**3GPP TSG-SA WG4 Meeting #132S4-251015**

**Japan, Fukuoka, 19 – 23 May 2025**

**Source: Xiaomi**

**Title: [VOPS] Thoughts on sample bitstream platform (conformance)**

**Agenda item: 9.5**

**Document for: DISCUSSION**

# Introduction

In the context of the VOPS Work Item, the 3GPP SA4 Work Group is envisioning to collect sample bitstreams which conforms to the upcoming TS 26.265.

This contribution provides some early thoughts on how to design the platform as well as the workflow to collect such conformant bitstreams.

We present an architecture for the sample bitstream platform composed of three components together with requirements and possible implementation for each of them.

# Sample bitstream platform overview

#  Suggested overall architecture

At high level, the sample bitstream platform is composed of three main components that are:

1. A database which contains the description of the available sample bitstreams
2. One or more servers to host the submitted bitstreams
3. A portal for external users to search through and download the sample bitstreams.

In addition, a bitstream validator validates prior to upload that the submitted bitstream does comply with the constraints defined in TS 26.265. To harmonize the bitstream description in the database, the bitstream validator would also generate a bitstream description file for the bitstream database.



Figure - Overall view of the sample bitstream platorm

# Thoughts on implementations

# The bitstream validator

# General

The bitstream validator shall:

1. validate that the bitstream is a compliant bitstream according to the corresponding video coding specification and profile.
2. validate that the bitstream is compliant with the bitstream constraints defined in TS 26.265.

For the point 1), the reference video decoder developped by JVET typically have conformance checks embedded. As a result, one way to achieve this validation is to execute the reference decoder with no picture output and verify the absence of error.

For the point 2), we need a way to programmatically express the constraints defined in TS 26.265 to validate the input bitstream. One way would be to extend the reference decoder to add the additional constraints. However, this would require SA4 delegates to learn and modify the reference decoder software project and write those rules directly in the decoder. Alternatively, the input bitstream could be first parsed and based on this parsing, the tool could generate a text dump of the bitstream information and structure (e.g. XML, JSON, etc…). Then, the set of VOPS constraints could be expressed against this text representation of the bitstream. Note that there has been in the past several MPEG standard developped for this purpose such the MPEG-21 Binary Syntax Description Language (BSDL) which could be potentially reused. This approach has the advantage of i) being codec agnostic, i.e. the constraint document could be used for any codec, ii) writing constraints does not require to be familiar with language programming, and iii) the bitstream description generated for the validation can be reused for the database, possibly filtered to keep the relevant information.

# Possible approaches

##### Approach #1: Extending existing validation tools from the DASH-IF conformance suite

The DASH-IF Conformance test suite [1] is a collection of tools to validate DASH MPD and segments. For segments, the ISOSegmentValidator parses ISOBMFF segments and validates their structures. Since NAL units can be present in decoder configuration box, this tool has some parsing functionality of NAL units. However, it does not implement video bitstream validation.

Other open-source tools such as GPAC and its MP4Box tool have the capability to parse NAL units and optionally to dump to an XML format the content of a video bitstream.

##### Approach #2: Developing XML-based validation as part of the SA4 specification work

In this approach, a specification which defines bitstream constraints would be developped along with an XML schema specifying those normative rules. This XML Schema would for instance be developped for each Operation Point in VOPS.

As an example, a basic Python has been developped to produce the following XML document in Figure 2. This XML example relies on the MPEG-B BSDL standard which defines built-in types for bitstream description, e.g. fields of 1 to 32 bits, undefined payload in the form of a byte range, variable-length encoded field, etc.

|  |
| --- |
| […] |

Figure 2 - HEVC XML bitstream description

Once generated, a further tool can validate this HEVC XML description document against a given XML schema specifying the conformance rules. Figure 3 provides such an example of such XML schema. If more complex validation rules, e.g. like those for DASH MPD, are needed, this approach would also require the definition of a schematron which provides higher-level validation capabilities than the XML schema.



Figure 3 - HEVC XML schema using BSDL built-in elements and custom elements

# The bitstream database

The bitstream database should contain the description of each available bitstream. In addition, each bitstream would be linked to a TS number and possibly a profile defined within this TS (using URNs defined to identify each profile).

Since using JSON (or other markup text file) for this purpose seems advantageous, the bitstream database could simply be a git repository on any web-base git service platform (e.g. GitHub, GitLab, etc…)

# The bitstream hosting server

For storing the bitstreams, any http server could be used. Using git LFS would add a finer control with versioning functions but also comes with added constraints for space and effort to set-up. This does not seem necessary to use git LFS at this point.

# The public portal

Ideally, the external users would be able to search for TS number, profiles and retrieve the corresponding bitstream. Additionally, it should also be possible to browse the database by TS number, codec, etc…

In a first version, the home page of the git repository (e.g. GitHub repository’s home page) could be sufficient. It times allows a simple static web frontend could also be developped. The web frontend would retrieve the information about the available bitstream from the database and present them to the user. Using static website would also have the advantage to maintain them on the same web-base git service provider, i.e. GitHub. GitLab, etc…

# Conclusion

We invite discussion on the thoughts provided in this document and encourage starting the work in the upcoming inter meeting period after SA4 #132.

1. Dash-Industry-Forum/DASH-IF-Conformance., <https://github.com/Dash-Industry-Forum/DASH-IF-Conformance/>
2. gpac/gpac: GPAC Ultramedia OSS for Video Streaming & Next-Gen Multimedia Transcoding, Packaging & Delivery, <https://github.com/gpac/gpac>