Conformance Software

Nomor Research GmbH
Munich, Germany
info@nomor.de

17 June 2020

Nomor Research GmbH
Brecherspitzstraße 8
81541 München

Table of Contents

Table of Figures

1 Introduction

DASH Conformance Tool is used to validate DASH content according to DASH-related media specifications. It aims to give information about the validity of the content against one or more developed media standards. Consequently, it reports on (un)expected behaviour that can be observed in provided media services that are aimed to be working in alignment to these standards. In this way, the tool provides a testing environment useful for content providers, service providers, media players, etc.

The development of the conformance tool started in 2012 and has been continuously updated with newer versions of the already supported standards and/or new standards as required. The DASH Conformance Tool is an open source software available on GitHub at https://github.com/Dash-Industry-Forum/DASH-IF-Conformance. This tool is also hosted by DASH-IF at https://conformance.dashif.org/.

The tool is aligned with a large set of specifications [1-7], namely

- MPEG-DASH
- ISO BMFF
- DASH-IF IOP
- Low Latency DASH
- CMAF
- DVB-DASH
- HbbTV
- CTA WAVE

It also integrates file format header level parsing of media codecs, including AVC, HEVC, AAC, HE-AAC, HE-AACv2, AC-3, AC-4, E-AC-3, WebVTT and TTML.

For each corresponding specification, the scope of the validation covers:

1. *Media Presentation Description (MPD) validation* where the MPD is checked if it is a well-formed file, appropriate according to DASH schema and MPD-level signalling is done correctly,
2. *Segment validation* where the media content pointed to by the MPD is validated at container level,
3. *Cross validation* of the MPD-level elements and attributes as well as of the media content(s) signalled at the same hierarchy.

The validation procedure can be completed in three steps:

1. *Provide MPD:* The user inputs an MPD either as a file or as a URL
2. *Select Profile:* By default, testing is done against the profiles signalled in the MPD but the user can optionally select any additional supported specifications from the interface
3. *Start testing:* The test results are progressively displayed in the interface in a nice structure for the user to analyse in real time.

The validation also includes the ability to validate a dynamic service. For live services the segments get available over time as the content is produced; and therefore, the precision of segment availability time signalling in the MPD becomes important for the media presentation. This signalling is used by the client to find out if the desired media segment is available before actually requesting it via the signalled segment location.

Dynamic Service Validator is developed to provide a testing software tool that validates the segment time signalling for the live services. It is available on GitHub at https://github.com/Dash-Industry-Forum/DynamicServiceValidator. This tool is also hosted by DASH-IF at http://vm1.dashif.org/DynamicServiceValidator.

The validator checks if the segments are (un)available at their corresponding signalled time by sending HTTP requests. The tool also features DASH-IF IOP guidelines to provide clock synchronization between server and client through UTC timing, round-trip time (RTT) and clock skew features. UTC timing information in the MPD is used to synchronize the client to wall-clock time. Apart from this, the tool gives information on the mean RTT for segment requests. By using this, the user can specify an RTT to account for the possible delays that can accumulate over time and cause misinformation on the client side about segment's availability on the server. Furthermore, problems such as real-time process load at the server and server applications not running on real-time operating systems can lead to inaccurate clock synchronization. Worst case this leads to wrong calculations of the segment availability times. Therefore, the tool integrates an optional clock skew set by the user providing a margin for the segment timing computation.

The conformance tool also supports HLS conformance only in the scope of segment validation and cross validation, i.e. the validation of the manifest is not supported. It is available on GitHub https://github.com/Dash-Industry-Forum/HLS/ and https://github.com/Dash-Industry-Forum/Conformance-Frontend-HLS/. It provides a separate user interface and reports on the media content conformance.

This document provides information on the source code design of mentioned conformance software. This is envisaged to be an introduction on the conformance software for the contributors. In this regard, Section 2-4 explains the framework and process flow for DASH conformance, HLS Conformance and Dynamic Service Validator, respectively.

2 DASH Conformance Framework

The conformance software comprises of different modules for each extension that is together hosted in the main part https://github.com/Dash-Industry-Forum/DASH-IF-Conformance. The different modules are:

a. Conformance-Frontend:
https://github.com/Dash-Industry-Forum/Conformance-Frontend/
b. DASH:
https://github.com/Dash-Industry-Forum/DASH/
c. ISOSegmentValidator:
https://github.com/Dash-Industry-Forum/ISOSegmentValidator/
d. CMAF:
https://github.com/Dash-Industry-Forum/CMAF/
e. CTAWAVE:
https://github.com/Dash-Industry-Forum/CTAWAVE/
f. HbbTV_DVB:
https://github.com/Dash-Industry-Forum/HbbTV_DVB/
g. DynamicServiceValidator:
https://github.com/Dash-Industry-Forum/DynamicServiceValidator/

(a), (b) and (c) are the core modules for the validation software. (a) contains the source code of the web interface. (b) is the core MPEG-DASH related validation module and is common to all the other extensions. (c) is the core ISOBMFF implementation for segment validation. (d), (e), (f) are the extensions to CMAF, CTA WAVE, HbbTV and DVB-DASH specifications. (e) was later on modularly merged into (d) as CTA WAVE builds on CMAF. Similarly, DASH-IF IOP and Low Latency DASH extensions reside in (b). (g) hosts the source code of the aforementioned Dynamic Service Validator tool.

The functional diagram of the conformance software and modules structure is provided in Figure 1. This diagram contains individual modules, the scripts each module contains, and their connection to each other. The software includes the implementation of “Client” and “Conformance Server” shown in the diagram.

The orange rectangles define the different modules with specific functionalities. Detailed explanation on each module is provided in the remainder of this section.

2.1 Core Modules

2.1.1 Conformance-Frontend

This module is the client UI for DASH manifests, containing all the UI-related functionalities to be provided to the user.

During conformance testing, the users can interact with the UI to obtain information on the progress, and intermediate results. For each conformance testing, a different session is created, i.e. separated by session folders. Each session folder contains the content and the reports generated for the corresponding test. The storage of session folders resides here as well.

This module uses AngularJS, AngularJS Material, HTML and PHP. “index.html” is the main landing page in this module that provides information on the scope of the validation to users.

“app/styles” folders contain the controller scripts and the validation pages that are responsible for initiating the conformance server processes and updating the client on the overall progress and the conformance results. The controller “startController.js” controls the common components that exit on all the pages, such as navigation pane and headline images. The controller “stepperController.js” controls the input providing page and the result providing page. It initially interacts with “Process.php” in “Utils” block as soon as RUN button is clicked on. After the conformance check process starts, the intermediate and final results are sent to the frontend module for display. These results contain:

- Input summary
- Content Information
- MPD validation results
- Segment validation checks
- Cross validation checks

“Featurelist.php” script is responsible for generating the MPD features to be sent to the validation page for display. “TabulateResults.php” is a script that converts the results logs generated by the conformance server to a tabulated form. “Estimate.php” is a script that provides information on minimum buffer time and bandwidth based on the information provided in each representation.

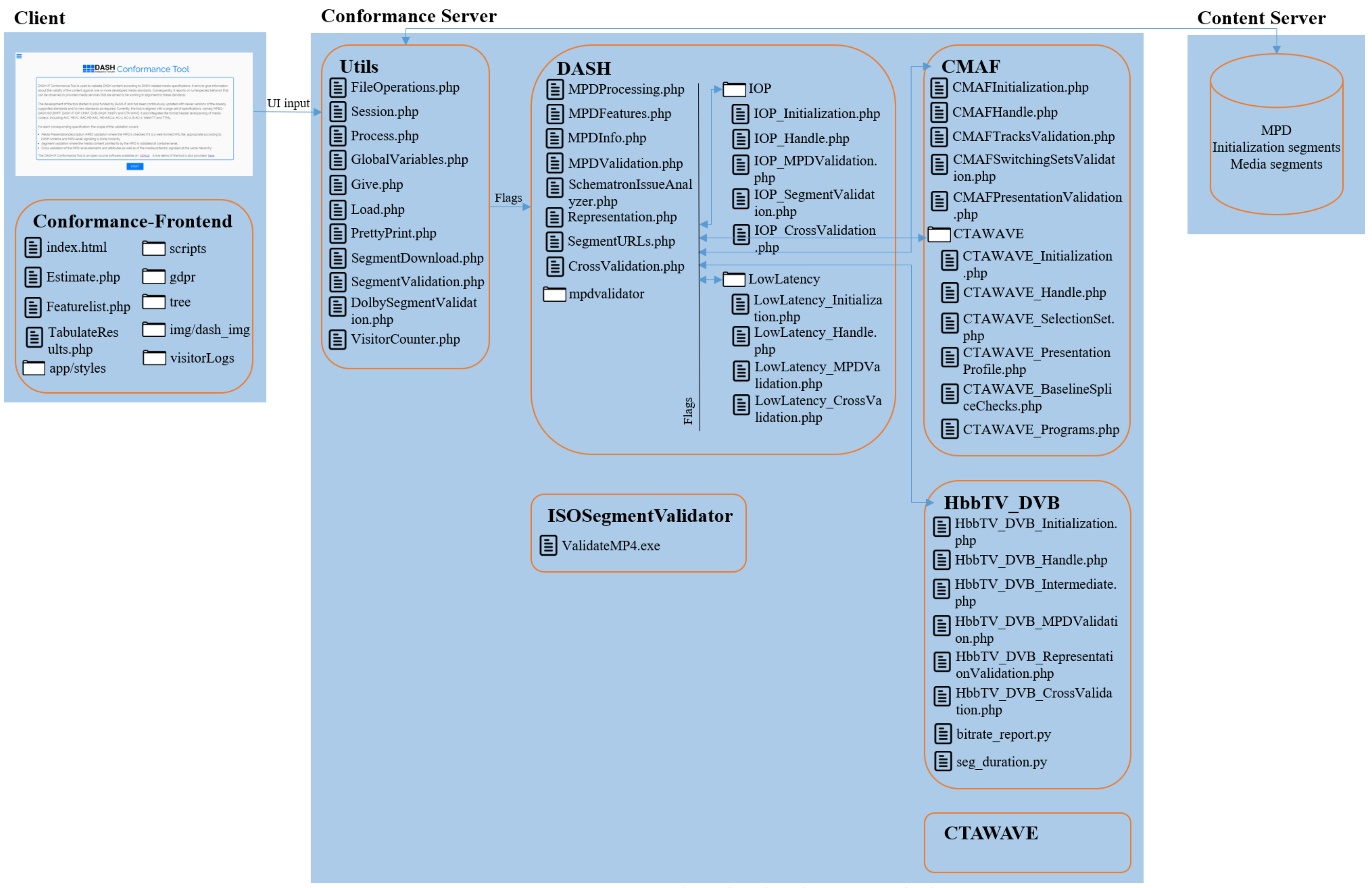

Figure 1: Functional diagram of DASH Conformance Software.

2.1.2 Utils

This block is the closest point to the frontend module. It contains useful functionalities that will be needed for DASH and the extensions (DASH-IF IOP, CMAF, HbbTV, DVB, and CTA WAVE).

“Process.php” script is the initial script to be called from the frontend submodule when the conformance testing is requested to be performed. It initiates the session creation (temporary storage folder) in the conformance frontend module for the individual request by calling “Session.php” in this block. It also initializes the related global parameters according to the user input in “GlobalVariables.php”. After this, it starts the conformance testing by calling “MPDProcessing.php” in DASH module.

“FileOperations.php” provides some useful helper functions regarding file operations, such as creating, deleting, opening, closing and renaming a file, etc. “Load.php” script is used to load provided any well-formed file into a DOM XML structure. “VisitorCounter.php” stores the session information for debugging and statistics collecting purposes.

“PrettyPrint.php” is used when the conformance software is used via command line rather than from the user interface (“Conformance-Frontend”) to print the result logs in a nicely-structured way onto console.

“SegmentDownload.php” is used when Segment Validation is performed to download the signalled initialization and media segments in each Representation in the provided MPD from the user input. “SegmentValidation.php” is executed to perform the core ISOBMFF parsing and validation of the downloaded segments in each Representation by calling “ValidateMP4.exe” in ISOSegmentValidator module.

2.1.3 DASH

This module contains all the core DASH conformance functionalities, such as parsing MPD, MPD validation, extracting segment(s) URL(s), DASH segment validation, etc. It also contains the DASH-IF IOP and Low Latency DASH modules.

“MPDProcessing.php” script is like the “int main()” function of the DASH conformance and the other extensions’ conformances. The main function in this script is process_MPD() which is called from “Process.php” in Utils block. This function in itself calls the functionalities that:

- Load the provided MPD (“Load.php” in Utils block),
- Parse it (“MPDFeatures.php” in this module),
- Perform the MPD validation (“MPDValidation.php” and “mpdvalidator” folder in this module),
- Compute the segment URLs to be downloaded (“SegmentURLs.php” in this module),
- Perform segment validation (“SegmentValidation.php” and “Representation” in Utils block) and
- Perform cross validation (“CrossValidation.php” in this module).

This script also calls the MPD, segment and cross validation functionalities in the supported extensions that are included by user input.

“MPDFeatures.php” is responsible for loading the MPD XML DOM structure to an array for making the code cleaner. “MPDInfo.php” extracts MPD information such as the determining the current period, determining the start times and durations of each period in the MPD, time parsing of the MPD attributes, determining the available segments for dynamic MPDs, and determining the profiles signalled in the MPD, etc.

“MPDValidation.php” is called for MPD validation including DASH MPD conformance and schema and schematron validation. These are done via the programs in “mpdvalidator” folder. “SchematronIssuesAnalyzer.php” is used to determine the schematron issues and report them to the frontend submodule for display, just as in the original conformance software. This part overall generates MPD validation report that is reported to frontend module or printed onto console for display.

“SegmentURLs.php” computes the segment(s) URL(s) for each representation for each adaptation set in each period from provided MPD. This segment access information corresponds to parsing the information provided in the MPD at any level through BaseURL, SegmentBase, SegmentTemplate, SegmentTimeline elements. It should be noted that SegmentList element is not supported in the conformance software. “Representation.php” generates the flags according to the profiles from the user input to be passed to the segment validation executable for each representation in ISOSegmentValidator module. This part overall generates segment validation report that is reported to frontend module or printed onto console for display.

“CrossValidation.php” script contains the validation points for MPD-level elements and attributes as well as of the media content(s) signalled at the same hierarchy. This part overall generates cross validation reports for each hierarchy that are reported to frontend module or printed onto console for display.

2.1.4 ISOSegmentValidator

This module contains the core ISOBMFF and segment validation software at the header level, meaning that elementary stream validation is out of the scope of this module. This module is responsible for the atom/box validation of the ISOBMFF packaged media content. It is called by the Utils block to validate the contents of the downloaded segments. After the execution, it provides information and error log files to be interpreted and reported by the DASH module to the user.

It should be noted that only the Linux environment executable of the ISOSegmentValidator is used and supported, support for Windows or MacOS is not provided.

2.2 Extension Modules

Extension modules perform validation checks against the corresponding extension specifications. They provide their specific MPD, segment validation results in the same MPD and segment validation log files as in DASH module whereas they provide cross validation results in their specific log files. The extension modules as mentioned before are:

- DASH-IF IOP with the folder name “IOP” under DASH module
- Low Latency DASH with the folder name “LowLatency” under DASH module
- CMAF with the folder name “CMAF” which is also a module
- CTA WAVE with the folder name “CTAWAVE” under CMAF module
- HbbTV and DV-DASH with the folder name “HbbTV_DVB” which is also a module

Each extension module is constructed in the same way to provide a common way to modify and existing extension and/or add a new extension. This can be listed as the following:

- *Initialization:* This file is used to include the rest of the PHP files in corresponding extension folder that this file also resides in. This way, the design remains modular and the extensions are treated as black box. This script also initializes the extension-related global variables.
- *Handle:* This script provides sort of an interface between the core modules and functions within an extension.
- *MPD Validation:* This script implements and performs the MPD-level validation checks for specific extension. Please note that this script does not exist in all the extensions as some extension do not cover MPD-level requirements, such as CMAF and CTA WAVE.
- *Segment Validation:* This script implements and performs Representation- and segment-level checks for specific extension.
- *Cross Validation:* This script implements and performs validation of MPD-level elements and attributes as well as of the media content(s) signalled at the same hierarchy for specific extension. This covers the validation points in corresponding specifications regarding:
    - Cross-representation, meaning the checks that puts constraints on all Representations within an Adaptation Set
    - Cross-adaptation set, meaning the checks that puts constraints on all Adaptation Sets within a Period
    - Cross-period, meaning the checks that puts constraints on all Periods within an MPD

  Please note that there can be more than one script in this category to separate the validation implementation of different hierarchies.

Each extension validation process is enabled by user input. This input can either be clicking the corresponding extension profile from the "Include additional tests" part in the frontend module or by including an MPD with @profiles attribute including the extension profile. When either one is the case, "GlobalVariables.php" in Utils block in the core modules includes the Initialization script of the corresponding extension which in turn includes all the rest of the scripts within the extension. When "MPDProcessing.php" goes over each validation scope (MPD, Segment, Cross), it also calls the Handle script of the extension to enable the validation of the same scope against the implemented extension. Depending on which validation scope is called by the Handle script, the script of that scope is run.

It is also important to note that as newer versions of extension specifications are published and there is a request for the software to be extended accordingly, the software is updated with the modifications/additions/deletions of validation points in the conformance software. This means that there is no backwards compatibility with older specifications or version-controlled validation in the software.

2.2.1 IOP

This extension module implements the DASH-IF IOP specification. Currently version 4.3 is supported in the conformance software. Whenever "DASH-IF IOP" option is clicked from the frontend module or any DASH-IF IOP profile string is included in the MPD@profiles string, this module is added to the overall validation process.

The mapping of the scripts to the common extension structure is as follows:

- “IOP_Initialization.php” corresponds to Initialization script
- “IOP_Handle.php” corresponds to Handle script
- “IOP_MPDValidation.php” corresponds to MPD Validation script
- “IOP_SegmentValidation.php” corresponds to Segment Validation script
- “IOP_CrossValidation.php” corresponds to Cross Validation script

2.2.2 LowLatency

This extension module implements the Low Latency DASH Change Request. Currently the community review version of this is supported in the conformance software. Whenever “LL DASH-IF” option is clicked from the frontend module or low latency DASH profile string is included in the MPD@profiles string, this module is added to the overall validation process.

The mapping of the scripts to the common extension structure is as follows:

- “LowLatency_Initialization.php” corresponds to Initialization script
- “LowLatency_Handle.php” corresponds to Handle script
- “LowLatency_MPDValidation.php” corresponds to MPD Validation script
- “LowLatency_CrossValidation.php” corresponds to Cross Validation script

2.2.3 CMAF

This extension module implements the CMAF specification. Currently ISO/IEC 23000-19 2nd Edition is supported in the conformance software. Whenever “CMAF” option is clicked from the frontend module, this module is added to the overall validation process.

The mapping of the scripts to the common extension structure is as follows:

- “CMAFInitialization.php” corresponds to Initialization script
- “CMAFHandle.php” corresponds to Handle script
- “CMAFTracksValidation.php” corresponds to Segment Validation script
- “CMAFSwitchingSetsValidation.php” and “CMAFPressentationValidation.php” correspond to Cross Validation script

2.2.4 CTA WAVE

This extension module implements the CTA WAVE specification. Currently CTA-5001 is supported in the conformance software. Whenever “CTA-WAVE” option is clicked from the frontend module, this module together with CMAF module is added to the overall validation process. This is becase CTA WAVE builds on CMAF specification.

The mapping of the scripts to the common extension structure is as follows:

- “CMAFInitialization.php” corresponds to Initialization script
- “CMAFHandle.php” corresponds to Handle script
- “CMAFTracksValidation.php” corresponds to Segment Validation script
- “CMAFSwitchingSetsValidation.php” and “CMAFPressentationValidation.php” correspond to Cross Validation script

2.2.5 HbbTV_DVB

This extension module implements the HbbTV and DVB-DASH specifications. Currently HbbTV 1.5, ETSI TS 102 796 v1.4.1 and ETSI TS 103 285 v1.1.1 4.3 are supported in the conformance software. Whenever “HbbTV” option is clicked from the frontend module or HbbTV profile string is included in the MPD@profiles string, this module is added to the overall validation process and only HbbTV-related parts in this module is used. Whenever “DVB (2019)” or “DVB (2018)” – as the two options are mutually exclusive – is clicked from the frontend module or HbbTV profile string is included in the MPD@profiles string, this module is added to the overall validation process and only DVB-related parts in this module is used. When both are clicked or both profiles are included in the MPD@profiles string, then this module is again enabled and all validation points are used.

The mapping of the scripts to the common extension structure is as follows:

- “HbbTV_DVB_Initialization.php” corresponds to Initialization script
- “HbbTV_DVB_Handle.php” corresponds to Handle script
- “HbbTV_DVB_Intermediate.php” does not correspond to any script mentioned above. It simply provides some helper functionalities needed before each validation scope
- “HbbTV_DVB_MPDValidation.php” corresponds to MPD Validation script
- “HbbTV_DVB_RepresentationValidation.php” corresponds to Segment Validation script
- “HbbTV_DVB_CrossValidation.php” corresponds to Cross Validation script

3 HLS Conformance Framework

The HLS conformance software comprises of different modules for each extension that is together hosted in the main part https://github.com/Dash-Industry-Forum/DASH-IF-Conformance. The different modules are:

a. Conformance-Frontend-HLS:
   https://github.com/Dash-Industry-Forum/Conformance-Frontend-HLS/
b. HLS:
   https://github.com/Dash-Industry-Forum/HLS/
c. ISOSegmentValidator:
   https://github.com/Dash-Industry-Forum/ISOSegmentValidator/
d. CMAF:
   https://github.com/Dash-Industry-Forum/CMAF/
e. CTAWAVE:
   https://github.com/Dash-Industry-Forum/CTAWAVE/

(a), (b) and (c) are the core modules for the validation software. (a) contains the source code of the web interface. (b) is the core HLS related validation module and is common to all the other extensions. (c) is the core ISOBMFF implementation for segment validation. (d) and (e) are the extensions to CMAF and CTA WAVE specifications.

The functional diagram of the conformance software and modules structure is provided in Figure 2. This diagram contains individual modules, the scripts each module contains, and their connection to each other. The software includes the implementation of “Client” and “Conformance Server” shown in the diagram.

The orange rectangles define the different modules with specific functionalities. Detailed explanation on each module is provided in the remainder of this section.

3.1 Core Modules

3.1.1 Conformance-Frontend-HLS

This module is the client UI for HLS manifests, containing all the UI-related functionalities to be provided to the user.

During conformance testing, the users can interact with the UI to obtain information on the progress, and results. For each conformance testing, a different session is created, i.e. separated by session folders. Each session folder contains the content and the reports generated for the corresponding test. The storage of session folders resides here as well.

This module uses HTML, JavaScript and PHP. “Conformancetest.html” and “Conformancetest.php” are the main scripts in this module that provide the client UI and responsible for initiating the conformance server processes and updating the client about the overall progress as well as the conformance results.

As in Conformance-Frontend module in DASH conformance, it initially interacts with “Process.php” in “Utils” block as soon as “GO” button is clicked on or local m3u8 file is uploaded. After the conformance check process starts, the final results are sent to the frontend module for display. These results contain:

- Representation checks
- Cross-representation checks

3.1.2 Utils

This block is the closest point to the frontend module. It contains useful functionalities that will be needed for HLS and the extensions (CMAF and CTA WAVE).

“Process.php” script is the initial script to be called from the frontend module when the conformance testing is requested to be performed. It initiates the session creation (temporary storage folder) in the conformance frontend module for the individual request by calling “Session.php” in this block. It also initializes the related global parameters according to the user input in “GlobalVariables.php”. After this, it starts the conformance testing by calling “HLSProcessing.php” in HLS module.

“FileOperations.php” provides some useful helper functions regarding file operations, such as creating, deleting, opening, closing and renaming a file, etc. “Load.php” script is used to load any well-formed file into a DOM XML structure. “VisitorCounter.php” stores the session information for debugging and statistics collecting purposes.

“PrettyPrint.php” is used when the conformance software is used via command line rather than from the user interface (“Conformance-Frontend-HLS”) to print the result logs in a nicely-structured way onto console.

“SegmentDownload.php” is used when Segment Validation is performed to download the signalled initialization and media segments in each Media Playlist in the provided Master Playlist from the user input. “SegmentValidation.php” is executed to perform the core ISOBMFF parsing and validation of the downloaded segments in each Representation by calling “ValidateMP4.exe” in ISOSegmentValidator module.

3.1.3 HLS

In conformance software, the validation checks are dependent on classification of the tracks, such as track/representation media type, switching sets/adaptation sets. In DASH, this is quite intuitive as the classification itself is already provided via MPD. However, from HLS manifest specification, it is observed that almost all tags in a manifest are optional to be included, such as “CODEC”, “AUDIO”, “VIDEO”, etc., which makes this type of classification difficult to be performed.

Therefore, for HLS content validation support, such classification is performed without relying on the HLS manifest tags but rather by analysing the track ISO BMFF headers (“hdlr” and sample description boxes). The complete algorithm for HLS processing is as follows:

1. Determine if the manifest is master playlist or media playlist. If it is a media playlist, skip to step 3.
2. Extract all the media playlist URL(s) provided in the master playlist and group them according to their pointers (“EXT-X-STREAM-INF”, “EXT-X-I-FRAME-STREAM-INF”, and “EXT-X-MEDIA”).
3. For each media playlist pointer
    a. Create media playlist pointer directory.
    b. For each extracted media playlist URL in pointer directory
        i. Create media playlist directory inside media playlist pointer directory.
        ii. Extract segment URL(s) provided in the media playlist.
        iii. Download the segment(s).
        iv. Store them in media playlist directory.
    c. For each extracted media playlist URL in pointer directory
        i. Assemble the segment(s) together into one media file.
        ii. Perform segment validation via “ISOSegmentValidator” submodule.
        iii. Determine the media type based on the output of segment validation (by looking at ISO BMFF “hdlr” and sample description boxes)
4. Group the tracks into Switching Sets, using the media type information.
    a. Create a new Switching Set directory.
    b. Move all tracks associated with this Switching Set in this directory.
    c. Example
5. Perform cross validation checks between all tracks in a Switching Set.

“HLSProcessing.php” is the script that performs the described procedures. The main function in this script is process_HLS() which is called from “Process.php” in Utils block when HLS manifest input is detected.

With this approach, the classification can be done after all tracks are downloaded; therefore, the validation reports are only displayed on the UI after all the validation checks are complete as opposed to providing the report after each media playlist right after their completion (Representation as in the case of DASH support). It should be noted that only segment and cross validation conformance is supported in HLS conformance and m3u8 manifest validation is out of scope.

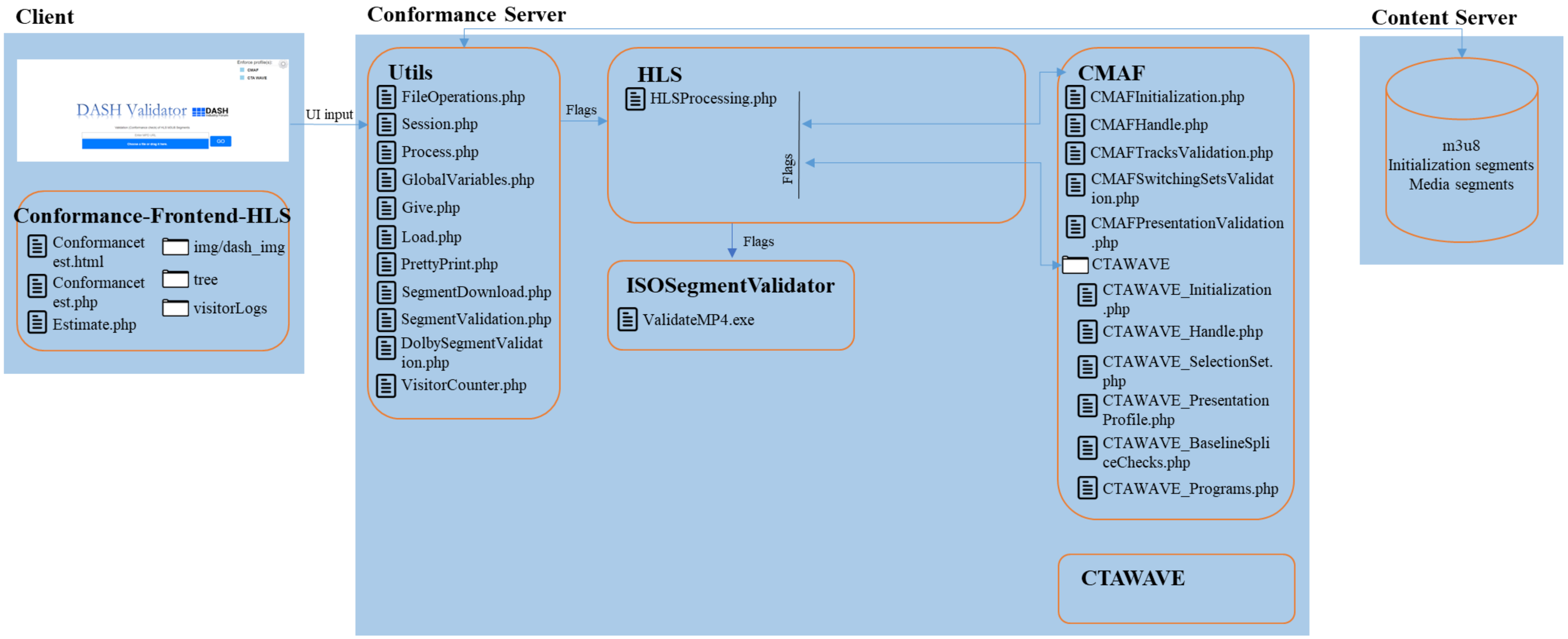

Figure 2: Functional diagram of HLS Conformance Software.

Nomor Research GmbH
Brecherspitzstraße 8
81541 München

3.2 *Extension Modules*

In HLS conformance, supported extension modules are CMAF and CTA WAVE. The detailed information on these extension modules are provided in Section 2.2.3 and Section 2.2.4. As in DASH conformance, these modules are added to the overall validation process whenever either or both “CMAF” and “CTAWAVE” options are clicked from the HLS frontend module.

4 Dynamic Service Validator Framework

Dynamic service validator code is hosted on GitHub at https://github.com/Dash-Industry-Forum/DynamicServiceValidator/. Similar to the other tools, it provides a web interface for user input and interaction written in HTML5 and the core functionalities is written in JavaScript.

When provided with input MPD with MPD@type of dynamic:

- It parses the MPD and derives the initialization and media segments location as well as computes their availability start and end times.
- It sends HTTP HEAD requests for each available segment at its availability start time to confirm that it is accessible and therefore its timing is correctly signalled in the MPD.
- It sends HTTP HEAD requests for each computed segment at its availability end time to confirm that it is no longer accessible and therefore its timing is correctly signalled in the MPD.
- It dynamically reports these findings onto the web interface for display.

It also supports MPD.Location, MPD updates, MPD.UTCTiming and low latency features. In case, RTT correction or clock skew input is provided these are taken into account in the segment availability times computations.