INVITATION TO BID FOR
Joint Streaming Conformance Software Project

1 Project Overview

DASH-IF (http://dashif.org) has developed conformance software for the validation of MPEG-DASH and DASH-IF IOP content as well as conformance to DASH-IF IOP guidelines throughout the years. Other organizations including DVB, CTA WAVE, and HbbTV have extended the conformance software for testing the content according to their relevant specifications.

In continuation of this conformance software development, DASH-IF, DVB, CTA WAVE, HbbTV, and ATSC (the "Conformance Partners") are interested to award a contract to improve the current conformance software to achieve the followings goals:

1. Establish a project management system and formal process for the documentation, contributions and code development, testing, metrics, reporting, versioning, and releases;
2. Develop uniform APIs to improve the development, testing, and server updates of the software;
3. Refactor of the current software to modularize and improve its performance,
4. Repair and maintenance project (Pre-existing bugs are excluded regardless of whether they have already been reported or are reported during the course of the project),

   and
5. Develop documentation of the software

and therefore, invite the interested parties to bid on this project.

2 Access to the conformance software

The software has a command-line interface and a web interface and is used to verify the conformance of the received DASH manifest as well as the corresponding media segments.

The open-source code of the conformance software can be found at https://github.com/Dash-Industry-Forum/DASH-IF-Conformance.

The web interface can be accessed at https://conformance.dashif.org/.

The conformance codebase is made of several different subprojects. These subprojects have been mainly written by one contributor and extended by different contributors, due to practical reasons, using different languages over the years. This has resulted in a project which is difficult to maintain. The same software is also used for the conformance test to other specifications, including HbbTV/DVB-DASH, CTA WAVE CMAF, and HLS, and may be extended in the future to additional specifications such as ATSC 3.0.

3 Objectives of this project

3.1 Refactoring

This section describes the refactoring goals at a high level.

3.1.1 Front-end and business logic

The front-end code is implemented in HTML/CSS/JavaScript and PHP. This is not an area of code to focus on replacing as it is currently adequate for the purpose. However, the overall stability of the product is a high priority and this will require changes to business logic/coordination code to manage such high-level issues as:

- Incomplete results being delivered due to premature failure of analysis in a subproject,
- Instability in subprojects.

Such changes that are required should be made with minimal impact on the current implementation.

3.1.2 ISO segment validator

ISO Segment Validator is currently too tightly coupled to the rest of the project. One goal of this project is to move it to a separate project. This would enable it to be developed and maintained following its own management rules.

This component runs independently from the rest. It is an external process called from the PHP logic. As C++ code needs to be built for the target platform since it does not build on many modern targets.

There are multiple possible approaches to replacement. One approach considered during the requirements capture phase of this project was to split the validator into two layers:

- A stream analyzer that parses ISO segments (and possibly component streams), producing a “trace” output of the content of the stream, and
- A trace analyzer that reviews the trace output based on profile requirements.

Implementing as multiple layers makes it easier to target each layer with different engineering needs (for instance, performance against extensibility). It also enables market-proven engineering solutions to be wrapped in a layer that fits more cleanly within the overall project logic.

3.1.3 DASH validator

The use of Java for the DASH validator requires a JVM during the runtime. Java is used in only the DASH validator, and only as a wrapper to the schematron analyzer. It is desirable to replace this module with another schematron wrapper available in a different language so that the use of JVM is not needed anymore.

3.2 Management & QA

3.2.1 Develop the test environment

As part of this project, a test environment should be developed to provide the following facilities:

1. System-wide integrated testing, which demonstrates the overall capability and performance of the system as a whole.
2. Module-level testing to prove sub-projects or modules in isolation. This makes it easier to make localized changes to a module and confirm them before changes are integrated.
3. Unit-level testing to cover testing function/method level changes.
4. Specialist testing such as UX and security testing.

While testing provides a demonstration of system quality and helps to keep the system correct and performant, it is notoriously difficult to demand specific levels of testing in public projects. Therefore, effort should be spent building a framework where developers can easily implement test coverage and monitor the level of testing implemented as a quality metric.

The framework should:

1. be replicable and usable by developers locally,
2. be integrated with a scheduled test runner for independent testing and reporting,
3. enable testing of all sub-projects, regardless of the language used for development,
4. allow for both positive and negative testing,
5. be able to provide metrics on the level of test coverage.

When test assets exist and are publicly available, these should be used as the starting point. For example, the DASH-IF test assets should all pass correctly through the DASH-IF's validator.

There should be the capability to add both known positive tests and known negative tests. This ensures the validator is stable to false positives, i.e., passing material that should not.

As a guideline, preparing the project for containerization will aid the test process and make the product easier for users to recreate.

3.2.2 Code quality metrics

Code quality metrics are useful to identify subprojects that may not have been reviewed and maintained. There are many tools and approaches to code analysis, based on the implementation language, and the test framework should enable a wide selection of these to be used. As a minimum, the following should be considered:

- Lint-like static code analysis
- xUnit-like unit and module test frameworks, ideally with code coverage tools

Documented coding conventions should be developed to help maintain consistency in the code development.

3.2.3 Management structure

Since the conformance software project is extended by different SDOs to support conformance tests for various standards, a uniform management structure needs to be established for various sub-projects. Therefore, a single set of support documentation should be developed to be used across all sub-projects.

3.2.4 Release management

We recognize that individual SDOs (e.g., DASH-IF, DVB, etc.) may have specific use cases for the validator, with their timescale requirements. A fixed and published release schedule will allow contributors to align their submissions. The schedule should be notified to subproject owners and a set of expected changes agreed.

3.2.5 Support documentation

Following the suggestions provided by GitHub (https://opensource.guide), the following minimum set of documentation is required:

1. Code of conduct
2. Contributing guidelines, including the need for test and code quality coverage
3. Issue templates, highlighting the need for supporting test material and adequate description of issues or proposals
4. Pull request template

Primary documentation should be created within the main project wiki. Subprojects may choose to include their documentation if required.

The development and user guide documentation, currently written in Microsoft Word and held in the main project repository, should also be moved to the project wiki to facilitate change as the project evolves (separate documents are very easy to get left behind as changes are made).

3.2.6 Management reporting

To demonstrate value, and to aid the decision process on further investment in the tool, reporting process should be established for:

1. Performance and usage of the online public validator,
2. Quality metrics for the codebase.

The report framework should cover:

1. Usage by module
2. Overall usage (CPU cycles cost, are we getting value for money?)
3. Uptime
4. Usage by region (are we hosting in the correct region?)

All reporting must be anonymized and non-traceable. It must be implemented only in the DASH-IF hosted instance, optional on self-hosted instances (with no central reporting of usage). If the management analysis and reporting require holding of user data, even temporarily ahead of anonymized reporting, then thought must be given to protecting potentially personal or sensitive data.

Note that, if it is found that test data must be kept then thought should be applied to the retention period to manage storage resources.

4 Project Deliverables

A set of milestones will be established based on the project scope and project plan. The contractor(s) are expected to deliver the following items of each milestone according to the milestone table:

1. Report on performed tasks
2. New and updated documentation of the software
3. Test results of running conformance software on the test vectors

5 Project Scope

The project's tasks are defined in the following table.

Catagory	Task class	Tasks	Cost (US $)	Duration (weeks)
C1: management & QA	T1: Management & QA	T1.1: Organize management structure T1.2: Organize release management T1.3: Documentation T1.4: Management reporting (metrics) T1.5: Running project management including triage current issues and allocate to owners for a pre-defined period (assume 12 months)		
C2: Re-factoring	T2: Test framework implementation	T2.1: Create a test and reporting runner framework T2.2: Create a system-wide test framework T2.3: Create module level test framework T2.4: Create unit level test framework T2.5: Create specialist testing framework (UX, security)		
	T3: Test material implementation	T3.1: Evaluate existing test material and identify where the material is incorrect. • requires T2.2 T3.2: System-wide integrated tests T3.3: Module-level tests: • Identify and allocate a fixed amount to bootstrap this activity (not aiming for total coverage) • Requires T4.1+ T3.4: Unit-level tests		

		• Identify and allocate a fixed amount to bootstrap this activity (not aiming for total coverage)		
	T4: APIs update	T4.1: Deep analysis of the existing core->module interfaces and proposing a unified API T4.2: Write ISOSegmentValidator rule dispatcher (but not feed it) T4.3: Porting the CMAF/CTA-WAVE module to the new API (we should start with a small one) T4.4: Porting the DASH core + IOP+ LL module to the new API T4.5: Porting the dynamic Service Validation module to the new API T4.6: Porting the HbbTV/DVB module to the new API T4.7: Porting the HLS module to the new API T4.8: Command-line interface (maybe part of the test framework implementation activity) T4.9: Improve reliability and reporting when runtime errors occur		
C3: repair & maintenance	T5: ISOSegmentValidator	Replace current ISOSegmentValidator with an implementation that keeps performance level but is more maintainable		
	T6: Updating components	T6.1: Bring OS and language versions (such as PHP) up to date T6.2: Investigate and repair stability issues at the server level. Possible solutions for this might include running as a service or container, ensuring CPU/mem/disk is not full T6.3: Make sure user reporting is complete / detect runtime errors and report them		
	T7: Rationalize the number of implementation languages (for instance, remove Java)			

Note: Tasks T2-T7 shall follow the guidelines and practices outlined by 3.2.5 and established during T1.

6 Schedule Summary

1. Project Start: immediately after signing the contract
2. Overall duration: 1 year

7 Warranty

This project requires a 12 month warranty period starting from the submission date of the invoice relating to the final delivery. The contractor(s) shall provide without delay any updates to the software and/or documentation if errors are found.

8 Project Management

The project is managed by the "Conformance Partners" (CP) representatives, mainly from the DASH-IF Interoperability (IOP) Working Group.

9 Bidding Requirements

9.1 Pricing

The received bid shall include the price for each task in section 5.

In addition to the itemized pricing, the bidder(s) may optionally also offer discounted pricing for one or more packages of tasks.

9.2 Additional Information

The bidder shall provide adequate information regarding:

1. The general background of the company
2. A summary of relevant software development projects in the past, in particular, related to conformance software development and/or multimedia streaming
3. The name of experts and their backgrounds intended to work on this project

10 Bidding Process

1. The bidder shall provide the detailed pricing and the duration for each task as well as all additional information through email to DASH-IF Administration admin@dashif.org, by 23.59 PDT on October 31, 2021. Any bid received after this deadline will not be considered.
2. The bidder may use the suggested Annex 1 proposal structure.
3. The bid shall include the contact person's email and phone number.
4. The Conformance Partners' reviewing committee (CPRC) will review all the bids and decide on the project winner(s) considering the received price as well as the credentials and background of the bidders and the proposed teams for this project.
    a. The CPRC may choose more than one winner and breakdown the project between them
    b. The CPRC may decide not to award all parts of the project.
    c. The CPRC may decide to implement the project in multiple phases.
    d. The CPRC may decide not to award the project if reasonable bids are not received.
    e. The bids are not shared with anyone outside of the CPRC or any other bidders.
5. The winner(s) is(are) expected to be contacted by December 15, 2021 to sign the contract(s).
6. Any questions regarding the bid should be sent to DASH-IF Administration admin@dashif.org.

Annex 1: Proposal Structure

1. Table of Contents
2. Executive Summary
3. Proposal breakdown
    - 3.1. Itemized pricing and duration of the project's tasks (Section 5)
    - 3.2. Suggested approaches, methodologies, and additional information on completing various tasks
    - 3.3. Packaged pricing (optional)
4. Bidder's general information
    - 4.1. The general background of the bidder
    - 4.2. A summary of relevant software development projects in the past, in particular, related to conformance software development and/or multimedia streaming
    - 4.3. The name of experts and their backgrounds intended to work on this project
5. Contact information

Each section should start on a new page.