INTEGRATED CONFORMANCE SOFTWARE

The existing conformance software (https://github.com/Dash-Industry-Forum/Conformance-Software) has been extended according to new standards, such as CMAF, DVB-DASH, and HbbTV over time. However, these extensions have been spread over different parts of the entire software; and this makes testing, managing and further extending processes quite difficult. Therefore, a code refactoring which would clearly separate the different modules, functionalities and extensions from each other has been initiated.

This repository contains common modules (Utils, webfe) and submodules (DASH, CMAF, HbbTV, ISOSegmentValidator). Each submodule is a repository on its own and all the submodules need the common modules.

The rest of the document describes the new conformance software tool structure, specifically, the overall functional overview, new module structure, and the detailed description of what each module is responsible for.

1. Installation

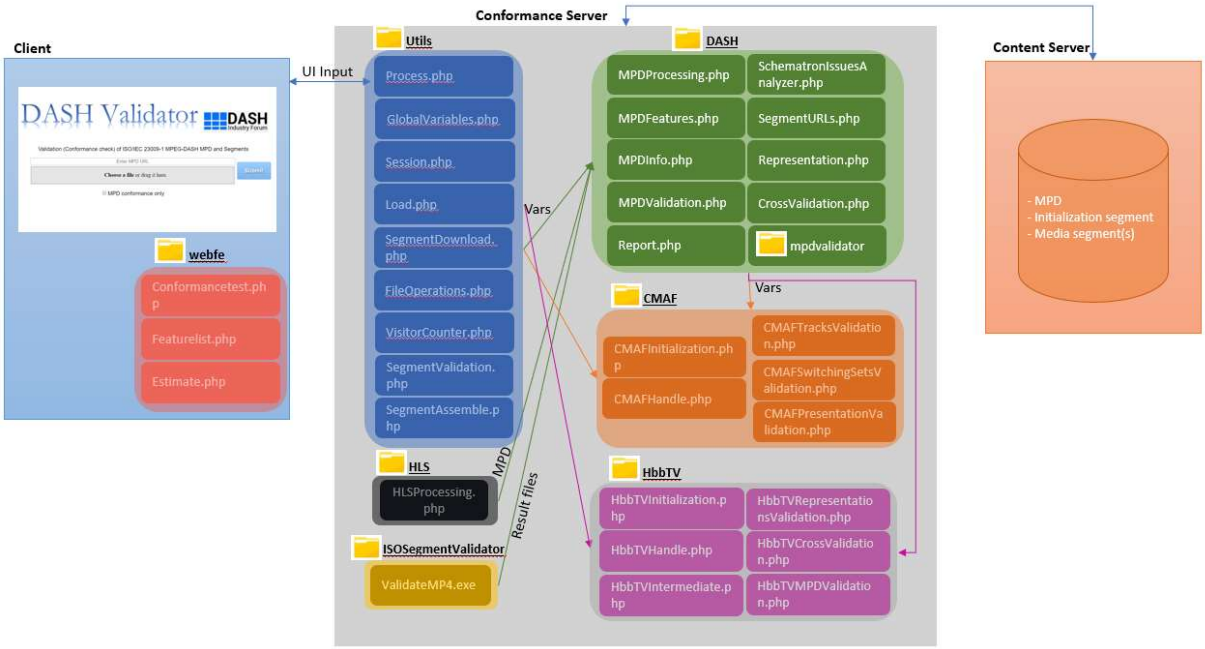
To clone the IntegratedConformance with all the submodules,

git clone --recurse-submodules https://github.com/Dash-Industry-Forum/IntegratedConformance

For the complete installation including dependencies etc, please refer [Installation guide]

2. Architectural Diagram

The functional diagram is drawn for the refactoring design and it is as the following. This diagram contains individual blocks, their scripts, and their connection to each other.



3. Modules

The envisioned module architecture is as shown in Figure below with red-dashed rectangles. As can be seen, there are six separate modules:

1) Integrated Conformance, the main module, containing the all source code of every block

2) DASH, containing DASH related functions and DASH MPD conformance checks

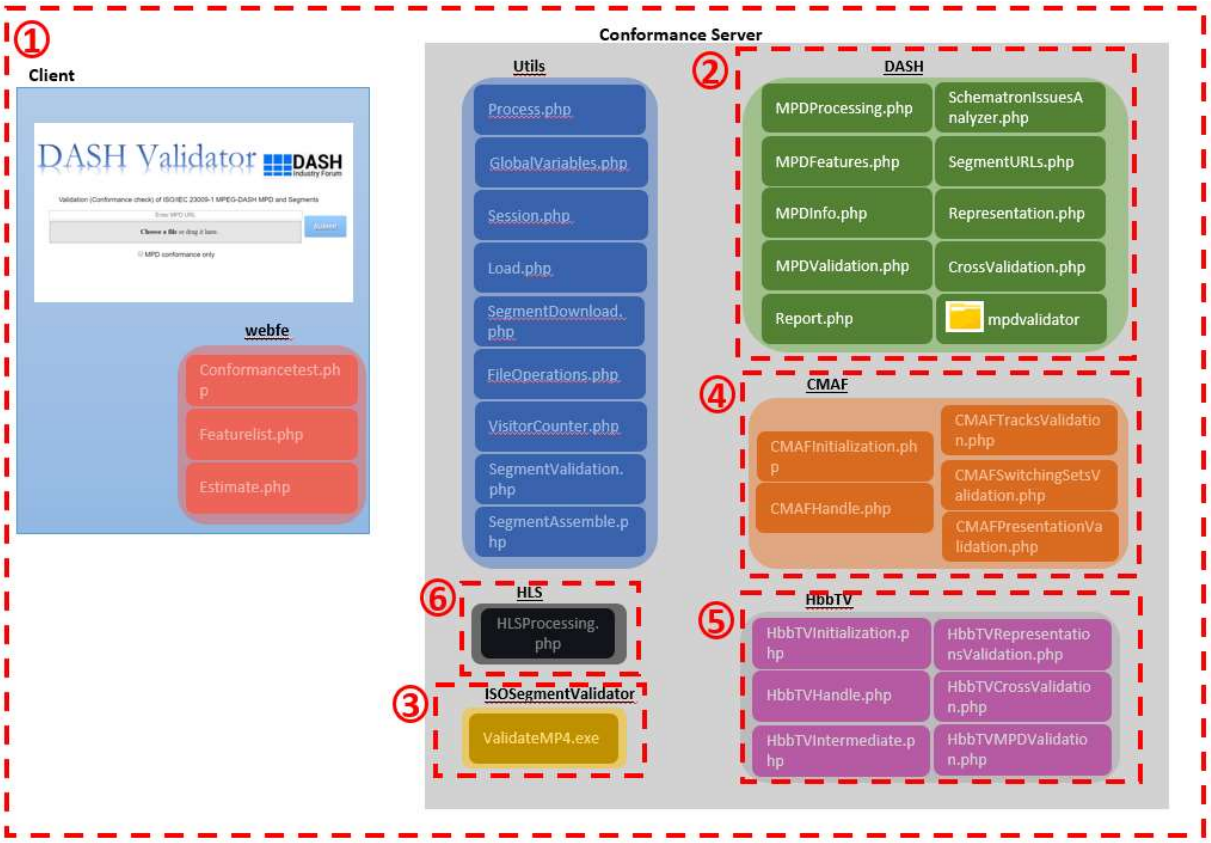
3) Backend, that contains the segment validation software

4) CMAF, containing CMAF related functions and CMAF conformance check

5) HbbTV, containing HbbTV-DVB related functions and the conformance check

6) HLS, containing HLS manifest conformance (still not yet added to the repository!)

These modules will have separate Github repositories; however, the first module will also contain the other modules as submodules.



4. Functional Blocks

4.1. Client - webfe

4.1.1. Conformancetest.php & Conformancetest.html

This module is the original client UI, responsible for initiating the conformance server processes and updating the client about the overall progress as well as the conformance results.

It initially interacts with “Process.php” in “Utils” block as soon as Submit button is clicked on or local MPD file is provided. This interaction is shown via X1 interface in the Figure. After the conformance check process starts, the intermediate and final results are sent to the Client block for display. These results contain:

- MPD validation results
- MPD feature list (via Featurelist.php)
- Representation checks
- Cross-representation checks

4.1.2. Featurelist.php

This script is responsible for generating the MPD features to be sent to the Client block for display, just as in the original conformance software.

4.1.3. Estimate.php

This script provides information on minimum buffer time and bandwidth based on the information provided in each representation.

4.2. Utils

This block is the closest point to the Client block. It contains useful functionalities that will be needed for DASH and other extensions (CMAF, HbbTV, CTA WAVE afterwards). The scripts in this are as seen in the Figure nearby. The detailed explanation of each script is as the following:

Process.php

This script is the initial script to be called from the Client block when the conformance testing is requested to be performed.

It adds the other scripts in this block and DASH block, as these two blocks constitutes the default-mode Conformance Server.

Additionally, it initiates the session creation (temporary storage folder) for the individual request by calling Session.php in this block. After the session creation, it initiates the conformance testing by calling MPDProcessing.php in DASH block.

GlobalVariables -> to store generic global variables and filename templates

Segmentdownload -> fetching segments and storing them

Session -> create and delete temp folders

FileOperations.php

This script provides some useful helper functions regarding file operations, such as opening, closing and renaming a file, directory creation and deletion, computing the relative path of a file, and a modified version of command line execution process which enables both STDERR and STDOUT output to be visible to PHP. As a side note, the relative path computing is done for reporting the Client about the locations of generated HTML files.

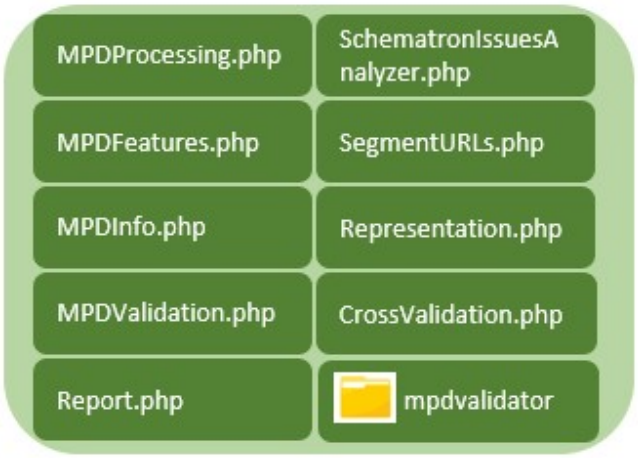
Load.php

This script is used to load provided MPD file into a DOM XML structure or any well-formed file into a DOM XML structure.

SegmentValidation.php

This script is responsible for the segment validation. It is called from MPDProcessing.php in this block for every representation. As input it requires segment(s) URL(s) corresponding to the specific representation. After this it downloads the segment(s) by calling download_data() in SegmentDownload.php in this block. If successfully downloaded, it adds the flags that will be passed to the Backend block. It calls the Backend block executable binary for segment validation, the resulting reports (whose names are saved in GlobalVariables.php in Utils block already) from the Backend block are renamed and reported to the Client block for display.

4.3. DASH



This block contains all the related DASH conformance functionalities, such as parsing MPD, MPD validation, extracting segment(s) URL(s), segment validation, etc. The scripts in this block can be seen in the nearby figure. Detailed explanation about each script is given below.

4.3.1. MPDProcessing.php

This script contains the main functionalities needed for 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 right after a session is created. This function in itself calls the functionalities that load the provided MPD (Load.php in Utils block), parse it (MPDFeatures.php in this block), perform the MPD validation (MPDValidation.php in this block), compute the segmentURLs (SegmentURLs.php in this block), perform segment validation (SegmentValidation.php in this block) and cross checks (CrossValidation.php in this block).

4.3.2. MPDFeatures.php

This script is responsible for loading the MPD XML DOM structure to an array. Currently, this array is the exact representation of the MPD XML DOM. But it is designed for making the actual presentation compact (meaning that the common elements and attributes on different hierarchical levels can be merged into one for each representation). This script is called from MPDProcessing.php as mentioned above.

4.3.3. MPDInfo.php

This script 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 "xs:duration" attributes, determining the availabile segments for dynamic MPDs, and determining the profiles for each representation in each adaptation set in the current period. The last one for example can be moved MPDFeatures.php for obtaining a compact presentation for the MPD.

4.3.4. MPDValidation.php

This script is called for MPD validation including DASH MPD conformance and schema validation (the called java executable is located in mpdvalidator folder in this block). It generates MPD validation report that is reported to Client block for display.

4.3.6. SchematronIssuesAnalyzer.php

This script is used to determine the schematron issues and report them to the Client block for display, just as in the original conformance software.

4.3.7. SegmentURLs.php

This script computes the segment(s) URL(s) for each representation for each adaptation set in the current period from MPD URL, BaseURL and segment access information. Segment access information corresponds to parsing the information provided in the MPD at any level through either SegmentBase or SegmentTemplate with or without SegmentTimeline element. SegmentList is not considered as it was the

case in the original conformance software tool. The information extracted from here can actually be moved to MPDFeatures.php to obtain a compact presentation for the MPD.

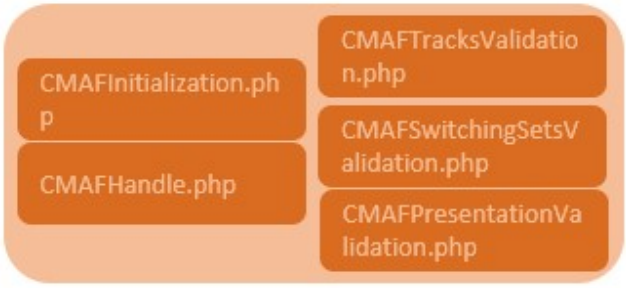
4.3.8. Representation.php

It generates the DASH flags to be passed to the segment validation executable for each representation.

4.3.9. CrossValidation.php

This script contains the DASH cross-representation checks. It is called from MPDProcessing.php after an adaptation set processing finishes and the result reports are then reported to Client block for display.

4.4. CMAF



This block is one of the extension conformances and contains the related functionalities for CMAF conformance. It makes use of Utils and DASH block for MPD processing as it is the same procedure. Different from these blocks, CMAF block introduces additional global variables and conformance checks specific to CMAF.

4.4.1. CMAFInitialization.php

The start of the CMAF process first starts when the query string for CMAF is provided. When this is provided on the Client block and sent to Process.php in Utils block with other input parameters and when Process.php determines this, Process.php includes CMAFInitialization.php script in this block. This script contains the include statements of the remaining scripts in this block and the global variables $function_name and $when_to_call and initialization of the other CMAF-related global variables that are defined as null in Utils block. $function_name is a string containing 'CMAFHandle' and $when_to_call is an array containing strings 'Tracks', 'SwitchingSets' and 'Presentation'. The idea behind these global variables is to provide the CMAFHandle() function in CMAFHandle.php to be called at specific times determined by the $when_to_call array. This way, CMAF is made a black box with an interface.

4.4.3. CMAFHandle.php

When CMAFHandle() is called with the a string from the $when_to_call array, it calls the corresponding script in this block and returns the result that is reported from the corresponding script.

4.4.4. CMAFTracksValidation.php

This script contains the representation checks and called from MPDProcessing.php in DASH block via CMAFHandle with the string 'Tracks'. This procedure is called for each representation.

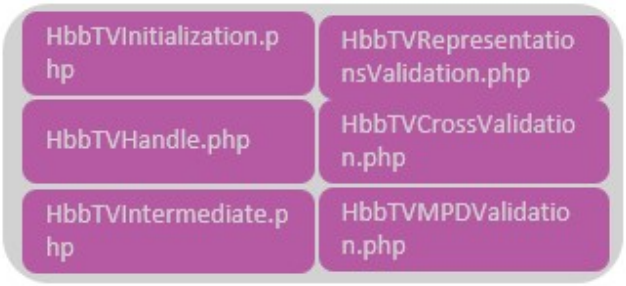
4.4.5. CMAFSwitchingSetsValidation.php

This script contains the adaptation set checks and called from MPDProcessing.php in DASH block via CMAFHandle with the string 'SwitchingSet'. This procedure is called for each adaptation.

4.4.6. CMAFPresentationValidation.php

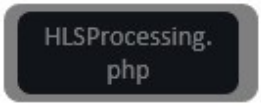
This script contains the cross adaptation set checks and the overall presentation checks. It is called from MPDProcessing.php in DASH block via CMAFHandle with the string 'Presentation'. This procedure is called after the processing of the current period finishes.

4.5. HbbTV



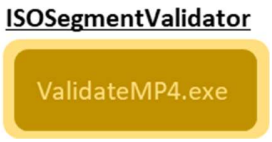
This block is the other extension conformance and contains the related functionalities for HbbTV and DVB-DASH conformance. It makes use of Utils and DASH block for MPD processing as it is the same procedure. Different from these blocks, HbbTV block introduces additional global variables and conformance checks specific to HbbTV just as in CMAF block. It should be noted that this block has not been integrated yet; therefore, this block representation in the figure is just a design. However, it is most probable that the structure will be like this.

4.6. HLS



This block is responsible for processing of HLS manifest. It requires DASH block-like features for this task. This feature is not implemented yet.

4.7. ISOSegmentValidator



This block contains the segment validation software known as the Backend. This module is responsible for the atom/box validation of the ISOBM File Format (MP4) files. This module is called by the main module/Utils to validate the contents after downloading from the URLs. After the execution, it provides information and error log files to be interpreted by the DASH block and reported to Client block from there.

Note: Only the Linux part of ISOSegmentValidator, as we do not support Windows anymore.