INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC 1/SC 29/WG 11
MPEG2017/M41732
October 2017, Macau, China

Source: **DMAG-UPC et al.**
Status: **Proposal**
Title: **MPEG-G: Reference Software for Part 1: File Format encapsulator and decapsulator**
Authors: **Daniel Naro , Jaime Delgado, Silvia Llorente (Distributed Multimedia Applications Group – Universitat Politècnica de Catalunya)**

Table of contents

1 Introduction

The proposed reference software provides a tool to generate and parse an MPEG-G file as described in [1]. In the current development status, the project creates an MPEG-G file with hardcoded parameters, and using the following files in the working directory:

Filename	Content
dgMetadataFileName	Metadata element content for the datasets group
dgProtectionFileName	Protection element content for the datasets group
dtParamsFileName	Parameters element content for the dataset
dtMetadataFileName	Metadata element content for the dataset
dtProtectionFileName	Protection element content for the dataset
streamMetadataFileName	Metadata element for the stream
streamProtectionFileName	Protection element for the stream
streamData	Data payload for the stream

The result of this encapsulation process is the generation of a file called exampleMPEGG in the working directory.

In the decapsulation phase, the previously generated file is parsed and the data of each stream written to disk (in this mock example, every stream contains the same information).

This simplistic behaviour is meant to show the use of the underlying library, and a further version will allow specifying more parameters.

The code is provided in the git repository at http://gitlab-scistimm.epfl.ch/MPEG-G/mpegg-reference-sw in the branch named part1.

2 Project structure

The project is structured around the different gen-info boxes described in [1]. For each of these boxes, the data structures provide the functionality to define them, compute their sizes, write them or parse them.

Additionally, other data structures are implemented to support the functionalities; namely data structures to read and write bit-streams (opposed to byte-streams), Signatures and signature-streams, and labels.

Although the new structures have been implemented in C, for the moment the project is still in C++ until the legacy code is translated. This mainly affects the classes implementing the gen-info boxes.

3 Encapsulator

When creating a new file, each gen-info box has to be created in memory, setting its parameters to the desired values. For example, in the case of creating the file header, the procedure would be as follows:

```
    File file;
    char majorBrand[] = "MPEG";
    uint32_t minorVersion = 1;
    file.setFileHeader(new FileHeader(majorBrand,minorVersion));
```

This project does not provide the structures to generate the content of some boxes, such as the metadata boxes or protection at the different levels. It rather expects to be instructed where to find the content to be placed in the corresponding structure. For example in the case of the dataset group's metadata box:

```
char dgMetadataFilename[] = "dgMetadataFileName";
DatasetsGroupMetadata* datasetsGroupMetadata =
    new DatasetsGroupMetadata(datasetsGroupContainer);
datasetsGroupMetadata->defineContent(dgMetadataFilename);
datasetsGroupContainer->setDatasetsGroupMetadata(datasetsGroupMetadata);
```

When the file is being written, the content of the file indicated with defineContent is written to disk as the content of the box.

4 Decapsulator

Whereas during the encapsulation phase, the methods are to be called in order to generate the box tree in memory, in case of the decapsulation, the same tree is obtained while parsing the file.

```
  FILE* inputFile = fopen(filename,"rb");
    File* file = File::parseFile(inputFile, filename);
    fclose(inputFile);
```

In this case the metadata and protection boxes store a seek point within the input file, to avoid having to load this content into memory.

The file can be cloned by calling the same write method, as previously seen, or each stream can be written to disk to be accessed with another software, by accessing the corresponding nodes in the tree:

```
    unsigned long datasetGroupContainersSize = file->getDatasetGroupContainersSize();
    for(int datasetGroupContainers_i=0;
        datasetGroupContainers_i<datasetGroupContainersSize;
        datasetGroupContainers_i++ ){
        DatasetsGroupContainer* datasetsGroupContainer =
file->getDatasetGroupContainer(datasetGroupContainers_i);
        if (datasetsGroupContainer == nullptr) continue;
        unsigned long datasetContainersSize =
                datasetsGroupContainer->getDatasetContainersSize();
        for(int datasetContainers_i=0;
                datasetContainers_i<datasetContainersSize;
                datasetContainers_i++){
            DatasetContainer* datasetContainer =
datasetsGroupContainer->getDatasetContainer(datasetContainers_i);
            if (datasetContainer == nullptr) continue;
            unsigned long streamsSize=datasetContainer->getStreamsSize();

            for(int stream_i=0; stream_i<streamsSize; stream_i++) {
```

```
                StreamContainer*
streamContainer=datasetContainer->getStream(stream_i);
                if (streamContainer != nullptr) continue;

                char outputFileName[150];

sprintf(outputFileName,"/home/gencom/ClionProjects/MPEGG_capsulator/dummyTestFiles/str
eam_%i_%i_%i",
                        datasetGroupContainers_i, datasetContainers_i, stream_i);
                FILE* streamOutputFile = fopen(outputFileName, "wb");
                streamContainer->getData().write(streamOutputFile);
                fclose(streamOutputFile);
            }
        }
    }
```

5 References

[1] ISO/IEC JTC 1/SC 29/WG 11 N17075 - Text of ISO/IEC 23092-1 WD 3, Transport and Storage of Genomic Information, July 2017.