support and RTP reception hint track processing
ISO/IEC JTC 1/SC 29, Dated 2011-01-28


[13]. XML Schema According to W3C specification www.w3.org/2001/XMLSchema


[16]. XSLT Tutorials – XSLT Basics http://www.w3schools.com/xsl/

[17]. Xpath Tutorials – Xpath Basics http://www.w3schools.com/xpath/


[21]. Cyril Concolato, Jean Le Feuvre, Romain Bouqueau
Telecom Paris Tech; Institute Telecom; CNRS LTCI
Paper on “Usage of DASH for Rich Media Services “
In Multimedia Systems Conference, San Jose, USA, February 2011
http://biblio.telecom-paristech.fr/cgi-bin/download.cgi?id=11064

[22]. Thomas Stockhammer, Qualcomm Incorporated
DASH: Segment and Subsegment Alignment
Contribution paper to INTERNATIONAL ORGANIZATION FOR STANDARDIZATION FOR
CODING OF MOVING PICTURES AND AUDIO
ISO/IEC JTC1/SC29/WG11, MPEG2010/M20376, July 2011, Torino, Italy

[23]. Advanced H.264 Encoding with Carbon Coder
Application Notes: Carbon Coder H.264 Encoding Guide
ROHZET, A Business Unit of Harmonic Inc.
APPENDIX A: SCHEMATRON SCHEMA

<?xml version="1.0" encoding="UTF-8"?>
<schema
xmlns="http://purl.oclc.org/dsdl/schematron"
xmlns:dash="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
queryBinding='xslt'
schemaVersion='ISO19757-3'>
<ns prefix="dash" uri="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"/>
<ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
<ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>
<title>Schema for validating DASH descriptions</title>

<pattern name="MPD element">
  <!-- R1.*: Check the conformance of MPD -->
  <rule context="dash:MPD">
    <!-- R1.0 -->
    <assert test="if (@type = 'Live' and not(@availabilityStartTime)) then false() else true()">If MPD is of type "Live" availabilityStartTime shall be defined.</assert>
    <!-- R1.1 -->
    <assert test="if (not(@minBufferTime) and not(child::dash:Period/@minBufferTime)) then false() else true()">If MPD has no attribute minBufferTime each Period element shall have this attribute.</assert>
    <!-- R1.2 -->
    <assert test="if (@type = 'OnDemand' and @timeShiftBufferDepth) then false() else true()">If MPD is of type "OnDemand" timeShiftBufferDepth shall not be defined.</assert>
  </rule>
</pattern>

<pattern name="Period element">
  <!-- R2.*: Check the conformance of Period -->
  <rule context="dash:Period">
    <!-- R2.0 -->
    <assert test="if (not(@start) and not(preceding-sibling::dash:Period[1]/@duration)) then false() else true()">If the start attribute is not defined for a Period the previous Period element shall contain the duration attribute.</assert>
    <!-- R2.1 -->
    <assert test="if (string(@bitStreamSwitchingFlag) = 'true' and string(@segmentAlignmentFlag) = 'false') then false() else true()">BitstreamSwitchingFlag shall not be set to true if segmentAlignmentFlag is set to false.</assert>
  </rule>
</pattern>

<pattern name="Representation element">
  <!-- R3.*: Check the conformance of Representation -->
  <rule context="dash:Representation">
    <!-- R3.0 -->
    <assert test="if (@id = preceding::dash:Representation/@id) then false() else true()">The id of each Representation shall be unique.</assert>
  </rule>
</pattern>
<pattern name="SegmentInfo element">
  <!-- R4.*: Check the conformance of SegmentInfo -->
  <rule context="dash:SegmentInfo">
    <!-- R4.0 -->
    <assert test="if (not(@duration) and not(child::dash:SegmentTimeline)) then if (count(child::dash:Url) > 1) then false() else true() else true()">If neither duration attribute nor SegmentTimeline element is present, then the Representation shall contain exactly one Media Segment.</assert>
    <!-- R4.1 -->
    <assert test="if (ancestor::dash:MPD/@type = 'OnDemand' and @startIndex != 1) then false() else true()">StartIndex shall not be present or set to 1.</assert>
  </rule>
</pattern>
APPENDIX B: DASH SCHEMA

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns="urn:mpeg:mpegB:schema:DASH:MPD:DIS2011">
  <xs:import namespace="http://www.w3.org/1999/xlink"
    schemaloctation="http://www.w3.org/1999/xlink.xsd"/>
  <xs:annotation>
    <xs:appinfo>Media Presentation Description</xs:appinfo>
    <xs:documentation xml:lang="en">
      This Schema defines Media Presentation Description for MPEG DASH.
      The namespace identifier is for this draft version of the specification.
      The final version will use a different namespace identifier to align with 3GPP.
      However, changes from the draft to the final version are not expected to change the information model, even if the schema changes, and trial implementation and comment on the schema presented here are both encouraged.
    </xs:documentation>
  </xs:annotation>
  <!-- MPD: main element -->
  <xs:element name="MPD" type="MPDtype"/>
  <!-- MPD Type -->
  <xs:complexType name="MPDtype">
    <xs:sequence>
      <xs:element name="ProgramInformation" type="ProgramInformationType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
      <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="#other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="profiles" type="URIVectorType"/>
    <xs:attribute name="presentationType" type="PresentationType" default="OnDemand"/>
    <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
    <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
    <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
    <xs:attribute name="minimumUpdatePeriodMPD" type="xs:duration"/>
    <xs:attribute name="minBufferTime" type="xs:duration"/>
    <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
    <xs:attribute name="maxSegmentDuration" type="xs:duration"/>
    <xs:attribute name="maxSubsegmentDuration" type="xs:duration"/>
    <xs:anyAttribute namespace="#other" processContents="lax"/>
  </xs:complexType>
  <!-- Type of presentation - live or on-demand -->
  <xs:simpleType name="PresentationType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="OnDemand"/>
      <xs:enumeration value="Live"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- Program information for a presentation -->
  <xs:complexType name="ProgramInformationType">
    <xs:sequence>
      <xs:element name="Title" type="xs:string" maxOccurs="0" minOccurs="0"/>
      <xs:element name="Source" type="xs:string" maxOccurs="0" minOccurs="0"/>
      <xs:element name="Copyright" type="xs:string" maxOccurs="0" minOccurs="0"/>
      <xs:any namespace="#other" processContents="lax" maxOccurs="0" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
<xs:complexType name="RepresentationBaseType">
    <xs:sequence>
        <xs:element name="ContentProtection" type="ContentDescriptorType" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element name="Accessibility" type="ContentDescriptorType" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element name="Rating" type="ContentDescriptorType" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element name="Viewpoint" type="ContentDescriptorType" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element name="MultipleViews" type="MultipleViewsType" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="group" type="xs:unsignedInt" />
    <xs:attribute name="width" type="xs:unsignedInt" />
    <xs:attribute name="height" type="xs:unsignedInt" />
    <xs:attribute name="parx" type="xs:unsignedInt" />
    <xs:attribute name="pary" type="xs:unsignedInt" />
    <xs:attribute name="lang" type="LanguageVectorType" />
    <xs:attribute name="mimeType" type="xs:string" />
    <xs:attribute name="codecs" type="xs:string" />
    <xs:attribute name="startWithRAP" type="xs:boolean" />
    <xs:attribute name="frameRate" type="xs:double" />
    <xs:attribute name="maximumRAPPeriod" type="xs:double" />
    <xs:attribute name="numberOfChannels" type="StringVectorType" />
    <xs:attribute name="samplingRate" type="StringVectorType" />
    <xs:attribute name="maxLayoutRate" type="xs:double" />
    <xs:attribute name="codingRate" type="xs:boolean" />
    <xs:attribute name="moreInformationURL" type="xs:anyURI" />
</xs:complexType>

<!-- Representation of the presentation content for a specific Period; extends RepresentationBaseType -->
<xs:complexType name="RepresentationType">
    <xs:complexContent>
        <xs:extension base="RepresentationBaseType">
            <xs:sequence>
</xs:sequence>
</xs:complexType>
Master of Science Thesis

<xsl:element name="SubRepresentationType" type="SubRepresentationType"

type="SubRepresentationType"

minOccurs="0" maxOccurs="unbounded"/>

<xsl:element name="SegmentInfo" type="SegmentInfoType"/>

<xsl:sequence>

</xsl:sequence>

<xsl:attribute name="id" type="xs:string" use="required"/>

<xsl:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>

<xsl:attribute name="qualityRanking" type="xs:unsignedInt"/>

<xsl:attribute name="dependencyId" type="StringVectorType"/>

<xsl:attribute name="bitstreamStructureId" type="StringVectorType"/>

</xsl:extension>
</xsl:complexType>

</xsl:complexType>

</xsl:element name="SegmentInfoType" minOccurs="0" maxOccurs="unbounded"/>

<xsl:element name="SegmentInfoDefault" type="SegmentInfoDefaultType" minOccurs="0"/>

<xsl:sequence>

</xsl:sequence>

<xsl:attribute ref="xlink:href"/>

<xsl:attribute ref="xlink:actuate" default="onRequest"/>

<xsl:attribute name="minBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkWidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxWidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkHeight" type="xs:unsignedInt"/>

<xsl:attribute name="maxHeight" type="xs:unsignedInt"/>

<xsl:attribute name="minFrameRate" type="xs:double"/>

<xsl:attribute name="maxFrameRate" type="xs:double"/>

<xsl:attribute name="subsegmentAlignment" type="x::boolean"/>

<xsl:attribute name="segmentAlignmentFlag" type="xs:boolean"/>

<xsl:attribute name="bitstreamSwitchingFlag" type="xs:boolean"/>

<xsl:attribute name="dependencyId" type="StringVectorType"/>

<xsl:attribute name="dependency" type="x:unsignedInt"/>

<xsl:attribute name="bandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkWidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxWidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkHeight" type="xs:unsignedInt"/>

<xsl:attribute name="maxHeight" type="xs:unsignedInt"/>

<xsl:attribute name="minFrameRate" type="xs:double"/>

<xsl:attribute name="maxFrameRate" type="xs:double"/>

<xsl:attribute name="subsegmentAlignment" type="xs:boolean"/>

<xsl:attribute name="segmentAlignmentFlag" type="xs:boolean"/>

<xsl:attribute name="bitstreamSwitchingFlag" type="xs:boolean"/>

<xsl:attribute name="dependencyId" type="StringVectorType"/>

<xsl:attribute name="dependency" type="x:unsignedInt"/>

<xsl:attribute name="bandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkWidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxWidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkHeight" type="xs:unsignedInt"/>

<xsl:attribute name="maxHeight" type="xs:unsignedInt"/>

<xsl:attribute name="minFrameRate" type="xs:double"/>

<xsl:attribute name="maxFrameRate" type="xs:double"/>

<xsl:attribute name="subsegmentAlignment" type="xs:boolean"/>

<xsl:attribute name="segmentAlignmentFlag" type="xs:boolean"/>

<xsl:attribute name="bitstreamSwitchingFlag" type="xs:boolean"/>

<xsl:attribute name="dependencyId" type="StringVectorType"/>

<xsl:attribute name="dependency" type="x:unsignedInt"/>

<xsl:attribute name="bandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkWidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxWidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkHeight" type="xs:unsignedInt"/>

<xsl:attribute name="maxHeight" type="xs:unsignedInt"/>

<xsl:attribute name="minFrameRate" type="xs:double"/>

<xsl:attribute name="maxFrameRate" type="xs:double"/>

<xsl:attribute name="subsegmentAlignment" type="xs:boolean"/>

<xsl:attribute name="segmentAlignmentFlag" type="xs:boolean"/>

<xsl:attribute name="bitstreamSwitchingFlag" type="xs:boolean"/>

<xsl:attribute name="dependencyId" type="StringVectorType"/>

<xsl:attribute name="dependency" type="x:unsignedInt"/>

<xsl:attribute name="bandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkWidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxWidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkHeight" type="xs:unsignedInt"/>

<xsl:attribute name="maxHeight" type="xs:unsignedInt"/>

<xsl:attribute name="minFrameRate" type="xs:double"/>

<xsl:attribute name="maxFrameRate" type="xs:double"/>

<xsl:attribute name="subsegmentAlignment" type="xs:boolean"/>

<xsl:attribute name="segmentAlignmentFlag" type="xs:boolean"/>

<xsl:attribute name="bitstreamSwitchingFlag" type="xs:boolean"/>

<xsl:attribute name="dependencyId" type="StringVectorType"/>

<xsl:attribute name="dependency" type="x:unsignedInt"/>

<xsl:attribute name="bandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="maxBandwidth" type="xs:unsignedInt"/>

<xsl:attribute name="minkWidth" type="xs:unsignedInt"/>
<xs:sequence>
  <xs:element name="Url" type="UrlType" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="Index" type="UrlType" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute ref="xlink:href"/>
<xs:attribute ref="xlink:actuate" default="onRequest"/>
<xs:attribute name="startIndex" type="xsd:unsignedInt"/>
</xs:complexType>

<xs:complexType name="ContentDescriptorType">
  <xs:sequence>
    <xs:element minOccurs="0" name="SchemeInformation" type="xsd:string" maxOccurs="unbounded"/>
    <xs:attribute name="SchemeIdUri" type="xsd:anyURI" use="required"/>
    <xs:attribute name="#other" processContents="lax"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MultipleViewsType">
  <xs:element name="FramePacking" type="ContentDescriptorType" minOccurs="0" maxOccurs="unbounded"/>
  <xs:attribute name="stereoId" type="StringVectorType"/>
  <xs:attribute name="maximumViews" type="xsd:unsignedInt" default="1"/>
  <xs:attribute name="#other" processContents="lax"/>
</xs:complexType>

<xs:simpleType name="StringVectorType">
  <xs:list itemType="xsd:string"/>
</xs:simpleType>

<xs:simpleType name="UIntVectorType">
  <xs:list itemType="xsd:unsignedInt"/>
</xs:simpleType>

<xs:simpleType name="URIVectorType">
  <xs:list itemType="xsd:anyURI"/>
</xs:simpleType>

<xs:simpleType name="LangVectorType">
  <xs:list itemType="xsd:language"/>
</xs:simpleType>

<xs:complexType name="BaseURLType">
  <xs:simpleContent>
    <xs:extension base="xsd:anyURI">
      <xs:attribute name="#other" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>