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
MPEG2018/M42106
January 2018, Gwangju, Korea

Source: **DMAG-UPC**
Status: **Proposal**
Title: **Proposal for CD text of MPEG-G Part 3: Genomic Information Metadata and Application Programming Interfaces (APIs)**
Authors: **Jaime Delgado, Silvia Llorente, Daniel Naro (Distributed Multimedia Applications Group – Universitat Politècnica de Catalunya)**

Contents

1 Introduction

The ZIP file corresponding to M42106 includes two documents:

- This document, which describes the modifications introduced in document N17141 [1] in order to prepare a proposal for CD text of MPEG-G Part 3: Genomic Information Metadata and Application Programming Interfaces (APIs).

- The CD text proposal itself.

2 Modifications

The main modifications introduced in N17141 are as follows:

- Application of CD template to the document, adding some new sections like Normative References, Abbreviations, etc.

- Resolution of some of the notes appearing in the document, including the inclusion of XML Schemas as Annexes.

- Harmonization of tables, including table captioning and numbering.

- Restructuring of APIs to MPEGG data section and subsections:
    - New grouping of operations, separating Access Operations into core and extensions.
    - Update of the data structures where the operations have to be applied to make them correspond to other standard parts.
    - Addition of missing operations at some levels.
    - Definition of the Access core operations at the Dataset level.

3 References

[1] N17141 - Text of ISO/IEC 23092-3 WD [1], Genomic Information Metadata and APIs, Macau, China, October 2017.

ISO 23092-3:2018

ISO/IEC JTC1/SC 29/WG11

Secretariat: JISC

Information Technology — ISO/IEC 23092 — Part 3: Genomic Information Metadata and Application Programming Interfaces (APIs)

CD stage

Warning for WDs and CDs

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by ISO/JTC1, Subcommittee SC29, Working Group 11.

This is the first edition of ISO/IEC 23092 Part 3. ISO/IEC 23092, Genomic Information Representation, is composed of the following parts:

Part 1: Transport and Storage of Genomic Information

Part 2: Coding of Genomic Information

Part 3: Genomic Information Metadata and Application Programming Interfaces (APIs)

Part 4: Reference Software

Part 5: Conformance Testing

Introduction

The development of High-Throughput Sequencing (HTS) technologies enables the usage of genomic information as everyday practice in several fields. The growing volume of data generated requires efficient representation of the genomic information to support interoperability among tools and systems. The lack of appropriate standard representations and efficient compression technologies of genomic data is widely recognized as a critical element seriously limiting its application potential in all fields using or willing to use genomic data.

This document was developed in response to worldwide demand for new effective interoperable solutions for genomic information processing applications covering all the chain from sequencing to storage and analysis.

This document includes the specification of syntax and semantics for the metadata and APIs (Application Programming Interfaces) for genomic information representation.

Information Technology — Genomic Information Representation — Part 3: Genomic Information Metadata and Application Programming Interfaces (APIs)

1 Scope

This document includes the specification of information metadata, SAM interoperability, protection metadata and programming interfaces to genomic information:

- Metadata storage and interpretation for the different available layers are treated in Section 5.
- Protection elements (providing confidentiality, integrity and privacy rules at the different layers coded in compliance with Parts 1 and 2 of ISO/IEC 23092) are treated in Section 6.
- Mechanisms for backward compatibility with existing SAM content, and exportation to this format are treated in Section 7.
- Interfaces to access genomic information coded in compliance with Parts 1 and 2 of ISO/IEC 23092 are treated in Section 8.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23092-1 Information technology -- Genomic Information Representation -- Part 1: Transport and Storage of Genomic Information

ISO/IEC 23092-2 Information technology -- Genomic Information Representation -- Part 2: Coding of Genomic Information

ISO/IEC 23092-4 Information technology -- Genomic Information Representation -- Part 4: Reference SW

ISO/IEC 23092-5 Information technology -- Genomic Information Representation -- Part 5: Conformance

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at https://www.iso.org/obp

3.1

access unit
Logical data structure containing a coded representation of information to facilitate the bit stream access and manipulation.

3.2

access unit start position
left-most genomic record position among all genomic records contained in the access unit

3.3

access unit end position
right-most genomic record position among all genomic records contained in the access unit

3.4

access unit range
genomic range comprised between the AU start position and the right-most genomic record position among all genomic records contained in the access unit

3.5

access unit size
number of genomic records contained in an access unit

3.6

access unit covered region
genomic range comprised between the au start position and the au end position

3.7

alignment
A sequence read mapped on a reference sequence

3.8

BAM
Compressed binary version of SAM

3.9

base
In the context of this document it is used as synonymous of nucleotide

3.10

CIGAR string
It is a sequence of base lengths and the associated operations used to indicate things like which bases align (either a match/mismatch) with the reference, are deleted from the reference, and are insertions that are not in the reference

3.11

dataset
Compression unit containing sequence reads and possibly alignment information

3.12

dataset group
Collection of one or more datasets

3.13

FASTA
GIR that includes read headers and sequence reads (nucleotides sequences)

3.14

FASTQ
GIR that includes FASTA plus quality scores

3.15

genomic record
Data structure encoding either a single sequence read or a paired sequence read optionally associated with alignment information, read identifier and quality values

3.16

genomic record length
Distance between the left-most mapped base coded in the record and the right-most mapped base coded in the record

3.17

genomic range
Interval of positions on a reference sequence defined by a start position s and an end position e such that $s \leq e$. the start and the end positions of a genomic range are always included in the range

3.18

genomic record position
Position of the left-most mapped base of the genomic record on the reference genome

3.19

indel
An additional or missing nucleotide in a DNA sequence with respect to a reference DNA sequence

3.20

mapped base
It is either:

- a base of the aligned read matching the corresponding base on the Reference Sequence
  or

- a base of the aligned read that does not match the corresponding base (a.k.a. single nucleotide polymorphism)

3.21

mate pairs
Two reads from the same (long) DNA strand extracted by sequencing machines. The orientation is the opposite of paired ends

3.22

paired ends
A couple of reads produced from the same (short) DNA fragment by sequencing both ends. The orientation is the opposite of mate pairs

3.23

quality score
It is assigned to each nucleotide base call in automated sequencing processes. It expresses the base-call accuracy

3.24

read header
Each sequence read stored in FASTA and FASTQ format starts with a textual field called “read header” containing a sequence identifier and an optional description

3.25

reference genome
It is a digital nucleic acid sequence database, assembled by scientists as a representative example of a species’ genetic material

3.26

reference sequence
It is a sequence of nucleotides associated to a one-dimensional integer coordinate system for which each integer coordinate is associated to a single nucleotide. Coordinate values can only be equal or larger than zero. This coordinate system in the context of this standard is zero-based (i.e. the first nucleotide has coordinate 0 and is said to be at position 0) and linearly increasing from left to right

3.27

SAM
GIR that is human readable and includes FASTQ plus alignment and analysis information

3.28

(genomic) segment
A contiguous sequence of nucleotides

3.29

sequence read
The readout, by a specific technology more or less prone to errors, of a continuous part of a segment of nucleotides extracted from an organic sample

3.30

soft clips
Soft clipped bases are portions of an aligned sequence read which do not match well to the reference genome on either side of the read and are therefore ignored for the alignment, but still kept in the aligned read

3.31

template
A DNA sequence which is sequenced on a sequencing machine or assembled from sequence reads

4 Abbreviations

GIR Genomic Information Representation

5 Conventions

5.1 Character encoding

This specification utilizes UTF-8 character encoding.

6 Information metadata

6.1 Introduction

The metadata structure and the set of elements is specified using XML [5].

This standard defines a minimum core set of metadata elements, which can then be extended by users and applications by including extra information elements. Metadata sets are specified for a Dataset Group as specified in Part 1 of this Standard and for a Dataset also specified in Part 1 of this Standard.

Extensions to (i.e., new elements for) the metadata set specified in this standard are represented with an information type identifier, a value and a pointer to a resource documenting the semantics of the given information type.

Metadata profiles are specific subsets of metadata sets specified using mechanisms provided in the standard. A metadata profile specified in this Standard may correspond to well-known metadata sets specified or used out of this standard, such as those in ENA or EGA [2] and NCBI specifications [3], as examples. This allows easy interoperability with already existing systems. A metadata profile includes a subset of core elements described in this standard, and a set of new elements specified with the extensions mechanism (see Clause 6.5).

The rest of Clauses of this standard specify Dataset group metadata (Clause 6.2), Dataset metadata (Clause 6.3), Extensions (Clause 6.4) and Profiles (Clause 6.5).

6.2 Dataset Group metadata

Dataset group metadata is associated to a genomic study and is stored within the dataset group metadata container. Table 1 presents the core set of metadata elements in a dataset group metadata container. Those elements that are necessary to identify and process the dataset group are classified as mandatory.

Table 1: Dataset group's metadata core set

Element name	Element type	Mandatory
Title	String	Yes
Type	Controlled vocabulary	Yes
Abstract	String	No
Project centre name	ProjectCentre type	No
Description	String	No
Samples	ListOfSamples type	Yes
Extensions	ListOfExtensions type	No

The conversion of Table 1 to an XML schema is done as follows. Each row is translated into one element of the type indicated in the element type column, with a maximum occurrence of one and a minimum occurrence depending on the mandatory nature of the element. In the case where the type is controlled

vocabulary, the XML schema represents the data as a string, but all words not included in the list of controlled vocabulary are considered as ill-formed. The resulting schema is provided in Annex A.1.

As previously introduced, an extensions type is the combination of three fields: the value, the identifier of the extension, and a link to a resource documenting the interpretation of the field. In the XML schema, this is translated as an element with two attributes: the identifier (of type string) and the resource (a URL); the value is represented as the element's text as UTF-8 characters text (in case of binary information, Base64 encoding is used). Additionally, a Boolean attribute of the element indicates if the extension is only relevant to the dataset group, or if the dataset also inherits it. The resource documentation might be human readable, and the extensions parsing is not required.

As Table 1 indicates, certain elements can be described with basic types, but other element types require more complex descriptions, such as the sample type. For those elements, their respective core set of fields is provided. Table 2 lists those for the sample type, and Table 3 for the project centre type.

Table 2: `Sample`'s metadata core set

Field name	Field type	Mandatory
`TaxonId`	Integer	Yes
`Title`	String	No
`Extensions`	ListOfExtensions type	No

Table 3: `Project centre`'s metadata core set

Field name	Field type	Mandatory
`ProjectcentreId`	Integer	Yes
`Title`	String	No
`Extensions`	ListOfExtensions type	No

6.3 Dataset metadata

Dataset metadata is associated to a genomic analysis and is stored in the dataset's metadata element. A dataset metadata element overwrites the corresponding element whose values differ from the one indicated at the dataset group level (i.e., the new value in the dataset is a specialization of the value at the dataset group level).

Table 4 specifies the core elements for dataset metadata. No elements are mandatory since they are inherited (unless overwritten with new values) from the dataset group metadata.

For example, we might have datasets for patients A, B and C; therefore the dataset group's metadata includes a list of samples representing A, B and C. The datasets then provide only one sample description (respectively of A, B or C). The base set of elements in the dataset's metadata is the same as for the dataset group, but the elements are not mandatory (so there is no need to repeat them), since per default their values are considered equal to the values indicated in the dataset group. This is always the case for the values belonging to the core set, or by default for the extensions except for those cases that have the inheritance parameter set to false.

As in the case of the dataset group metadata, the information is represented as an XML document, the schema of which is derived from Table 4, using the previously described methodology. The resulting schema is provided in Annex A.2.

Table 4: Dataset's metadata core set

Element name	Element type	Description	Mandatory
Title	String		No
Type	Controlled vocabulary		No
Abstract	String		No
Project centres	ListOfProjectCentres type	Contact information of centres participating in the generation of the described study's data.	No
Description	String		No
Samples	ListOfSamples type	Identification of the samples, based on taxonomy/scientific name, common name or anonymized name and further attributes defined in a controlled library.	No
Extensions	ListOfExtensions type		No

Also as in the case of the dataset group, the extension mechanism is available to include new attributes where necessary. See section on extensions for an example in the case of dataset's metadata.

6.4 Mechanism for extensions of the metadata set

A mechanism for adding new elements to the different core metadata sets (dataset group and dataset levels) is provided.

An extended element consists of:

- information type identifier (provided in the form of a URI),
- value.

In the case of extensions at the dataset group level, a fourth value, the inheritance flag of type Boolean is optionally present. By default, and even if not present, it is considered to be equal to True. In case of being set to True, the value of the extension is inherited by the datasets belonging to the group. If set to False, the value only applies to the dataset group.

The extension schema is defined in Annex A.1. It is used in both the Dataset group and Dataset schema, however the inheritance flag is meaningless in Dataset.

Through the use of extensions, the core metadata sets can be adapted to multiple use cases. The standard defines profiles (see Clause 6.5), which rely on well-known extensions, defined in the standard and for which the URI pointer is known. To be compliant with a profile specified in Part 3 of this Standard, a tool has to implement the list of extensions included in the profile.

6.4.1 Examples of extensions

This sub-clause presents two examples:

- For dataset group, based on currently existing metadata sets, as those from ENA, EGA, NCBI or others.
- For dataset, using the concept of label from Part 1.

6.4.1.1 Example for Dataset group metadata extensions

In order to formalize the support of the broad sets of attributes used by the SRA schema [2] or NCBI [3] specifications, the extensions mechanism could be used. For example, in the case of SRA's sample metadata [2], the value of the pointer to the semantics would link to an additional schema defining the set of elements presented in Table 5 (taken over the schema provided by EBI in [2]). In this case, the fact that the semantic specification is provided as an XML schema would simplify an automatic integration of the content. The value of the extension would be a string containing the XML file name.

Table 5: `Sample`'s metadata element extended for a specific profile

Field name	Field type	Mandatory
Sample Name – scientific	String	No
Sample Name – common name	String	No
Sample Name – anonymized name	String	No
Sample Name – individual name	String	No
Description	String	No
Links	URI	No

In the case of NCBI's BioSample metadata, the specification is split in multiple cases [3]. Each of these subtypes is a different extension, the definition of which is constructed on the same principles: one XML element per attribute, using the data types indicated in the specification.

Although BioSample provides compatibility with the Minimum Information about any (x) Sequence (MIxS) [4], extensions dedicated to MIxS could be also specified, once again using the same strategy.

6.4.1.2 Example for Dataset metadata extensions

Part 1 of this Standard introduces the concept of dataset's `label`, which allows giving unique names to regions of the data. As such, this does not allow documenting what that region represents. A possible extension to the `sample` metadata is a translation tool from the label name to an ontology term, using the information indicated in Table 6.

Table 6: `Sample`'s metadata extended with a `label` linked to an ontology

Field name	Field type	Mandatory
Label name	Integer	Yes
Ontology term	URN	Yes

6.5 Metadata Profiles

Profiles are specific metadata sets. They are specified using the mechanisms provided in Clause 6.5.1. Clause 6.5.2 provides a formalized profile.

A profile corresponds to a well-known metadata set specified or used out of this standard, such as the one defined to support the Run sets of the SRA (Sequence Read Archive) schema [2].

A profile includes a subset of the core elements described in this standard, and a set of new elements specified with the extensions mechanism (see Clause 6.4).

6.5.1 Metadata profiles specification

The metadata schemas Annex A.1 and A.2 define an attribute in the Dataset Group and Dataset metadata XML element, to define the profile being used. The profile is identified with a URI, in case where no profile is active, the attribute is not used.

6.5.2 Example of metadata profile: Run

The MPEG-G dataset metadata shares characteristics with both the concepts of run and analysis, for example as used by EGA [12]. This example of MPEG-G profile ("Run") provides interoperability with existing metadata schemas.

Table 7 presents the set of elements included in this profile. Some of them are already part of the core metadata set, and the rest are the extensions needed to match the elements specified in [2].

Table 7: Metadata elements of the Run profile

Element name		Element type	Extension description
`Title`		`String`	
`Type`		Controlled vocabulary	
`Abstract`		`String`	
`Project centres`		`ListOfProjectCentres` type	
`Description`		`String`	
`Samples`		`ListOfSamples` type	
`Extensions`		`ListOfExtensions` type	
	`Spot`	`SRA.common.xsd/ SpotDescriptorType`	*The SPOT_DESCRIPTOR specifies how to decode the*

			individual reads of interest from the monolithic spot sequence. The spot descriptor contains aspects of the experimental design, platform, and processing information. There will be two methods of specification: one will be an index into a table of typical decodings, the other being an exact specification.
	`Platform`	`SRA.common.xsd/ PlatformType`	*The PLATFORM record selects which sequencing platform and platform-specific runtime parameters. This will be determined by the Center.*
	`Processing`	`SRA.common.xsd/ ProcessingType`	
	`Related links`	`SRA.run.xsd/ RUN_LINKS`	*Links to resources related to this RUN or RUN set (publication, datasets, online databases).*
	`Attributes`	`SRA.run.xsd/ RUN_ATTRIBUTES`	*Properties and attributes of a RUN. These can be entered as free-form tag-value pairs. For certain studies, submitters may be asked to follow a community established ontology when describing the work.*

The descriptions of the extensions are taken from the definition of schemas for `run` and `common` in [2].

This profile extends the MPEG-G dataset core metadata to match the run schema used by EGA. The file element (that points towards the file from within the metadata) from SRA's run metadata schema is not needed in MPEG-G because the metadata is placed within the element it refers to. Further constraints need to be applied to ensure the compatibility between the external metadata set and this profile: namely the description element in the dataset group's metadata schema has to be mandatory, and in this profile the TaxonID field can only be equal to 9096 (human).

7 Protection metadata

7.1 Introduction

Part 1 of this Standard defines containers (or `gen_info` structures) to support the protection of the information at the different layers of the hierarchy. These containers provide information to guarantee, if desired, the confidentiality and integrity of the information, alongside the privacy rules to be applied to the information they refer to. The protection `gen_info` elements are constructed as XML content, the root element of which is of type "Protection". The specific XML Schemas are included as Annex A.3 for Datasetgroup's protection and A.4 for dataset's protection.

This clause is divided into the three main aspects of protection: encryption, privacy rules and integrity (in all cases, of `gen_info` and payload structures). For the first and third aspects, the sub-clauses are further divided into the cases concerning containers and headers.

7.2 Encryption of elements in the format structure and genomic data

This sub-clause specifies the details on how the encryption parameters are conveyed in the protection of the `gen_info` elements specified in Part 1 of this Standard. The *protection* `gen_info` structure conveys the information on how its sibling boxes and the protection boxes of a layer below are encrypted. This information is represented with a list of *xenc:EncryptedData*, as specified in [9]. The data reference element of the XML Encryption tag (*xenc:EncryptedData*) uses the same set of resources identifiers. These references are constructed using the URI syntax described in section 7.4.1 of this Part of the Standard. If an element is encrypted, then it has to be listed with its corresponding *xenc:EncryptedData* element, and the payload of its box is replaced by the corresponding ciphertext (obtained applying the steps described in the *xenc:EncryptedData* element) prepended with the Initialization Vector (IV) used. The `gen_info` identifier and length cannot be encrypted, but the length has to be corrected to take into consideration any size variation between plaintext and ciphertext plus IV.

In the case of encrypting blocks (which are not gen_info structures), the URI specifies three parameters (type, id_start, id_last). The blocks of the Access unit of the type indicated, within an Id greater or equal than id_start, and smaller than id_last are concatenated and encrypted. The cipher bytes corresponding to a specific block are stored at the original location of the block. If the encryption method chosen requires to store an IV, the IV is prepended to the cypher of the first block.

7.3 Privacy Rules for the use of the genomic information

In the *protection* structure, the privacy rules tag has to be a valid policy element specified according to the XACML specification [10]. Exporting this tag as the root element of a new document, a valid policy document is obtained.

At the dataset group and dataset levels, the privacy rules indicate which rules are at hand for sibling boxes and for the protection boxes of a layer below, which are identified with the URIs listed in sub-clause 7.4.1.

The URI indicated in clause 7.4.1 for blocks encryption is used to define privacy rules for specific regions of the genome. When a rule uses such URI, the domain where it applies includes all Access Units of the type indicated in the URI, with an Id greater or equal than id_start, and smaller than id_last.

7.4 Digital Signature of `gen_info` structure

7.4.1 General case

At each level, the protection container may include authentication information in the form of a digital signature. This includes signing a subset of the `gen_info` elements listed in Table 8.

Table 8: Definition of `gen_info` elements

Protection `gen_info` at level	Can sign the content of
Dataset group (`dgcn`)	• Dataset group header (`dghd`) • Reference genome (`rfgn`) • Dataset group's metadata (`dgmd`) • All Dataset protection within the dataset group (`dtpr`)
Dataset (`dtcn`)	• Dataset header (`dthd`) • Master index table (`mitb`) • Parameters sets (`pars`) • Dataset's metadata (`dtmd`) • All Descriptors stream protection within the dataset (`dspr`) • All Access unit protection within the dataset (aupr) • All Blocks within the dataset
Descriptors stream (`dscn`)	• Descriptors stream header (`dshd`) • Descriptors stream's metadata (`dsmd`)
Access units (`aucn`)	• Access unit header (`auhd`) • Access unit information (`auin`)

Each signature is provided as an XML detached signature [8]. No canonicalization of the data is performed: the input of the authentication algorithm is the byte stream as stored on the storage medium following the standard [6]. The reference URI's are constructed as described in Table 9.

Table 9: URI construction

Protection box at level	Content to point to	URI construction:
Dataset group (`dgcn`)	`dghd`	`<file URI>/datasetgroup/{id}/header`
	`rfgn`	`<file URI>/datasetgroup/{id}/refgen`
	`dgmd`	`<file URI>/datasetgroup/{id}/metadata`
	`dtpr`	`<file URI>/datasetgroup/{id}/dataset/{d_id}/protection`
Dataset (`dtcn`)	`dthd`	`<file URI>/datasetgroup/{id}/dataset/{d_id}/header`
	`mitb`	`<file URI>/datasetgroup/{id}/dataset/{d_id}/mitb`

	pars	<file URI>/datasetgroup/{id}/dataset/{d_id}/pars
	dtmd	<file URI>/datasetgroup/{id}/dataset/{d_id}/metadata
	dspr	<dataset URI>/destream/{id}/protection
	Blocks	<dataset URI>/blocks/{type AU}/{id_start}/{id_end}
Descriptors stream (dscn)	dshd	<dataset URI>/destream/{id}/header
	dsmd	<dataset URI>/destream/{id}/metadata
Access unit (aucn)	auhd	<dataset URI>/aunit/{id}/header
	auin	<dataset URI>/aunit/{id}/info
	aupr	<dataset URI>/aunit/{id}/protection

The content to sign for each box corresponds to the payload in each gen_info structure, without including the Key and the Length.

In the case of signing blocks (which are not gen_info structures), the URI specifies three parameters (type, id_start, id_last). The blocks of the Access unit of the type indicated, within an Id greater or equal than id_start, and smaller than id_last are concatenated and signed. The resulting signature is stored in the signature element.

There are no requirements on which boxes to sign and each element can be signed multiple times.

7.4.2 Authenticity of the dataset group protection gen_info

Optionally, an enveloped signature can be provided, which will be located within the protection tag of the dataset group gen_info, such that the rest of the Protection tag is authenticated.

8 SAM interoperability

This clause aims at providing backward compatibility with the SAM format specification da805be [6].
[Note: This clause needs to be updated to reflect the last version of Part 2 of this Standard.]

In this specification a Key, Length, Value format is used for the data structures defined in this document.

8.1 SAM Header

The information contained in a SAM file header can be encoded in the DT_metadata `gen_info` structure defined in Part 1 of this standard. To store this information we define a new extension for metadata. Clauses 8.1.1-8.1.5 summarizes the fields needed. The extension can be found at clause 8.1.6.

8.1.1 HD field

Table 10: HD Field definition

Type	Description	SAM header tag
`char[Length]`	Format version. Accepted format: /^[0-9]+\.[0-9]+$/.	`VN`
`uint8`	Sorting order of alignments. Valid values: • 0x00: unknown (default), • 0x01: unsorted, • 0x02: queryname, • 0x03: coordinate. For coordinate sort, the major sort key is the RNAME field, with order defined by the order of @SQ lines in the header. The minor sort key is the POS field. For alignments with equal RNAME and POS, order is arbitrary. All alignments with `*' in RNAME field follow alignments with some other value but otherwise are in arbitrary order.	`SO`
`uint8`	Grouping of alignments, indicating that similar alignment records are grouped together but the file is not necessarily sorted overall. Valid values: • 0x00: none (default), • 0x01: query (alignments are grouped by QNAME), • 0x02: reference (alignments are grouped by RNAME/POS).	`GO`

8.1.2 SQ section

SN tag

The SN tag is replaced by the Ref_ID field in the Dataset Header.

LN tag

When transcoding from SAM to this standard, the LN tag values shall be used to validate the provided references to be encoded in the Dataset Header.

When transcoding from this standard to SAM, the value of the LN tags shall be calculated from the retrieved reference.

AS tag

This is encoded in the Reference_genome field of the Reference Genome `gen_info` defined in Part 1 of this standard.

M5 tag

When transcoding from SAM to this standard, the value of the MD5 checksum shall be replaced with the SHA256 checksum as defined in Part 1 of this standard.

When transcoding from this standard to SAM, the MD5 checksum shall be re-calculated.

UR tag

The URI of the sequence shall be encoded in the Ref_URI field of the Reference Genome `gen_info` defined in Part 1 of this standard.

SP tag

Table 11: SP tag description

Type	Description	SAM header tag
`char[Length]`	Species	`SP`

8.1.3 Read Group (RG)

Table 12: Read Group definition

Type	Description	SAM header tag
`char[Length]`	Read group identifier. Each read group must have a unique identifier. The value of this field is used in the 0x001c auxiliary field of alignment records. Must be unique among all read groups in header section. These fields may be modified when merging SAM files in order to handle collisions.	`RG-ID`
`char[Length]`	Name of sequencing center producing the read.	`CN`
`char[Length]`	Description	`DN`
`char[Length]`	Date the run was produced (ISO8601 date or date/time).	`DT`

char[Length]	Flow order. The array of nucleotide bases that correspond to the nucleotides used for each flow of each read. Multi-base flows are encoded in IUPAC format, and non-nucleotide flows by various other characters. Format: /*|[ACMGRSVTWYHKDBN]+/	FO
char[Length]	The array of nucleotide bases that correspond to the key sequence of each read.	KS
char[Length]	Library	LB
char[Length]	Programs used for processing the read group.	PG
uint32	Predicted median insert size.	PI
char[Length]	Platform/technology used to produce the reads. Valid values: CAPILLARY, LS454, ILLUMINA, SOLID, HELICOS, IONTORRENT, ONT, and PACBIO.	PL
char[Length]	Platform model. Free-form text providing further details of the platform/technology used.	PM
char[Length]	Platform unit (e.g. flowcell-barcode.lane for Illumina or slide for SOLiD). Unique identifier.	PU
char[Length]	Sample. Use pool name where a pool is being sequenced.	SM

8.1.4 Program Records (PG)

Table 13: Program Records

Type	Description	SAM header tag
char[Length]	Program record identifier. The value of this identifier is used in the alignment 0x001e field and 0x14 fields of other program records. Program record identifiers may be modified when merging SAM files in order to handle collisions.	PG-ID
char[Length]	Program Name	PN
char[Length]	Command Line	CL
char[Length]	Previous program record identifier. Must match another 0x11 field. Program records may be chained using 0x14 fields, with the last record in the chain having no 0x14 field. This chain defines the order of programs that have been applied to the alignment. Values of the 0x14 field may be modified when merging SAM files in order to handle collisions of Program record identifiers. The first Program Record in a chain (i.e. the one referred to by the PG tag in a SAM record) describes the most recent program that operated on the SAM record. The next program record in the chain describes the next most recent program that operated on the SAM record. The Program record identifier	PP

	on a SAM record is not required to refer to the newest program record in a chain. It may refer to any program record in a chain, implying that the SAM record has been operated on by the program in that PG record, and the program(s) referred to via the 0x14 field.	
`char[Length]`	Description	`DS`
`char[Length]`	Program version	`VN`

8.1.5 Comments (CO)

Table 14: Comments Field Definition

Type	Description	SAM header tag
`char[Length]`	Text comment.	`CO`

8.1.6 SAM interoperability extension

In the SAM specification, the header of the file defines fields for metadata information. In order to make the round trip possible (from SAM to MPEG-G and back to SAM), the following extension defines a mechanism for MPEG-G to contain the information present in the SAM header, in order to populate the same section in case of exporting the content back to the SAM format.

Table 15: SAM interoperability extension

Field name		Field type	Mandatory
SAM_format_version		String /^[0-9]+\.[0-9]+$/	No
sorting_order		Restricted vocabulary: unknown, unsorted, queryname, coordinate	No
grouping alignment		Restricted vocabulary: none, query, reference	No
species		String	No
read_group (multiple allowed)			No
	read_group_identifier	String (uniqueness constraint)	Yes
	sequencing_center	String	No
	Description	String	No
	date_run	Date/Time	No

	flow_order	String /*|[ACMGRSVTWYHKDBN]+/	No
	key_sequence	String	No
	Library	String	No
	programs	String	No
	predicted_median_insert_size	Integer	No
	Platform	Restrited values: CAPILLARY, LS454, ILLUMINA, SOLID, HELICOS, IONTORRENT, ONT, and PACBIO	No
	platform_model	String	No
	platform unit	String	No
	Sample	String	No
programs (multiple allowed)			No
	program:identifier	String	Yes
	program name	String	No
	command_line	String	No
	previous_program	String (value must be listed in some program_identifier field)	No
	description	String	No
	program_version	String	No
Comments		String	No

8.2 Auxiliary fields mapping

This section aims at providing backward compatibility with the specification of the optional fields in the alignment section of the SAM format specification. The information is stored in the `AU_info` `gen_info` element defined in Part 1, as a sequence of `gen_aux` structures as defined below. The `read_identifier` used in the `gen_aux` corresponds to the `read_name`. The `gen_tag` associate for each segment associated to a `read_identifier` the auxiliary fields. The information is stored in the order of appearance of the segments in the Access Unit.

```
struct gen_aux
{
      string     read_identifier;
      uint8      segments;
      gen_tag[]  aux_fields[segments];
}
```


The array of `gen_tag` elements specify which auxiliary fields apply to the read segment specified by the `read_identifier`, segment tuple. The possible values for the key element are listed in Clause 8.2.1.

```
struct gen_tag
{
      char        Key[2];
      uint64      Length;
      uint8       Value[];
}
```

The fields in bold encode information already encoded in Part 2 of this standard. If the information associated to the field does not match the one encoded according to Part 2, priority should be given to the latter.

Key values from 0x0000 to 0x03ff are reserved to fields corresponding to the tags defined in the SAM specification, while values from 0x0400 to 0xffff are reserved for user defined fields.

Table 16: Key values range

Key values range	Scope
0x0000 – 0x03ff	Reserved for SAM tags
0x0400 – 0xffff	User defined fields

8.2.1 SAM auxiliary fields

This sections lists the elements to be used to support SAM auxiliary fields.

Table 17: Keys for SAM auxiliary fields

Key	Type	Description
`AM`	uint8	The smallest template-independent mapping quality of segments in the rest.
`AS`	uint8	Alignment score generated by the aligner
`BQ`	char[Length]	Offset to base alignment quality (BAQ), of the same length as the read sequence. At the i-th read base, $BAQ_i = Q_i - (BQ_i - 64)$ where Qi is the i-th base quality.
`BD`	char[Length]	Indels quality scores
`BI`	char[Length]	Indels quality scores
`E2`	char[Length]	The 2nd most likely base calls. Same encoding and same length as QUAL.
`FS`	char[Length]	Segment suffix. It identifies different readouts from the same template, e.g. if the read was read out from the forward or reverse

		strand.
PQ	uint8	Phred likelihood of the template, conditional on both mappings being correct.
SM	uint8	Template independent mapping quality
U2	char[Length]	Phred probability of the 2nd call being wrong conditional on the best being wrong. The same encoding as the quality values.
UQ	uint8	Phred likelihood of the segment, conditional on the mapping being correct.
LB	char[Length]	The library from which the read has been sequenced. *If the DT_Metadata structure contains a list of Libraries, this field must match one of the Libraries present in the DT_metadata structure as defined in section 8.1.3 of this document.*
PG	char[Length]	Value matches the header PG-ID tag if @PG is present.
PU	char[Length]	The platform unit in which the read was sequenced. If @RG headers are present, then platform unit must match the RG-PU field of one of the headers.
CO	char[Length]	Free-text comments
BC	char[Length]	Barcode sequence, with any quality scores stored in the 0x0022 field.
QT	char[Length]	Phred quality of the barcode sequence in the 0x0021 (or 0x0023) tag. Same encoding as the quality values.
RT	char[Length]	Deprecated alternative to 0x0021 field originally used at Sanger.
OC	char[Length]	Original CIGAR string, usually before realignment.
OP	uint64	Original mapping position, usually before realignment.
OQ	char[Length]	Original base quality, usually before recalibration.
CT	char[Length]	*strand ;type (;key (=value))** Complete read annotation tag, used for consensus annotation dummy features. The CT tag is intended primarily for annotation dummy reads, and consists of a strand, type and zero or more key=value pairs, each separated with semicolons. The strand field has four values as in GFF3 (GenericFeature Format v3) [1] and supplements FLAG bit 0x10 to allow unstranded (`.'), and stranded but unknown strand (`?') annotation. For these and annotation on the forward strand (strand set to `+'), do not set FLAG bit 0x10. For annotation on the reverse strand, set the strand to `-' and set FLAG bit 0x10. The type and any keys and their optional values are all percent encoded according to RFC3986 to escape meta-characters `=', `%',

		`;', `|' or non-printable characters not matched by the isprint() macro (with the C locale). For example a percent sign becomes `%2C'.
PT	char[Length]	*start ;end ;strand ;type (;key (=value))*(|start ;end ;strand ;type (;key (=value))*)** Read annotations for parts of the padded read sequence. This field value has the format of a series of tags separated by `|', each annotating a sub-region of the read. Each tag consists of start, end, strand, type and zero or more key=value pairs, each separated with semicolons. Start and end are 1-based positions between one and the sum of the M/I/D/P/S/=/X CIGAR operators, i.e. sequence length plus any pads. Note any editing of the CIGAR string may require updating this field coordinates, or even invalidate them. As in GFF3, strand is one of `+' for forward strand tags, `-' for reverse strand, `.' for unstranded or `?' for stranded but unknown strand. The type and any keys and their optional values are all percent encoded as in the 0x0027 field.
FZ	uint16[Length/2]	Flow signal intensities on the original strand of the read, stored as (uint16) round(value * 100.0).
CM	uint32	Edit distance between the color sequence and the color reference (see also 0x0013)
CS	char[Length]	Color read sequence on the original strand of the read. The primer base must be included.
CQ	char[Length]	Color read quality on the original strand of the read. Same encoding as the quality values; same length as 0x002b.

8.2.2 User defined fields

The key values in the range 0x0100 – 0xffff can be used for user-defined fields such as those defined in the SAM specification as tags starting with 'X', 'Y', 'Z'. [Update according to previous changes]

8.3 Transcoding to/from SAM

This section describes how transcoding, to/from SAM, genomic data compliant with Part 2 of this standard shall be performed when a unique mapping in both directions is not possible due to ambiguities of the SAM file.

8.3.1 SAM Flags

This section contains a list of wrong SAM flags configuration that express alignment characteristics that cannot be associated at the same time to one mapped read or read pair.

The values of SAM flags according to the SAM specification this document refers to are reported in Table 18.

Table 18: SAM flags values

Int value	Hex value	Description
1	0x1	template having multiple segments in sequencing
2	0x2	each segment properly aligned according to the aligner
4	0x4	segment unmapped
8	0x8	next segment in the template unmapped
16	0x10	SEQ being reverse complemented
32	0x20	SEQ of the next segment in the template being reverse complemented
64	0x40	the first segment in the template
128	0x80	the last segment in the template
256	0x100	secondary alignment
512	0x200	not passing filters, such as platform/vendor quality controls
1024	0x400	PCR or optical duplicate
2048	0x800	supplementary alignment

8.3.1.1 Flag 151

The value 151 for the SAM flags corresponds to the case where the read is supposed to be at the same time mapped in a proper pair AND unmapped.

In this case the other SAM fields providing information on the read mapping (POS, MPOS, RNAME, CIGAR, RNEXT) shall be parsed to evaluate if they are concordant and represent a properly mapped read.

If the alignment information is consistent with the read sequence the 0x4 flag shall be ignored.

This choice is justified by the fact that one single flags is contradicting several SAM fields describing consistently a proper mapping and can therefore be supposed to have been wrongly generated by the aligner.

8.3.2 Unmapped mate with PNEXT value

A SAM record with the 0x8 flag set (next segment unmapped) may present a valid value for the PNEXT field. In case the SAM record containing the next segment (read pair) contains an unmapped read (flag 0x4 set and no mapping information) the PNEXT value in the first record should be discarded and the correct transcoding to SAM from MPEG shall set PNEXT = 0.

8.3.3 Duplicate records

In some SAM/BAM files identical records may appear as in the example shown below. The two records are identical and refer to a mate that is present only as a single record. The data are therefore inconsistent as they do not represent two separate pairs but one pair with one duplicate read.

In this case one of the replicated reads shall be discarded.

```
HSQ1004:134:C0D8DACXX:4:1305:12191:72218        99      chr1    247278658       60      101M    =       247278690       133
TCGGGGGAAGCCCAGGGATCTCTGTCACTGGGATCTCTGTCAGTGAGACAGTCAACTGTGATGCAGGCACCCCAGGGGGCCAGAGGCCAGGACAGCAGTGG
CCCFFFFFHHGHHJJJJJJJJJIJJJJJJJJJJJJJJJJJJJJIJHHHHEHFFFFFFEDEDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD>   RG:Z:NA12878
XT:A:U  NM:i:1  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:1  XO:i:0  XG:i:0  MD:Z:49C51

HSQ1004:134:C0D8DACXX:4:1305:12191:72218        99      chr1    247278658       60      101M    =       247278690       133
TCGGGGGAAGCCCAGGGATCTCTGTCACTGGGATCTCTGTCAGTGAGACAGTCAACTGTGATGCAGGCACCCCAGGGGGCCAGAGGCCAGGACAGCAGTGG
CCCFFFFFHHGHHJJJJJJJJJIJJJJJJJJJJJJJJJJJJJJIJHHHHEHFFFFFFEDEDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD>   RG:Z:NA12878
XT:A:U  NM:i:1  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:1  XO:i:0  XG:i:0  MD:Z:49C51

HSQ1004:134:C0D8DACXX:1:2306:12242:78140        1107    chr1    247278690       60      101M    =       247278658       -
133     ATCTCTGTCAGTGAGACAGTCAACTGTGATGCAGGCACCCCAGGGGGCCAGAGGCCAGGACAGCAGTGGATCCTGGGATAGGATGAGAATTATTTTGGCTG
:CCC@::(CCDDCCCCDDDCC>@CA>@EDEEFEDB=;HGGJHJIIIGGGHHGIJJJJJIIIIJJJJJJJJJJIJJJJIJJJJJJJJJJHHHGGFFFFF@CC   RG:Z:NA12878
XT:A:U  NM:i:1  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:1  XO:i:0  XG:i:0  MD:Z:17C83

HSQ1004:134:C0D8DACXX:4:1305:12191:72218        147     chr1    247278690       60      101M    =       247278658       -
133     ATCTCTGTCAGTGAGACAGTCAACTGTGATGCAGGCACCCCAGGGGGCCAGAGGCCAGGACAGCAGTGGATCCTGGGATAGGATGAGAATTATTTTGGCTG
DDDDDDDDDDDDDDDCDDDDDDDDDEEEEEEFFFFEBJIHIIJJJJJJHHGJJJJJJJJIIJJIJJIJJJJJJJJJJJJJJJJJJJJJHHHHHFFFFFCCC   RG:Z:NA12878
XT:A:U  NM:i:1  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:1  XO:i:0  XG:i:0  MD:Z:17C83
```

8.3.4 SAM headers errors

In some cases the names of sequences in the SAM file header don't match the reads. In the example below, the SAM file header says that the reads use a reference sequence named "chr11_gl000202_random". Note that the third fields for the two SAM records shown below have respectively the values "chr11" and "chr12". When transcoded to MPEG-G records, priority will be given to the information contained in the SAM record, if no reference labelled with the value carried by the third field of the SAM record (i.e. "chr11" and "chr12" in the example) is found, then the transcoder shall generate an error and skip the SAM record as corrupted.

A transcoding tool from SAM to MPEG-G cannot fix any SAM inconsistency found in a SAM record. Tools exists to try to do this in the SAM content itself before transcoding can take place [ref.].

```
MICHAELJACKSON_0007:5:110:10401:1393#0        89        chr11     134801779     50        76M     *     0     0
TCCTGCTTTAGAAATCCAGAAATTGGGAGGCCGAGGCAGGTAGATCATGAGGTCAGGAGATCAAGACCATCCTGGC
HEGEEEB>CGFGGGBDGG2EGE@DD@GGEBGBHHHDHHHHGHEHHHHFHHHG@GGBHHDGHGGGGGEHHGHHHHHH   AS:i:0   XN:i:0   XM:i:0   XO:i:0
XG:i:0   NM:i:0   MD:Z:76 YT:Z:UU NH:i:1   XS:A:+

MICHAELJACKSON_0007:5:42:9610:3853#0        97        chr12     60101     50        76M     =     60355     330
GTCCATTCCCTAGAAGGCTGGCTGCCCCTGGGGATGTTTTGCACCAAGCCACTGTCTCCAGCTGGGGACTAGCATC
HHHFHHHHHHGHHHHGHH>HHHBHHHHHDHDGGCEHCHDHFFF@FFHDEEGGCEDEDDCA@@@@B@@>B<AAAA>>   AS:i:0   XN:i:0   XM:i:0   XO:i:0
XG:i:0   NM:i:0   MD:Z:76 YT:Z:UU NH:i:1   XS:A:-
```

8.3.5 Mapping position error

In test item 09 from the MPEG-G database the SAM record below can be found:

```
XOH00:00970:02945       0       MT      12017   19      72M     *       0       0
      CACCCACCACATTAACAACATAAAACCCTCATTCACACGAGAAAACACCCTCATGTTCATACACCTATCCCC        ABA@=774654<4-
44,4:5444(3=2::><:66545=<5555(4::>474744<9<>>=:==8200000&        XA:Z:map2-1     MD:Z:72 XE:i:3  XF:i:1
```

```
PG:Z:tmap       RG:Z:ID NM:i:0  AS:i:72 XS:i:60
```

According to the information contained in the fourth SAM field, the read is supposed to map to position 12016 (0-indexed) in the reads. However, in the available reference genome, the read actually maps perfectly to position 12015 (0-indexed). A transcoding tool from SAM to MPEG-G cannot fix any SAM inconsistency found in a SAM record. Tools exists to try to do this in the SAM content itself before transcoding can take place [ref.].

8.3.6 Unmapped reads with reference and position values set

In the case the original SAM file had unmapped reads with a position or a reference set (despite being unmapped), these values are lost. In case of half mapped pairs, the conversion from MPEG-G to SAM might not respect the recommended practice of setting RNAME and POS to the values of the mapped one.

9 APIs to MPEG-G data

9.1 Introduction

This clause contains the API specification. It specifies which actions are to be available in a tool implementing Part 3 of the standard. The actions are divided in different types: methods to get information, and to set them; methods to create new elements or update them; methods to list elements or search them according to certain criteria. Furthermore, the API defines the methods which should be available to use the protection mechanisms: methods such as verifying the authenticity of the data, verifying the compliance with the privacy rules or decrypting the elements.

9.2 API specification

In order to facilitate access to and manipulation of MPEG-G compliant genomic content and the fields it contains, an Application Programming Interface (API), which could be implemented locally or remotely, is specified.

The operations provided by this API affect different aspects of genomic information and its associated metadata, protection information and other fields contained at each level. By level we understand File, Dataset Group, Dataset, Access Unit and MPEG-G Record, as described in Part 1 of this Standard. They may include functionalities such as providing access, performing modifications, authorizing operations or integrity verification. Each level specifically defines the functionality of each operation.

Table 19 shows a classification of the different kind of operations defined in the API. Each operation category may contain different operations, depending on the information available for each level of genomic information.

Table 19: Operations classification

Category	Description
Access	Gives the requested information to the user.
Stream	Streams the requested information to the user.
Modification	Changes the information indicated by the user.
Authorization	Checks that the user has permission to perform an operation.
Verification	Checks the integrity of some information indicated by the user.
Conversion	Converts some information from / to MPEG-G to other existing GIR formats.
Beacon-like	Provides information about MPEG-G in the form of beacons (statistical, appearance, etc.) [11].

To facilitate implementation of the API operations, several groups are defined. This facilitates interoperability with other existing API's, like the one defined by GA4GH [11]. The groups are divided into a core group and several extensions groups, as specified below.

The definition of the groups is related to the categories defined in Table 19. The groups are:

- Access (core operations). These operations allow getting information from one or more fields, listing and search. This is the core group of this specification.
- Access (additional operations). They also allow getting information from MPEG-G specific data structures.
- Stream. These operations provide streaming capabilities.
- Modification. These operations modify the structure of the genomic information, including addition and update of existing information.
- Authorization. These operations check if a user is allowed to perform some operation over genomic information.
- Verification. These operations check the integrity of some information indicated by the user.
- Conversion. These operations perform conversions between MPEG-G and other existing GIR formats.
- Beacon-like. These operations provide information about MPEG-G in the form of beacons (statistical, appearance, etc.) [11].

Tables 20 to 24 briefly describe the operations considered in different categories. Specifically, Table 20 lists access operations, Table 21 lists extra access operations, Table 22 lists streaming operations, Table 23 lists modification operations and Table 24 lists the rest of foreseen operations, indicating which category they belong to.

Table 20: Access Operations - Core

Operation name	Brief description
`GetData`	Returns the content of a level, that is, the payload.
`GetHeader`	Returns the content of the complete header of the corresponding level.
`GetHeaderField`	Returns the content of a specific header field of the corresponding level.
`GetMetadata`	Returns the content of the complete metadata element of the corresponding level.
`GetMetadataField`	Returns the content of a specific metadata field of the corresponding level.
`IsSetField`	Checks if a field has a value in the corresponding level, in order to access such field using one of the access methods.
`ListData`	Lists all data, payload, contained in the corresponding level.

ListMetadata	Lists all the metadata contained in the corresponding level.
ListMetadataField	Lists all the values of a metadata field contained in the corresponding level.
SearchData	Searches for some value inside the data contained in the corresponding level.
SearchMetadata	Searches for some value inside the metadata contained in the corresponding level.
SearchMetadataField	Searches for some value inside a specific field of the metadata contained in the corresponding level.

Table 21: Access Operations – Extension

Operation name	Brief description
GetLabel	Returns the content of a specific label inside the corresponding level.
GetProtection	Returns the content of the complete protection element of the corresponding level.
GetProtectionField	Returns the content of a specific protection field of the corresponding level.
ListLabel	List labels inside the corresponding level.
ListProtection	Lists all the protection information contained in the corresponding level.
ListProtectionField	Lists all the values of a protection field contained in the corresponding level.
SearchLabel	Searches for some value inside labels in the corresponding level.
SearchProtection	Searches for some value inside the protection contained in the file.
SearchProtectionField	Searches for some value inside a specific field of the protection contained in the file.

Table 22: Streaming Operations

Operation name	Brief description
StreamData	Send stored data using streaming.

Table 23: Modification Operations

Operation name	Brief description
AddData	Adds new content at the corresponding level.
AddHeaderField	Adds a new specific header field at the corresponding level.
AddLabel	Adds a new label at the corresponding level.
AddMetadata	Adds a new metadata element at the corresponding level.
AddMetadataField	Adds a new metadata field at the corresponding level.
AddProtection	Adds a new protection element at the corresponding level.
AddProtectionField	Adds a new protection field at the corresponding level.
UpdateData	Updates the content for the corresponding level.
UpdateHeader	Updates the header of the corresponding level.
UpdateHeaderField	Updates a specific field of the header of the corresponding level.
UpdateLabel	Updates a label at the corresponding level.
UpdateMetadata	Updates the metadata element of the corresponding level.
UpdateMetadataField	Updates a metadata field in the corresponding level.
UpdateProtection	Updates the protection element of the corresponding level.
UpdateProtectionField	Updates a protection field in the corresponding level.

Table 24: Other Operations

Category	Operation name	Brief description
Authorize	Authorize	Checks if it is possible to perform an operation over some information contained in the file, applying the privacy rules defined at the corresponding level.
Beacon-like	Beacon	Allows performing remote questions in a beacon-like form.
Conversion	ConvertFrom	Converts genomic information from a specified format into MPEG-G.
Conversion	ConvertTo	Extracts information from a genomic information file and converts it to the specified format.
Verify	Verify	Checks the integrity of the corresponding level.

Table 25 contains the mapping matrix between operations and levels, indicating which operation is available at each level.

Operations are presented in alphabetical order.

Table 25: Operations matrix

Level / Operation	File	Dataset group	Dataset	Access Unit	MPEGG Record
AddData	x	x	x	x	x
AddHeaderField	x	x	x	x	
AddLabel		x	x		
AddMetadata		x	x	x	
AddMetadataField		x	x	x	
AddProtection		x	x	x	
AddProtectionField		x	x	x	
Authorize		x	x	x	
Beacon		x			
ConvertFrom		x	x		
ConvertTo		x	x		
GetData	x	x	x	x	x
GetHeader	x	x	x	x	
GetHeaderField	x	x	x	x	
GetLabel		x	x		
GetMetadata		x	x	x	
GetMetadataField		x	x	x	
GetProtection		x	x	x	
GetProtectionField		x	x	x	
IsSetField	x	x	x	x	
ListData	x	x	x	x	x
ListLabel		x	x		
ListMetadata		x	x	x	
ListMetadataField		x	x	x	
ListProtection		x	x	x	
ListProtectionField		x	x	x	
SearchData	x	x	x	x	x
SearchLabel		x	x		
SearchMetadata		x	x	x	
SearchMetadataField		x	x	x	

SearchProtection		x	x	x	
SearchProtectionField		x	x	x	
UpdateData	x	x	x	x	x
UpdateHeader	x	x	x	x	
UpdateHeaderField	x	x	x	x	
UpdateLabel		x	x		
UpdateMetadata		x	x	x	
UpdateMetadataField		x	x	x	
UpdateProtection		x	x	x	
UpdateProtectionField		x	x	x	
StreamData		x	x	x	
Verify		x	x	x	

9.2.1 REST API

Table 26 describes the Access Core operations for the Dataset level. For each of it, the corresponding row provides name, URL for a REST-based service and brief description.

The URL is constructed following the hierarchy of the MPEG-G data structures. In this specific case, to access a dataset inside a dataset group, the base URL has to be constructed as follows:

`/datasetgroup/{id}/dataset/{did}`

where

`/datasetgroup/{id}` represents the datasetgroup idenfied by {id} and

`/dataset/{did}` represents the dataset idenfied by {did}.

After this base URL, further fields and parameters can be added to indicate which specific information stored inside the dataset is to be requested, as described in Table 26.

Access operations are mapped to the GET HTTP method [8]. To define modification operations (add and update), mapping to POST HTTP method should be used.

Table 26: Access Core Operations for the Dataset level

Operation name	URL (for REST-based API's)	Description
GetHeader	GET /datasetgroup/{id}/dataset/{did}/header	Returns the content of the datasets header for the dataset identified by {did}.
GetHeaderField	GET /datasetgroup/{id}/dataset/{did}/hfield={field_name}	Returns the content of the dataset's header field with name {field_name}
GetMetadata	GET /datasetgroup/{id}/dataset/{did}/	Returns the content of the

	metadata	datasets metadata.
GetMetadataField	GET /datasetgroup/{id}/dataset/{did}/mfield={field_name}	Returns the field identified by field_name from the dataset metadata.
GetData	GET /datasetgroup/{id}/dataset/{did}	Returns the payload of the dataset identified by {did}.
ListMedatadata	GET /datasetgroup/{id}/dataset/{did}/metadata	Returns a list of active metadata fields
ListMetadataField	GET /datasetgroup/{id}/dataset/{did}/mfield={field_name}	Returns a list of active values in the metadata field
SearchMetadata	GET /datasetgroup/{id}/dataset/{did}/search_metadata?{search_criteria}	Returns a list of metadata fields matching the provided {search_criteria}
SearchMetadataField	GET /datasetgroup/{id}/dataset/{did}/search_metadataField?{search_criteria}	Returns a list of values in a metadata field matching the provided {search_criteria}

Annex A
(informative)

XML Schemas

This annex describes the XML schemas corresponding to metadata and protection elements.

A.1 Dataset group metadata dgmd XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema                    targetNamespace="urn:mpeg:mpeg-g:metadata:dataset_group:2017"
xmlns="http://www.w3.org/2001/XMLSchema"          xmlns:mpg-meta-data-gr="urn:mpeg:mpeg-
g:metadata:dataset_group:2017">

    <complexType name="ProjectCentreType">
    <sequence>
       <element name="ProjectCentreName" type="string"/>
        <element name="Extensions" type="mpg-meta-data-gr:ExtensionsType" minOccurs="0"
maxOccurs="1"/>
    </sequence>
    </complexType>

    <element name="DatasetGroup" type="mpg-meta-data-gr:DatasetGroupType"/>

    <complexType name="DatasetGroupType">
    <sequence>
       <element name="Title" type="string" minOccurs="1" maxOccurs="1"/>
       <element name="Type" type="string" minOccurs="1" maxOccurs="1"/>
       <element name="Abstract" type="string" minOccurs="0" maxOccurs="1"/>
        <element     name="ProjectCentre"     type="mpg-meta-data-gr:ProjectCentreType"
minOccurs="0" maxOccurs="1"/>
       <element name="Description" type="string" minOccurs="0" maxOccurs="1"/>
        <element name="Samples" type="mpg-meta-data-gr:SamplesType"/>
       <element name="Extensions" type="mpg-meta-data-gr:ExtensionsType"/>
    </sequence>
    <attribute name="profile" type="anyURI" use="optional"/>
    </complexType>

    <complexType name="SamplesType">
    <sequence>
       <element    name="Sample"    type="mpg-meta-data-gr:SampleType"    minOccurs="1"
maxOccurs="unbounded"/>
    </sequence>
    </complexType>

    <complexType name="SampleType">
    <sequence>
        <element name="TaxonId" type="int" minOccurs="1" maxOccurs="1"/>
       <element name="Title" type="string" minOccurs="0" maxOccurs="1"/>
        <element name="Extensions" type="mpg-meta-data-gr:ExtensionsType" minOccurs="0"
maxOccurs="1"/>
    </sequence>
    </complexType>

    <complexType name="ExtensionType">
    <sequence>
```



```
        <element name="Type" type="anyURI"/>
         <element name="Inheritable" type="boolean"/>
        <any minOccurs="1"/>
    </sequence>
    </complexType>

    <complexType name="ExtensionsType">
    <sequence>
        <element  name="Extension"  type="mpg-meta-data-gr:ExtensionType"  minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
    </complexType>
</schema>
```

A.2 Dataset metadata `dtmd` XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema                          targetNamespace="urn:mpeg:mpeg-g:metadata:dataset:2017"
xmlns="http://www.w3.org/2001/XMLSchema"          xmlns:mpg-meta-data-gr="urn:mpeg:mpeg-
g:metadata:dataset_group:2017"
xmlns:mpg-meta-dataset="urn:mpeg:mpeg-g:metadata:dataset:2017">
    <import                      namespace="urn:mpeg:mpeg-g:metadata:dataset_group:2017"
schemaLocation="DatasetGroupSchemaOneProfile.xsd"/>
    <complexType name="DatasetType">
        <sequence>
                <element name="Title" type="string" minOccurs="1" maxOccurs="1"/>
                <element name="Type" type="string" minOccurs="1" maxOccurs="1"/>
                <element name="Abstract" type="string" minOccurs="0" maxOccurs="1"/>
                <element  name="ProjectCentre"  type="mpg-meta-data-gr:ProjectCentreType"
minOccurs="0" maxOccurs="1"/>
                <element name="Description" type="string" minOccurs="0" maxOccurs="1"/>
                <element        name="Samples"        type="mpg-meta-data-gr:SamplesType"
minOccurs="1" maxOccurs="1"/>
                <element     name="Extensions"     type="mpg-meta-data-gr:ExtensionsType"
maxOccurs="1" minOccurs="0"/>
        </sequence>
        <attribute name="profile" type="anyURI" use="optional"/>
    </complexType>

    <element name="Dataset" type="mpg-meta-dataset:DatasetType"/>
</schema>
```

A.3 Dataset group protection `gen_info` XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema       attributeFormDefault="unqualified"       elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:mpeg:mpegen/protection_datasetgroup"
xmlns="urn:mpeg:mpegen/protection_datasetgroup">
  <xs:import                                  namespace="http://www.w3.org/2001/04/xmlenc#"
schemaLocation="https://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd"/>
  <xs:import                                 namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="https://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd#enveloped-signature"/>
  <xs:element name="protection" type="protectionType"/>

  <xs:complexType name="protectionType">
    <xs:sequence>
```

```
      <xs:element type="encryptionsType" name="encryptions"/>
      <xs:element type="signaturesType" name="signatures"/>
      <xs:element      type="xd:SignatureType"      name="signature"      minOccurs="0"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="encryptionsType">
    <xs:sequence>
      <xs:element     ref="xe:EncryptedData"     maxOccurs="unbounded"     minOccurs="0"
xmlns:xe="http://www.w3.org/2001/04/xmlenc#"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="signaturesType">
    <xs:sequence>
      <xs:element      ref="xd:Signature"      maxOccurs="unbounded"      minOccurs="0"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.4 Dataset protection `gen_info` XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema       attributeFormDefault="unqualified"       elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="protection" type="protectionType"/>
  <xs:complexType name="protectionType">
    <xs:sequence>
      <xs:element type="encryptionsType" name="encryptions"/>
      <xs:element type="signaturesType" name="signatures"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="encryptionsType">
    <xs:sequence>
      <xs:element     ref="xe:EncryptedData"     maxOccurs="unbounded"     minOccurs="0"
xmlns:xe="http://www.w3.org/2001/04/xmlenc#"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="signaturesType">
    <xs:sequence>
      <xs:element      ref="xd:Signature"      maxOccurs="unbounded"      minOccurs="0"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```


Bibliography

[1] L. Stein, "Generic Feature Format Version 3 (GFF3)," 2013. [Online]. Available: https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md.

[2] EBI, "Read domain XML 1.5 metadata format", Available: https://www.ebi.ac.uk/ena/submit/read-xml-format-1-5

[3] NCBI, "Preview BioSample types and attributes,". Available: https://submit.ncbi.nlm.nih.gov/biosample/template/.

[4] G. S. C. "MIxS GSC Project," 26 07 2016. [Online]. Available: http://gensc.org/projects/mixs-gsc-project/.

[5] W3C, "Extensible Markup Language (XML) 1.1 (Second Edition), August, 16th 2016, Available: https://www.w3.org/TR/xml11/

[6] SAM/BAM Format Specification Working Group, Sequence Alignment/Map format specification, 1 06 2017, Available: https://samtools.github.io/hts-specs/SAMv1.pdf

[7] IETF, Hypertext Transfer Protocol – HTTP/1.1 rfc2616

[8] W3C, XML Signature Syntax and Processing Version 1.1, 11 04 2013, Available: https://www.w3.org/TR/xmldsig-core1/

[9] W3C, XML Encryption Syntax and Processing Version 1.1, 11 04 2013, Available: https://www.w3.org/TR/xmlenc-core1/

[10] OASIS, eXtensible Access Control Markup Language (XACML) Version 3.0, 22 01 2013, Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf

[11] Global Alliance for Genomics and Health. A federated ecosystem for sharing genomic, clinical data. Science 352, 1278–1280, 2016, Available https://www.ga4gh.org/

[12] EGA, European genome-phenome archive, Available: https://ega-archive.org/

