Text

Description automatically generated ISO/IEC JTC 1/SC 29/WG 3 N01620

**ISO/IEC JTC 1/SC 29/WG 3**

**MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Procedures for test scenarios and reference software development for MPEG-I Scene Description

**Status:** Approved

**Date of document:** 2025-10-23

**Source:** ISO/IEC JTC 1/SC 29/WG 3

**Expected action:** ACT

**Action due date:** 2025-10-23

**No. of pages:** 14 (with cover page)

**Email of Convenor:** young.L@samsung.com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 3**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 3 N** **0** **1620**

**Geneva, Switzerland – October 2025**

|  |  |
| --- | --- |
| **Title** | **Procedures for test scenarios and reference software development for MPEG-I Scene Description** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **25594** |

Contents

[1 Scope 4](#_Toc153446168)

[2 Test scenarios 4](#_Toc153446169)

[2.1 Requirements 4](#_Toc153446170)

[2.2 Scenarios 4](#_Toc153446171)

[2.3 Template for test scenario 4](#_Toc153446172)

[2.4 Call for test data 5](#_Toc153446173)

[2.5 Timeline 5](#_Toc153446174)

[2.6 Available test assets 5](#_Toc153446175)

[2.7 Agreed test scenarios 6](#_Toc153446176)

[2.8 Test assets for validation 6](#_Toc153446178)

[2.9 Large 3D scan datasets 6](#_Toc153446179)

[2.9.1 3RScan dataset overview 6](#_Toc153446180)

[2.9.2 3DSSG dataset overview 7](#_Toc153446181)

[3 Contributions for Extensions 8](#_Toc153446182)

[3.1 General 8](#_Toc153446183)

[3.2 Extension Principles 9](#_Toc153446184)

[4 Software 9](#_Toc153446185)

[4.1. Reference software 9](#_Toc153446186)

[4.2. Conformance software 9](#_Toc153446187)

[5 Gitlab Management 9](#_Toc153446188)

[5.1 Git commit convention 9](#_Toc153446189)

[5.2 Branch convention 10](#_Toc153446190)

[5.2.1. Branch creation 10](#_Toc153446191)

[5.2.2. Branch update 10](#_Toc153446192)

[5.2.3. Tree model 10](#_Toc153446193)

[5.4 Scenarios 13](#_Toc153446194)

[5.5 Summary logistics 13](#_Toc153446195)

[5.6 Coordinators 13](#_Toc153446196)

[6 Promotion 14](#_Toc153446197)

[6.1. MPEG glTF-validator 14](#_Toc153446198)

[6.1.1. Requirements 14](#_Toc153446199)

[6.1.2. Procedure 14](#_Toc153446200)

# Scope

This document provides information and agreed processes to support the development of ISO/IEC 23090-14, "Scene Description for MPEG Media" as well as ISO/IEC 23090-24, "Conformance and Reference Software for MPEG-I Scene Description".

# Test scenarios

## Requirements

The work of the MPEG-I scene description is based on the requirements defined in N18965, later revised to N19511. The coverage of the requirements and the progress is documented in WG3\_N0294.

## Scenarios

An extension to MPEG-I Scene Description should be supported with a well-defined and agreed scenario(s). WG3\_N0294 also covers the mapping of requirements to scenarios.

The template for test scenarios can be found in Section 2.3.

The description of the agreed scenarios can be accessed at https://git.mpeg.expert/MPEG/Systems/SceneDescription/test-assets#test-scenarios. The corresponding test assets as well as additional test assets can be accessed at

<https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/assets>.

***Note****: access and contribution to this requires an account. To request an account, please contact the test asset coordinators (see clause 5.4)*

A new scenario must be proposed with an input contribution to an MPEG meeting with the following information as listed in clause 2.3.

## Template for test scenario

The following table should be used to propose test scenarios for scene description:

|  |  |
| --- | --- |
| **Item** | **Description** |
| **Title** | <give it a catchy title, e.g., as those listed in clause 2> |
| **Description** | * What is the basic use case? * How does it relate to MPEG-I Requirements and Use Cases? |
| **Required test assets** | * 3D scene, real-time assets for media (2D/3D) * Anything else * References to test assets |
| **Current Support** | * How can glTF Scene Description be used today * What are gaps/inefficiencies of glTF2.0 to address this scenario? |
| **Criteria** | * What are relevant criteria for the user experience/QoE? * What are relevant criteria for passing the test scenario? |

## Call for test data

Among others, we solicit the following material to be used as content for the creation and validation of MPEG-Scene Descriptions:

* 2D content that can server as overlays, video textures
* 2D and 3D content that is captured from a local camera, e.g., representing a conference room or flat surfaces for overlay
* 3D game content, e.g., provided in Unity, that can be used for the online gaming scenario
* 3D cinematographic content that includes complete scenes
* VR content and 3D mesh and point cloud content that can be used for VR scenes
* etc.

We welcome contributions of content that can be made available to the MPEG community for the development and progress of the MPEG-I Scene Description activity.

## Timeline

The data sets should be submitted as input contributions to an MPEG meeting but early submission into AHG is welcome.

## Available test assets

The following table lists the available assets and provides a brief description:

|  |  |
| --- | --- |
| **Asset** | **Description** |
| conferenceroom.zip | a glTF asset that represents a conference room. |
| livingroom.zip | a glTF asset that represents a living room. |
| island.zip | a glTF asset that represents an island. |
| chair.zip | a glTF asset that represents a chair. |
| bbb.mp4 | Big Buck Bunny video file in mp4 format. |
| longdress\_frame.ply | a binary PLY file from the longdress point cloud sequence. |
| Scenario 11 | Test Assets:  1. Pine Forest  "author": "fangzhangmnm (https://sketchfab.com/fangzhangmnm)",  "license": "CC-BY-4.0 (http://creativecommons.org/licenses/by/4.0/)",  "source": "https://sketchfab.com/3d-models/pine-forest-ece69535f7584e099488f65f2072264e",  2. woodland-5\_trim\_SN3D.wav  Obtained and modified from EigenScape.  EigenScape is a database of acoustic scenes recorded spatially using the mh Acoustics EigenMike. https://doi.org/10.5281/zenodo.1012809  Marc Green <marc.c.green@york.ac.uk> |
| Test content for cluster spatial audio sources | A simple VR scene where a user can navigate among several audio sources, with the corresponding media streams being delivered over DASH |
| Anchor |  |

***Note:*** *that the first four assets are downloaded from sketchfab and are available for download and usage under the Creative Commons license as describe in CC Attribution License:* [*https://creativecommons.org/licenses/by/4.0/*](https://creativecommons.org/licenses/by/4.0/)*.*

## Agreed test scenarios

Agreed test scenarios are provided here: https://git.mpeg.expert/MPEG/Systems/SceneDescription/test-assets#test-scenarios .

## Test assets for validation

A test asset is an asset which is used to check the validity of the extension. The test assets are used to run simple unit tests which could check for the common and edge cases of the extension. The test assets are glTF files which may not necessarily have an associated buffer. The test asset files are stored in the MPEG glTF-validator repository. For each extension, as shown in Figure 1, a test file is added and it contains the dart implementation to test the extension e.g., a dart file `mpeg\_mpeg.dart`. Alongside, an asset.json file and data folder are created. The asset.json file is a JSON files which provides information on the property to be tested and the corresponding test file to be used. The test files are stored in the data folder. This workflow is inspired by the workflow used by Khronos for glTF-validation. For more information, it is encouraged to check the implementation of the test asset.

A screen shot of a computer

Description automatically generated

Figure 1 File structuring

The workflow for creating unit tests for each extension is described below.

1. Create sample glTF file(s) including the extension specified in ISO/IEC 23090-14.
2. Generate a validation report for each sample file.
3. Create an **assets.json** file which provides path information of the sample glTF files.
4. Implement the testing scripts.
5. Run ‘dart run’ to run the tests in the root folder of Khronos glTF-validator.

## 2.9 Large 3D scan datasets

In academia, large point cloud datasets for 3D scene description. Some of the examples of the point cloud datasets are 3DSSCG (<https://3dssg.github.io/>) and 3Rscan (<https://waldjohannau.github.io/RIO/>). These large point cloud datasets contain multiple 3D snapshots of naturally changing indoor environments. Each scene contains multiple objects whose positions change over time, ground truth annotations of object instances, and a mapping of each 6DoF between rescans.

### 2.9.1 3RScan dataset overview

3RScan includes multiple files to represent a scene. 3RScan represents a scene as a frame image file containing the video of a scene and an object text file with all its information. In addition, there are obj, mtl, png, zip, ply, and json files, which contain the reconstructed mesh, its material files, the corresponding mesh textures, the calibrated RGB-D sensor stream with camera pose, the semantic segmentation visualization, the over-segmentation of the annotated mesh, and the instance segmentation of the mesh (including labels). An overview of the file structure of 3RScan is shown in Figure 2.

A screenshot of a computer

Description automatically generated

Figure 2 3Rscan dataset file structure

### 2.9.2 3DSSG dataset overview

3DSSG is a large dataset that we extended by adding a scene graph to 3RScan, annotating the semantic scene graph with relationships, attributes, and class hierarchies to provide a 3D semantic scene graph for 3RScan. 3DSSG provides 1482 scene graphs totaling 488,000 object nodes and 544,000 edges. Nodes are a set of attributes that drive visual and physical appearance and behavior, with a class hierarchy, and edges represent semantic relationships between these nodes. A 3DSSG is defined as the set of tuples between nodes and edges. This data can be used to easily render a 3D scene graph in 2D.

A 3DSSG consists of a JSON file and a txt file. The JSON file contains a list of all object instances (nodes) in the semantic graph, a list of all semantic relations (edges) in the semantic graph, and a graph partitioned into subsets of edges that are used for training. The txt file organizes the contents of each class, including its affordances, a list of all attributes, and their categories. The file organization of 3DSSG is shown in Figure 3.

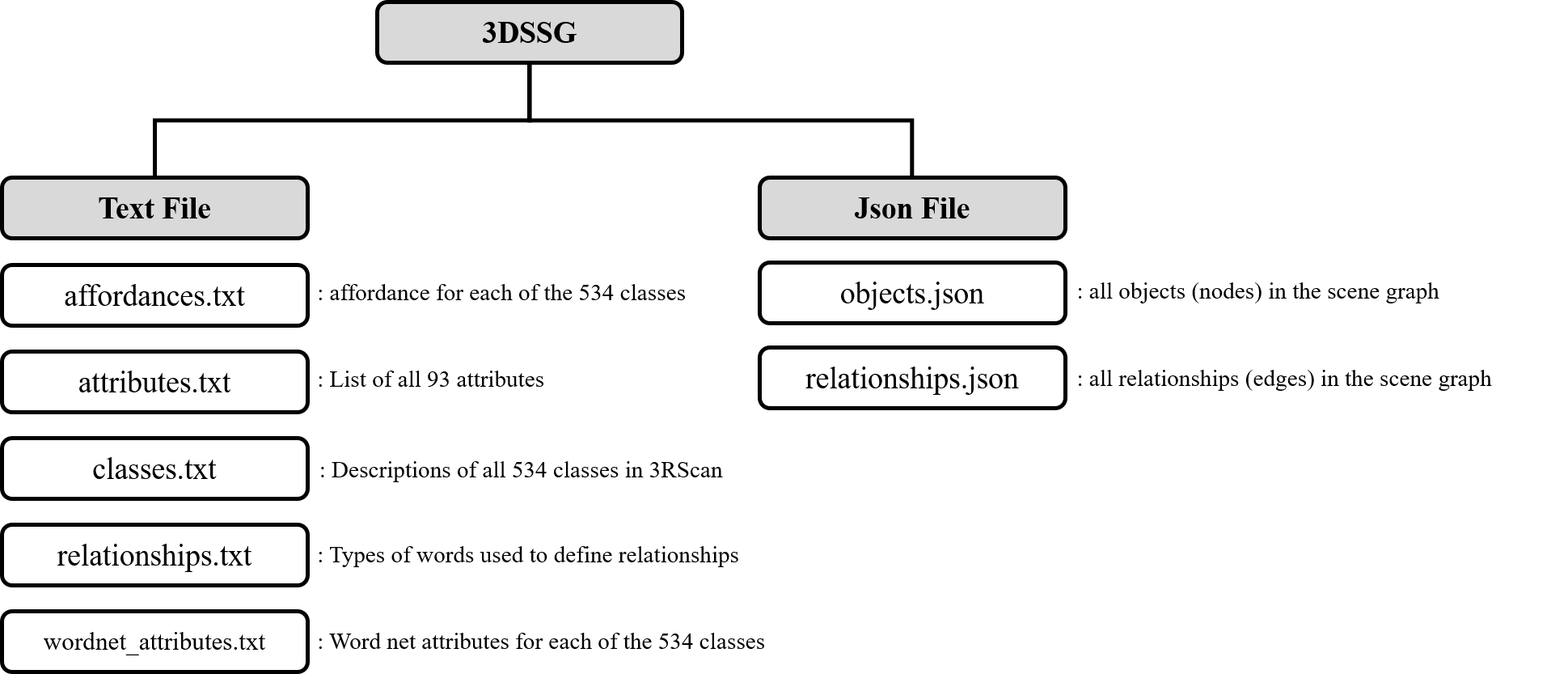


Figure 3 3DSSG dataset file structure

# Contributions for Extensions

## General

For every extension documented in ISO/IEC 23090-14 under the framework in clause 3 the following information is expected to be provided:

* The schema for the extension as part of the standard as well as a JSON document
* The semantics for the extension
* The processing model on the "Presentation Engine"
* The conformance description, i.e. conformance requirements for the Presentation Engine that supports the extension
* *A promise for example content that uses the extension that is finally available within one meeting after the technology was added. If not fulfilled, the feature is expected to be removed and this will be documented as a note in the draft standard.*
* *A promise of a reference implementation to the reference software as well as the conformance software as documented in clause 4, that is finally available within two meetings after the technology was added. If not fulfilled, the feature is expected to be removed and this will be documented as a note in the draft standard.*
* *The reference software implementation should implement the functionality to use the properties defined in the carriage format. Implementations in the reference software should support the mechanism and deliver an expected behaviour.*

**Note**: In an agile manner, this may be achieved using a JSON file which provides all the necessary data which would be included in a sample in ISOBMFF container. The reference software takes the JSON files as input and apply the properties accordingly as per the processing model. Later, the reference software implementation may incorporate and integrate a proper demuxer for the carriage formats.

* *To progress and integrate the support for carriage formats with the reference software, it is expected to bring supporting implementation in the carriage library.*
* *Upon any new merge request, a description of the merge request should be provided. A template description for merge request is opened whenever a new merge request is created to the `develop` branch.*
* *The information contained in the merge request will be provided as an input to the MPEG-I Scene Description AhG at the latest MPEG meeting.*

Hence, contributions addressing extensions to glTF under the framework in clause 3 should include the following:

* The scenarios that the extension is addressing. The scenarios are documented in clause 2.7.

In case, all the above-mentioned information is not available, a documented extension is not moved into the WD/CD but is maintained in the Technology under Consideration (TuC) document. The status of the completed information and the missing one is documented in the TUC.

The following text processes is recommended, but needs final verification:

*To fulfill the requirement on the reference software, it is sufficient to demonstrate that the reference software is able to properly process the test scenario. The test scenario content shall at least have a scene description file in glTF textual format that makes use of the proposed extension. The test scene description glTF document should use one of the available assets. The proposal must indicate any dependencies on other extensions.*

*The following is an example of this procedure:*

* *A test scenario is defined around support for video textures*
* *The proposal is to make use of the MPEG\_video\_texture extension*
* *A sample content is proposed based on the "conferenceroom" glTF file, which is part of the assets. The glTF file is extended to include the MPEG\_video\_texture extension. The bbb.mp4 asset is used to describe the video texture, which is attached to a rectangular mesh in the "conferenceroom" scene.*
* *The reference software is run with the modified scene description document and the expected behavior is demonstrated, showing the video texture.*

## Extension Principles

The following extension principles apply

* If the extension adds a new top-level array (by extending the root glTF object), its elements should inherit all properties of glTFChildOfRootProperty.schema.json.
* Other objects introduced by the extension should inherit all properties of glTFProperty.schema.json.
* By glTF 2.0 conventions, schemas should allow additional properties.
* Names **MUST** begin with an MPEG prefix, followed by an underscore.
* Names MUST use lowercase snake-case following the prefix, e.g., MPEG\_materials\_sand.
* Names **SHOULD** be structured as MPEG\_<scope>\_<feature>, where scope is an existing glTF concept (e.g. mesh, texture, image) and feature describes the functionality being added within that scope. This structure is recommended, but not required.
* Scope **SHOULD** be singular (e.g. mesh, texture), except where this would be inconsistent with an existing Khronos extension (e.g. materials, lights).

## 3.3. Carriage format

A Python-based ISOBMFF file parser is used with the reference software project. The Python-based ISOBMFF file parser will support parsing of typically boxes and sample definitions along with additional boxes and formats as defined in ISO/IEC 23090-14.

The group still maintains and develops the C++-based SDCarriageLibrary to support the function of packaging MP4 files with the carriage formats described in ISO/IEC 23090-14.

A sample file must include at least one format definition in ISO/IEC 23090-14. A sample file must be provided for a new format definition along with source code in the reference software. The information carried in the sample file must be integrated with the reference software. It is also advised to provide source code for the SDCarriage library to allow for a broader interest of file packaging capabilities.

# Software

## 4.1. Reference software

The reference software for MPEG-I scene description is documented in WD of ISO/IEC 23090-24

## 4.2. Conformance software

The conformance software for MPEG-I scene description is documented in the WD of ISO/IEC 23090-24.

# Gitlab Management

## Git commit convention

git commit -m "<optional WIP> <type>(#<issue id> <optional scope>): <description>"

The optional WIP information is to indicate your commit is in the “Work In Progress” state.

Type possible:

* **feat:** *The new feature you’re adding to a particular application*
* **fix:** *A bug fix*
* **hotfix:** *A bug fix to correct a major issue*
* **style:** *Feature and updates related to styling*
* **refacto:** *Refactoring a specific section of the codebase*
* **test:** *Everything related to testing*
* **doc:** *Everything related to documentation*
* **chore:** *Regular code maintenance [something which not fit with other previous types]*

## 5.2 Branch convention

### 5.2.1. Branch creation

git branch <type>/<initials>\_<why>

* **type:** *Everything which can work for a commit message*
* **initials:** *Initials of the owner*
* **why:** *The purpose of this branch written in PascalCase*

### 5.2.2. Branch update

When two developers are working on the same project, they will have their own working branch. If one merge his/her work to the develop branch, the second person should update his/her work to fit with the latest state of the develop branch. There are two possible ways to resolve such a situation:

* Rebasing develop branch to the working branch (recommended solution)
* git checkout <my\_branch\_name>

git rebase develop

* Merging develop branch to the working branch (recommended solution)
* git checkout <my\_branch\_name>

git merge develop

On the other hand, when a working branch is finished and needs to move into develop branch. A pull request needs to be generated directly on GitLab. Once completed in the platform, the working branch will be merged with the develop branch.

### 5.2.3. Tree model

Figure 1 illustrates an example of tree model for development.

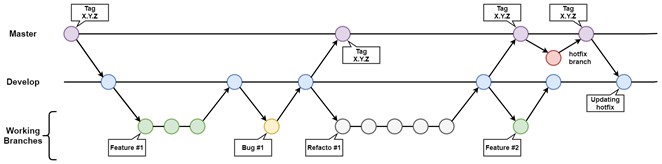


Figure 4. Tree model for branch

* 1. Merge request template

A merge request template is provided as shown in Figure 3. The merge request template is incorporated for the reference software and MPEG glTF-validator software.

****

Figure 3 Merge request template

## Scenarios

To provide use cases that are to be supported by the standard, test scenarios are collected. Scenarios are described on what the basic setup an experience is expected to be and provides along with this test assets and test vectors (may be compressed or uncompressed) that may be used in the scenario. These test scenarios are collected at <https://git.mpeg.expert/MPEG/Systems/SceneDescription/test-assets>.

## Summary logistics

|  |  |  |
| --- | --- | --- |
| **Asset** | **Hosting** | **Location name** |
| Repository | MPEG Gitlab | https://git.mpeg.expert/MPEG/Systems/SceneDescription/ |
| Public repository | Public GitHub | https://github.com/MPEGGroup/Scene-Description |
| Reference software | MPEG Gitlab | https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/reference |
| glTF-validator | MPEG Gitlab | <https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/23090-24-gltf-validator> |
| Carriage library | MPEG Gitlab | <https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/23090-24-carriage> |
| Pipelines | MPEG Gitlab | <https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/pipelines> |
| 3RScan and 3DSSG to glTF conversion tool | MPEG GitLab | https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/gltfconversiontool |
| Test assets | MPEG GitLab with LFS | https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/assets |
| Unit tests for validation software | MPEG GitLab | <https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/23090-24-gltf-validator>  Any test content which is used to test and validate the implementation of the MPEG extension in the glTF-validator must be provided with a corresponding unit test implementation in glTF-validator. The test content must be hosted along with. |

For uploading content to the Test Assets, the preferred option is to create a new branch at <https://git.mpeg.expert/MPEG/Systems/SceneDescription/software/assets> with README.md file describing the test asset and extensions in use. The content will be pushed to the newly created branch with LFS enabled for large binary files. Proponents can also bring a supporting input contribution describing the test asset to the MPEG meeting; however, it is not advised.

## Coordinators

If you have created MPEG Git account but you cannot access the site, then please share the following information.

• Your name

• Your MPEG Git username

Please then send an email containing this information to the GitLab managers as listed in Table 1.

Table 1 Gitlab managers

|  |  |
| --- | --- |
| **Name** | **Email address** |
| Emmanuel Thomas | thomase@xiaomi.com |
| Imed Bouazizi | [bouazizi@qti.qualcomm.com](mailto:bouazizi@qti.qualcomm.com) |

# Promotion

## MPEG glTF-validator

A list of the MPEG-I Scene description extensions is implemented as detailed in the README.md file on the git repository. It is therefore foreseen that the glTF-validator with MPEG extensions will serve as a helpful tool for implementors to validate a glTF asset with MPEG extensions and generate validation reports.

### Requirements

The group is currently investigating and study to promote the tool with Khronos glTF group for adoption. To submit the source code to Khronos glTF-validator repository, a pull request must be created. A Khronos Open-Source Contributor License Agreement (CLA) should be signed upon opening the pull request by the committers. The information about the Open Source CLA can found under the following link: <https://cla-assistant.io/KhronosGroup/glTF-Validator>. The maintainers of the Khronos glTF-validator will review the source code and may provide comments, if any. After the review process, the maintainers will merge the pull request to the official Khronos glTF-validator repository.

### Procedure

One possible procedure is to fork the Khronos glTF-validator repository to MPEGGroup public github organization (<https://github.com/MPEGGroup>) as a separate repository. The source code for MPEG extensions with test assets can be push to the newly forked repository. A pull request from the MPEGGroup/glTF-validatior can be made to Khronos/glTF-validator. The pull request must fulfill the requirements as described in Section 6.1.1.

## 5G-MAG

5G Media Action Group is an international non-profit organization promoting collaboration across the media and telecom industries. 5G-MAG is implementing a software platform for XR media services over 5G. 5G-MAG has adopted several open-source libraries to support 3D and XR services in 3GPP-based on the MPEG-I Scene Description as defined in specification ISO/IEC 23090-14. For more details; please follow the news from 5G-MAG in <https://5g-mag.github.io/Getting-Started/pages/xr-media-integration-in-5g/>. Some of the activities in 5G-MAG are also captured in <https://github.com/MPEGGroup/Scene-Description>.