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/M41733
October 2017, Macau, China

Source: **DMAG-UPC et al.**
Status: **Proposal**
Title: **Genomic Information Representation. Proposal for WD of Part 3 on Protection, Application Programming Interfaces and Metadata**
Authors: **Jaime Delgado, Silvia Llorente, Daniel Naro (Distributed Multimedia Applications Group – Universitat Politècnica de Catalunya),**
+ ***Other authors of previous contributions***

Contents

1 Introduction 4

2 Scope 4

3 Normative references 4

4 Terms and definitions 4

5 Character encoding 5

6 SAM interoperability 5

6.1 SAM Header 5

6.1.1 HD field 5

6.1.2 SQ section 6

6.1.3 Read Group (RG) 7

6.1.4 Program Records (PG) 7

6.1.5 Comments (CO) 8

6.2 Auxiliary fields mapping 8

6.2.1 SAM auxiliary fields 8

6.2.2 User defined fields 12

6.3 Transcoding to/from SAM 12

6.3.1 SAM Flags 12

6.3.2 Unmapped mate with PNEXT value 13

6.3.3 Duplicate records 13

6.3.4 SAM headers errors 13

6.3.5 Mapping position error 14

6.3.6 Unmapped reads with reference and position values set 14

7 Supported FASTA format 14

8 Metadata 15

8.1 Dataset group metadata 15

8.2 Dataset metadata 16

8.3 Extensions 17

8.3.1 Examples of extensions 17

8.4 Profiles 19

8.4.1 Profiles specification 19

8.4.2 Example of profile: EGA's metadata schema 19

8.4.3 Example of profile: NCBI's metadata schema 20

8.5 Access units metadata 20

9 Protection 20

9.1 Encryption 20

9.1.1 *xenc:EncryptedData* 20

9.1.2 Encryption-blocks 21

9.2 Privacy Rules 22

9.3 Digital Signature 22

9.3.1 General case 22

9.3.2 Authenticity of the dataset group protection box 23

9.3.3 Access units protection box ... 23
10 APIs ... 23
10.1 API definition ... 23
10.1.1 REST API ... 27
10.1.2 C-like API ... 34
11 References ... 34
12 Annexes ... 35
Annex I – Protection boxes XML Schemas ... 35
I.1 Dataset group protection box XML schema ... 35
I.2 Dataset protection box XML schema ... 35
I.3 Descriptor stream protection box XML schema ... 36
Annex II – Examples of XACML rules ... 37
Annex III – Examples of metadata ... 41

1 Introduction

The development of Next Generation Sequencing (NGS) 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.

This document includes the specification of

- how backward compatibility with existing SAM content is supported,
- metadata and protection elements coded in compliance with Parts 1 and 2 of ISO/IEC 23092,
- interfaces to access genomic information coded in compliance with Parts 1 and 2 of ISO/IEC 23092.

[Note: To be completed]

2 Scope

3 Normative references

4 Terms and definitions

Term	Definition
Alignment	A sequence read mapped on a reference sequence
BAM	Compressed binary version of SAM
Barcode	Pre-defined oligonucleotides appended to the beginning and/or end of a template. Used for example to multiplex several samples during sequencing.
CIGAR string	A CIGAR string is a sequence of base lengths and the associated operations used to indicate alignment differences between the sequence and the reference, such as in-step with reference (either a match or mismatch), insertion or deletion to/from the reference and trimmed sequence ends.
contig	A contig (from contiguous) is a set of overlapping DNA segments that together represent a consensus region of DNA.
CRAM	GIR that includes SAM + compression configuration
FASTA	GIR that includes read headers and sequence reads (nucleotides sequences)
FASTQ	GIR that includes FASTA + quality scores
GIR	Genomic Information Representation
Hit	Alignment result in terms of mapping position for a read on a reference sequence
Indel	An additional or missing nucleotide in a DNA sequence with respect to a reference DNA sequence.
MAF	Mutation Annotation Format. File format used to mark the genes and other biological features in a DNA sequence.
Paired reads	A couple of reads produced from the same DNA fragment by sequencing both ends.
Quality score	A quality score is assigned to each nucleotide base call in automated sequencing processes. It expresses the base-call accuracy.
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.

SAM	GIR that is human readable and includes FASTQ + alignment and analysis information
Segment	A contiguous sequence of nucleotides
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.
Template	A DNA sequence part of which is sequenced on a sequencing machine

[Note: To be completed]

5 Character encoding

This specification utilizes UTF8 character encoding.

6 SAM interoperability

This section aims at providing backward compatibility with the SAM format specification e087be0 [ref.].

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

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

6.1 SAM Header

The information contained in a SAM file header shall be encoded in the DT_metadata `gen_info` structure defined in Part 1 of this standard.
This section specifies how to use the Value field of DT_metadata to encode information present in a SAM file header. This information is encoded as `gen_tag` structures according to the syntax specified below.

6.1.1 HD field

Key	Type	Description	SAM header tag
0x0000	char[Length]	Format version. Accepted format: /^[0-9]+\.[0-9]+$/.	VN
0x0001	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	SO

		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.	
0x0002	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

6.1.2 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

Key	**Type**	**Description**	**SAM header tag**
0x0003	char[Length]	Species	SP

6.1.3 Read Group (RG)

Key	Type	Description	SAM header tag
0x0004	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
0x0005	char[Length]	Name of sequencing center producing the read.	CN
0x0006	char[Length]	Description	DN
0x0007	char[Length]	Date the run was produced (ISO8601 date or date/time).	DT
0x0008	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
0x0009	char[Length]	The array of nucleotide bases that correspond to the key sequence of each read.	KS
0x000a	char[Length]	Library	LB
0x000b	char[Length]	Programs used for processing the read group.	PG
0x000c	uint32	Predicted median insert size.	PI
0x000d	char[Length]	Platform/technology used to produce the reads. Valid values: CAPILLARY, LS454, ILLUMINA, SOLID, HELICOS, IONTORRENT, ONT, and PACBIO.	PL
0x000e	char[Length]	Platform model. Free-form text providing further details of the platform/technology used.	PM
0x000f	char[Length]	Platform unit (e.g. flowcell-barcode.lane for Illumina or slide for SOLiD). Unique identifier.	PU
0x0010	char[Length]	Sample. Use pool name where a pool is being sequenced.	SM

6.1.4 Program Records (PG)

Key	Type	Description	SAM header tag
0x0011	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
0x0012	char[Length]	Program Name	PN
0x0013	char[Length]	Command Line	CL
0x0014	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	PP

		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 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.	
0x0015	char[Length]	Description	DS
0x0016	char[Length]	Program version	VN

6.1.5 Comments (CO)

Key	**Type**	**Description**	**SAM header tag**
0x0017	char[Length]	Text comment.	CO

6.2 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 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.

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

6.2.1 SAM auxiliary fields

This sections lists the elements to be used to support SAM auxiliary fields. Key values in bold signal that the element conveys information associated to the read according to the Part 2 of this standard.

Key	**Type**	**Description**	**SAM tag**
0x0000	uint8	The smallest template-independent mapping quality of segments in the rest.	AM
0x0001	uint8	Alignment score generated by the aligner	AS

0x0002	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.	BQ
0x0003	char[Length]	Indels quality scores	BD
0x0004	char[Length]	Indels quality scores	BI
0x0005	char[Length]	Reference name of the next hit; `=' for the same chromosome.	CC
0x0006	uint64	Leftmost coordinate of the next hit.	CP
0x0007	char[Length]	The 2nd most likely base calls, same length as the corresponding read	E2
0x0008	uint8	The index of the segment in the template	FI
0x0009	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.	FS
0x000a	uint32	Number of perfect hits	H0
0x000b	uint32	Number of 1-difference hits	H1
0x000c	uint32	Number of 2-difference hits	H2
0x000d	uint64	Query hit index, indicating the alignment record is the i-th one stored in the SAM equivalent file	HI
0x000e	uint32	Number of alignments that contain the query in the current record	IH
0x000f	char[Length]	CIGAR string for the mate/next read	MC
0x0010	char[Length]	The MD field aims to achieve SNP/indel calling without looking at the reference. For example, a string `10A5^AC6' means from the leftmost reference base in the alignment, there are 10 matches followed by an A on the reference which is different from the aligned read base; the next 5 reference bases are matches followed by a 2bp deletion from the reference; the deleted sequence is AC; the last 6 bases are matches. *This field ought to match the alignment information associated to the read according to the Part 2 of this standard.*	MD
0x0011	uint8	Mapping quality of the mate/next segment	MQ
0x0012	uint32	Number of reported alignments (i.e. alignments written to the SAM file) that contain the query in the corresponding SAM record	NH
0x0013	uint32	Edit distance to the reference including ambiguous base but excluding clipping	NM
0x0014	uint8	Phred likelihood of the template, conditional on both mappings being correct.	PQ
0x0015	char[Length]	Phred quality of the mate/next segment sequence in the 0x0016 field. Same encoding as quality scores. *This field ought to match the quality values information associated to the mate/next segment according to the Part 2 of this standard.*	Q2
0x0016	char[Length]	Sequence of the mate/next segment in the template. *This field ought to match the sequence information*	R2

		associated to the read according to the Part 2 of this standard.	
0x0017	char[Length]	(rname ,pos ,strand ,CIGAR ,mapQ ,NM ;)+ Other canonical alignments in a chimeric alignment, formatted as a semicolon-delimited list. Each element in the list represents a part of the chimeric alignment. Conventionally, at a supplementary line, the first element points to the primary line. *This field ought to match the sequence information associated to the read according to the Part 2 of this standard. Once spliced alignments are specified in Part 2*	SA
0x0018	uint8	Template independent mapping quality	SM
0x0019	uint8	The number of segments in the template. *This field ought to match the sequence information associated to the read according to the Part 2 of this standard.*	TC
0x001a	char[Length]	Phred probability of the 2nd call being wrong conditional on the best being wrong. The same encoding as the quality values.	U2
0x001b	uint8	Phred likelihood of the segment, conditional on the mapping being correct.	UQ
0x001c	char[Length]	The read group to which the read belongs. If the DT_Metadata structure contains a list of Read Group Identifiers, this field must match one of the Read Group Identifiers present in the DT_metadata structure as defined in section 5.1.3 of this document.	RG
0x001d	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 5.1.3 of this document.	LB
0x001e	char[Length]	Value matches the header PG-ID tag if @PG is present.	PG
0x001f	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.	PU
0x0020	char[Length]	Free-text comments	CO
0x0021	char[Length]	Barcode sequence, with any quality scores stored in the 0x0022 field.	BC
0x0022	char[Length]	Phred quality of the barcode sequence in the 0x0021 (or 0x0023) tag. Same encoding as the quality values.	QT
0x0023	char[Length]	Deprecated alternative to 0x0021 field originally used at Sanger.	RT
0x0024	char[Length]	Original CIGAR string, usually before realignment.	OC

0x0025	uint64	Original mapping position, usually before realignment.	OP
0x0026	char[Length]	Original base quality, usually before recalibration.	OQ
0x0027	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'.	CT
0x0028	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.	PT
0x0029	uint16[Length/2]	Flow signal intensities on the original strand of the read, stored as (uint16) round(value * 100.0).	FZ
0x002a	uint32	Edit distance between the color sequence and the color reference (see also 0x0013)	CM

0x002b	char[Length]	Color read sequence on the original strand of the read. The primer base must be included.	CS
0x002c	char[Length]	Color read quality on the original strand of the read. Same encoding as the quality values; same length as 0x002b.	CQ

6.2.2 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'.

6.3 Transcoding to/from SAM

This section aims at describing how transcoding of genomic data representation compliant with Part 2 (N17076) of this standard to/from SAM shall be performed when a straightforward conversion is not possible due to ambiguities of the SAM file.

6.3.1 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 below:

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

6.3.1.1 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.

6.3.2 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.

6.3.3 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
CCCFFFFFHHGHHJJJJJJJJJIJJJJJJJJJJJJJJJJJJJJIJHHHHEHFFFFFFEDEDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD>   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
CCCFFFFFHHGHHJJJJJJJJJIJJJJJJJJJJJJJJJJJJJJIJHHHHEHFFFFFFEDEDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD>   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=;HGGJHJIIIGGGHHGIJJJJJIIIIJJJJJJJJJJIJJJIJJJJJJJJJJHHHGGFFFFF@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
```

6.3.4 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:-
```

6.3.5 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       CACCCACCACATTAACAACA
TAAAACCCTCATTCACACGAGAAAACACCCTCATGTTCATACACCTATCCCC        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.].

6.3.6 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.

7 Supported FASTA format

The FASTA format supported by this specification is represented as a series of lines in an ASCII text file.
The first line in the FASTA file shall start with a ">" (greater-than) symbol.
Each line starting with a ">" (greater-than) symbol shall be interpreted as the identifier (a.k.a. name) of the sequence of nucleotides represented by the following one or more lines.
Each line starting with a ">" (greater-than) symbol shall be followed by one or more lines of uppercase symbols representing nucleotides as defined in Clause 5.1 of Part 2 of this standard.
The following is an example of supported FASTA.

Line	Content	Description
1	`>1 dna:chromosome chromosome:GRCh37:1:1:249250621:1`	First sequence identifier
2	`ACGTTGACTATCGATCTATTAGCGGCGATGCA`	Sub-sequences of nucleotides representing the entire first sequence
3	`TGACTATCGATCTATTAGCGGCGATGCTTCCA`	
4	`ACGTTGACAAACCGATAAGCGGCGATGCAAAC`	
…	…	
N	`>2 dna:chromosome chromosome:GRCh37:2:1:243199373:1`	Second sequence identifier
N+1	`TGACTATCGATCTATTAGCGGCGATGCTTCCA`	Sub-sequences of nucleotides representing the entire second sequence
N+2	`ACGTTGACAAACCGATAAGCGGCGATGCAAAC`	
N+3	`TTGACAAACCGATAAGCGGCGATGCAAACAGT`	
…	…	
…	…	…

8 Metadata

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

The standard defines a minimum core set of metadata elements, which can then be extended by users and applications by including extra information elements. Sets are specified for a Dataset Group and for a Dataset.

Extensions to (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.

Profiles are specific metadata sets specified using mechanisms provided in the standard. A profile corresponds to a well-known metadata set specified or used out of this standard, such as those from EBI or NCBI. This allows easy interoperability with already existing systems.

A profile includes a subset of core elements described in this standard, and a set of new elements specified with the extensions mechanism (see Clause 8.4).

The rest of Clauses of this standard specify Dataset group metadata (Clause 8.1), Dataset metadata (Clause 8.2), Extensions (Clause 8.3) and Profiles (Clause 8.4).

8.1 Dataset group metadata

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

Table 1: Base dataset group's metadata core set

Element name	Element type	Mandatory
Title	String	Yes
Type	Controlled vocabulary	Yes
Abstract	String	No
Project centre name	String	No
Description	String	No
Samples	List of sample types	Yes
Extensions	List of extension types	No

The conversion of this table 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. [NOTE: Provide type's controlled vocabulary]

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 provides those for the *sample* type, and Table 3 for *project center*.

Table 2: Sample's metadata core set

Field name	Field type	Mandatory
TaxonId	Integer	Yes
Title	String	No
Extensions	List of extensions	No

Table 2: Project centre's metadata core set

Field name	Field type	Mandatory
ProjectcentreId	Integer	Yes
Title	String	No
Extensions	List of extensions	No

8.2 Dataset metadata

Dataset metadata is associated to a genomic analysis and is stored in the dataset's metadata element. A dataset metadata overwrites those elements whose values differ from the ones 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 element names for dataset metadata. No elements are mandatory since they are inherited 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 except for those cases that have the inheritance parameter set to false. [Note: Inheritance parameter to be specified.]

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.

Table 4: Base dataset's metadata

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

Annex III provides an example for datasets group metadata and dataset metadata.

8.3 Extensions

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;
- value;
- pointer to a resource documenting the semantics of the given information type: this resource provides information for auto-discovery of the extension.

8.3.1 Examples of extensions

This Clause presents two examples:

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

8.3.1.1 Example for Dataset group metadata extensions

In order to support the broad sets of attributes used by EGA or NCBI, the extensions mechanism could be used. For example, in the case of EGA's sample metadata [2], we could give as semantic reference to the extension a link to an extra schema which would define the elements presented in Table 5 (taken over the schema provided by EGA). 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.

Table 5: Sample's metadata extended to EGA's specification

Field name	Field type	Mandatory
Sample Name – taxon id	Integer	No
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 by NCBI.

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.

8.3.1.2 Example for Dataset metadata extensions

Part 1 of the standard introduces the concept of dataset's labels: they allow 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 would be 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 labels

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

8.4 Profiles

Profiles are specific metadata sets. They are specified using the mechanisms provided in clause 8.4.1 [Note: Mechanisms to be specified]. Clauses 8.4.2 to 8.4.3 provide formalized profiles. [Note: Currently, clause 8.4.2 provides an example of a possible profile.]

A profile corresponds to a well-known metadata set specified or used out of this standard, such as those from EGA or NCBI. This allows easy interoperability with already existing systems.

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 8.3).

8.4.1 Profiles specification

[Note: To be completed]

8.4.2 Example of profile: EGA's metadata schema

The MPEG-G dataset metadata shares characteristics with both EGA's run and analysis. We here introduce a MPEG-G profile ("EGArun") that maps to EGA's run schema [2].

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 EGA's run schema.

Table 7: Metadata elements of the *EGArun* profile

Element name		Element type	Extension description
Title		String	
Type		Controlled vocabulary	
Abstract		String	
Project centres		List of type project centres	
Description		String	
Samples		List of type sample	
Extensions		List of extensions	
	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 EGA's run schema. The file element (that points towards the file from within the metadata) from EGA's run 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 EGA's metadata 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).

8.4.3 Example of profile: NCBI's metadata schema

[Note: To be completed]

8.5 Access units metadata

[Note: To be completed]

9 Protection

The protection boxes are constructed as XML content, the root element of which is of type "Protection". Refer to the provided XSD files [ref.] for the concrete structure.

9.1 Encryption

9.1.1 *xenc:EncryptedData*

The protection box 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 [ref.]. 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 the section 7.4.2 of [ref.]. 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 ciphertext (obtained applying the steps described in the *xenc:EncryptedData* element) prepended with the IV used. The box 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.

9.1.2 Encryption-blocks

The XML tag `<encryption-blocks>` indicates which blocks are encrypted and using which key. Refer to [ref.] for the schema of this element. The child `profiles` tag, aggregates a collection of encryption profiles, each specified within a `profile` tag with a `KeyInfo` tag as specified in [ref.], and an `IV` element in Base64. The profile's id attribute is bounded between 1 and 255 (included) and cannot be repeated. The `KeyInfo` has to return a valid key for a AES-256 cipher. The user has to take into account the danger of reusing the same key. The `encryptionActive` element represented in Base64, stores which is the encryption profile used for each block. It is constructed as a byte array composed of a repetition of the following tuple: a byte indicates the id of the profile (the value 0 is used to indicate no encryption), followed by 4 bytes (as an unsigned integer in little endian) storing over how many blocks this profile is active.

```
<xs:complexType name="encryption-blocks">
    <xs:sequence>
      <xs:element ref="xd:KeyInfo"
                  xmlns:xd="http://www.w3.org/2000/09/xmldsig#"/>
      <xs:element type="xs:base64Binary" name="iv"/>
      <xs:element type="xs:base64Binary" name="encryption-active"/>
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="profileType">
    <xs:sequence>
      <xs:element                                          ref="xd:KeyInfo"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#"/>
      <xs:element type="xs:base64Binary" name="iv"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="id" use="optional"/>
  </xs:complexType>
  <xs:complexType name="profilesType">
    <xs:sequence>
      <xs:element           type="profileType"           name="profile"
maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="encryption-blocksType">
    <xs:sequence>
      <xs:element type="profilesType" name="profiles"/>
      <xs:element type="xs: base64Binary" name="encryption-active"/>
    </xs:sequence>
  </xs:complexType>
</xs:complexType>
```

The data in the encrypted blocks is replaced by the output of an AES-256 cipher in CTR mode initialized with the key specified in the `KeyInfo` element and the initialization vector contained in the `IV` tag. The payload of all the blocks in one stream is treated as one input for the cipher: in order to obtain the correct cipher text for one block, the cipher seeks the initial position of the plaintext in the overall stream descriptors plaintext, and encrypts it.

There is no requirement on which blocks to encrypt.

9.2 Privacy Rules

The privacy rules tag has to be a valid policy element according to the XACML specification [ref.]: by exporting this tag as the root element of a new document, the privacy handling application has to have a valid policy document.

At each level, the privacy rules indicate which rules are at hand for sibling boxes and the protection boxes of a layer below, which are identified with the URIs listed in section 7.4.2. The privacy rules have to grant access to the protection boxes to any user requesting access to some information stored in the protection box's layer or at a layer below.

At the descriptor stream level, the policy can define access rights to specific regions of the genome by specifying one of the label Ids. In order to correctly interpret the privacy rules, any reference to a label has to be translated to a new resource description model to be defined. [reference to labels in Part 1 to be added]
[Note: This new description model should contain the indexing attributes needed to resolve the privacy rules when the user submits a random access query based on the indexing.]

9.3 Digital Signature

9.3.1 General case

At each level, the protection box may include authentication information in the form of a digital signature. This includes signing a subset of the boxes listed below.

Protection box at level	Can sign the content of
Dataset group (dgcn)	• Datasets 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)
Descriptors stream (dscn)	• Descriptors stream header (dshd) • Local index table (litb) • Descriptors stream's metadata (dsmd)

Each signature is provided as an XML detached signature [ref.]. 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 [ref.]. The reference URI's are constructed as follows:

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
Descriptors stream (dscn)	dshd	<dataset URI>/destream/{id}/header
	litb	<dataset URI>/destream/{id}/litb
	dsmd	<dataset URI>/destream/{id}/metadata

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

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

9.3.2 Authenticity of the dataset group protection box

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

9.3.3 Access units protection box

[Note: to be completed]

10 APIs

This section shall contain the API definition. The current text is a first draft contribution for discussion.

10.1 API definition

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, Descriptor Stream and Block (including those for storage and streaming), as described in Part 1 of this standard (N17075). 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 7.1 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 7.1 – Operations classification

Category	Description
Access	Gives 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.) [ref.].

Tables 7.2 to 7.4 briefly describe the operations considered in different categories. Specifically, table 7.2 lists access operations, table 7.3 lists modification operations and table 7.4 lists the rest of foreseen operations, indicating which category they belong to.

Table 7.2 – Access Operations

Operation name	Brief description
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.
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.
GetLabel	Returns the content of a specific label inside a dataset.
GetDatasetGroup	Returns the content of a specific dataset group.
GetDataset	Returns the content of a specific dataset.
GetData	Returns the content of a level, that is, the payload. It can be filtered by positions and reference sequence.
isSetField	Checks if a field has a value in the corresponding level, in order to access such field using one of the access methods.
ListMetadata	Lists all the metadata contained in a file.
ListMetadataField	Lists all the values of a metadata field contained in a file.
ListProtection	Lists all the protection information contained in a file.
ListProtectionField	Lists all the values of a protection field contained in a file.
ListLabel	List labels inside a dataset.
SearchMetadata	Searches for some value inside the metadata contained in the file.
SearchMetadataField	Searches for some value inside a specific field of the metadata contained in the file.
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.
SearchLabel	Searches for some value inside labels in a dataset.
StreamData	Send stored data using streaming.

Table 7.3 – Modification Operations

Operation name	Brief description
AddHeaderField	Adds a new specific header field 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.
AddData	Adds new content at the corresponding level.
UpdateHeader	Updates the header of the corresponding level.
UpdateHeaderField	Updates a specific field of the header of 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.
UpdateData	Updates the content for the corresponding level.

Table 7.4 – 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.
Verify	Verify	Checks the integrity of the corresponding level.
Conversion	ConvertTo	Extracts information from a genomic information file and converts it to the specified format.
Conversion	ConvertFrom	Converts genomic information from a specified format into MPEG-G.
Beacon-like	Beacon	Allows performing remote questions in a beacon-like form.

Table 7.5 contains the mapping matrix between operations and levels, indicating which operation is available at each level.

Table 7.5 – Operation matrix

Level / Operation	File	Datasets Group	Dataset	Descriptor Stream	Block	Transport Block	Packet
GetHeader	x	x	x	x			
GetHeaderField	x	x	x	x			
GetMetadata		x	x	x			
GetMetadataField		x	x	x			
GetProtection		x	x	x			
GetProtectionField		x	x	x			
GetLabels			x				
GetDatasetGroup	x						
GetDataset		x					
GetData					x	x	x
AddHeaderField	x	x	x	x			
AddMetadata		x	x	x			
AddMetadataField		x	x	x			
AddProtection		x	x	x			
AddProtectionField		x	x	x			
AddLabel			x				
AddData					x	x	x
UpdateHeader	x	x	x	x			
UpdateHeaderField	x	x	x	x			
UpdateMetadata		x	x	x			
UpdateMetadataField		x	x	x			
UpdateProtection		x	x	x			
UpdateProtectionField		x	x	x			
UpdateLabel			x				

UpdateData					x	x	x
isSetField	x	x	x	x			
ListMetadata		x	x	x			
ListMetadataField		x	x	x			
ListProtection		x	x	x			
ListProtectionField		x	x	x			
ListLabel			x				
SearchMetadata		x	x	x			
SearchMetadataField		x	x	x			
SearchProtection		x	x	x			
SearchProtectionField		x	x	x			
SearchLabel			x				
Authorize		x	x	x			
Verify		x	x	x	x	x	x
ConvertTo		x	x	x			
ConvertFrom		x	x	x			
StreamData		x	x	x	x		
Beacon		x					

10.1.1 REST API

The following tables list every operation available. For each of it, the corresponding row provides name, URL for a REST-based service and brief description. Access operations are currently mapped to the GET HTTP method [ref.] and modification operations (add and update) are mapped to the POST HTTP method. The use of the PUT HTTP method for modification operations needs further discussion.

Table 7.6 – Operations for the File level

Operation name	URL (for REST-based API's)	Description
getHeader	GET /header	Returns the content of the file header
getHeaderField	GET /header/hfield={id}	Returns a field identified by field_name from the file header
getDatasetGroup	GET /datasetgroup/{id}	Returns the entire dataset group with id {id}.

addHeaderField	POST / hfield={field_name}	Adds a field identigied by field_name to the field header (field has to be marked as optional in Part 1, or can be extended as the compatible brands).
updateHeader	POST / header	Updates the content of the file header
updateDatasetsGroupHeader Field	POST / hfield={field_name}	Updates a field identified by field_name in the file header.
isSetField	GET / hfield={field_name}	Returns true if field is set, false otherwise

Table 7.7 – Operations for the Datasets Group level

Operation name	URL (for REST-based API's)	Description
getDatasetsGroupHeader	GET /datasetgroup/{id}/header	Returns the content of the datasets group header for the dataset group identified by {id}.
updateDatasetsGroupHeader	POST /datasetgroup/{id}/header	Updates the content of the datasets group header for the dataset group identified by {id}.
getDatasetsGroupHeader Field	GET /datasetgroup/{id}/hfield={field_name}	Returns a field identified by field_name from the datasets group header.
updateDatasetsGroupHeader Field	POST /datasetgroup/{id}/hfield={field_name}	Updates a field identified by field_name in the datasets group header.
getDatasetsGroup Metadata	GET /datasetgroup/{id}/metadata	Returns the content of the datasets group metadata for the dataset group identified by {id}.
getDataset	GET /datasetgroup/{id}/dataset/{id2}	Returns the entire dataset with id {id2} contained within the dataset group with id {id}.
addHeaderField	POST /datasetgroup/{id}/hfield={field_name}	Adds a field identigied by field_name to the datasetgroup header (field has to be marked as optional in Part 1).
addDatasetsGroupMetadata	POST /datasetgroup/{id}/metadata	Adds the content of the datasets group metadata for the dataset group identified by {id}.
updateDatasetsGroupMetadata	POST /datasetgroup/{id}/metadata	Updates the content of the datasets group metadata for the dataset group identified by {id}.
getDatasetsGroup MetadataField	GET /datasetgroup/{id}/mfield={field_name}	Returns a field identified by field_name from the datasets group metadata.
addDatasetsGroupMetadata Field	POST /datasetgroup/{id}/mfield={field_name}	Adds a field identified by field_name in the datasets group metadata.
updateDatasetsGroupMetadataField	POST /datasetgroup/{id}/mfield={field_name}	Updates a field identified by field_name in the datasets group metadata.
getDatasetsGroupProtection	GET /datasetgroup/{id}/protection	Returns the content of the datasets group protection for the dataset group identified by {id}.
addDatasetsGroupProtection	POST /datasetgroup/{id}/protection	Adds the content of the datasets group protection for the dataset group identified by {id}.
updateDatasetsGroupProtection	POST /datasetgroup/{id}/protection	Updates the content of the datasets group protection for the dataset group identified by {id}.
getDatasetsGroupProtection Field	GET /datasetgroup/{id}/pfield={field_name}	Returns a field identified by field_name from the datasets group protection.
addDatasetsGroupProtection Field	POST /datasetgroup/{id}/pfield={field_name}	Adds a field identified by field_name in the datasets group protection.

updateDatasetsGroupProtectionField	POST /datasetgroup/{id}/pfield={field_name}	Updates a field identified by field_name in the datasets group protection.
isSetField	GET /datasetgroup/{id}/hfield={field_name}	Returns true if field is set, false otherwise
listMedatadata	GET /datasetgroup/{id}/metadata	Returns a list of active metadata fields
listMetadataField	GET /datasetgroup/{id}/mfield={field_name}	Returns a list of active values in the metadata field
listProtection	GET /datasetgroup/{id}/protection	Returns a list of active protection fields
listProtectionField	GET /datasetgroup/{id}/pfield={field_name}	Returns a list of active values in the protection field
searchMetadata	GET /datasetgroup/{id}/search_metadata?{search_criteria}	Returns a list of metadata fields matching the provided {search_criteria}
searchMetadataField	GET /datasetgroup/{id}/search_metadataField?{search_criteria}	Returns a list of values in a metadata field matching the provided {search_criteria}
searchProtection	GET /datasetgroup/id/search_protection?{search_criteria}	Returns a list of protection fields matching the provided {search_criteria}
searchProtectionField	GET /datasetgroup/id/search_protectionField?{search_criteria}	Returns a list of protection values matching the provided {search_criteria}
Authorize	POST /datasetgroup/{id}	Posts a XACML request, receives a XACML resolution.
Verify	GET /datasetgroup/{id}	Returns the list of signatures which could be verified.
convertTo	GET /datsetgroup/{id}?formatId={fid}	Returns as many files as necessary to convert the datsetgroup's content to the selected format.
convertFrom	POST /datasetgroup/{id}	Creates a datasetgroup, or replaces the datasetgroup with the content of the POSTED file.
streamDatasetsGroup	GET /datasetgroup/{id}/stream	Streams the data contained in the datasets group identified by id.
Beacon	GET /datasetgroup/{id}/beacon?positionId={pId}&statId={sId}	Returns the requested statistical information for the requested position

Table 7.8 – Operations for the Datasets Group level

Operation name	URL (for REST-based API's)	Description
getDatasetHeader	GET /datasetgroup/{id}/dataset/{did}/header	Returns the content of the datasets header for the dataset identified by {did}.
getDatasetHeaderField	GET /datasetgroup/{id}/dataset/{did}/hfield={field_name}	Returns the content of the dataset's header field with name {field_name}
getDatasetMetadata	GET /datasetgroup/{id}/dataset/{did}/metadata	Returns the content of the datasets metadata.
getDataset MetadataField	GET /datasetgroup/{id}/dataset/{did}/mfield={field_name}	Returns the field identified by field_name from the dataset metadata.
getDatasetProtection	GET /datasetgroup/{id}/dataset/{did}/protection	Returns the content of the datasets protection for the dataset identified by {did}.

getDatasetsProtectionField	GET /datasetgroup/{id}/dataset/{did}/pfield={field_name}	Returns a field identified by field_name from the datasets protection.
getLabel	GET /datasetgroup/{id}/dataset/{did}/label={lid}	Returns the definition of the label with id {lid}
addHeaderField	POST /datasetgroup/{id}/dataset/{did}/hfield={field_name}	Adds a field identified by field_name to the dataset header (field has to be marked as optional in Part 1).
addDatasetMetadata	POST /datasetgroup/{id}/dataset/{did}/metadata	Adds the content of the datasets metadata for the dataset identified by {did}.
addDatasetsMetadataField	POST /datasetgroup/{id}/dataset/{did}/mfield={field_name}	Adds a field identified by field_name in the datasets metadata.
addDatasetsProtection	POST /datasetgroup/{id}/dataset/{did}/protection	Adds the content of the datasets protection for the dataset identified by {did}.
addDatasetsProtectionField	POST /datasetgroup/{id}/dataset/{did}/pfield={field_name}	Adds a field identified by field_name in the datasets protection.
addLabel	POST /datasetgroup/{id}/dataset/{did}/label	Adds a new label description to the existing dataset's list (possibly empty) of labels.
updateDatasetsHeader	POST /datasetgroup/{id}/dataset/{did}/header	Updates the content of the datasets header for the dataset identified by {did}.
updateDatasetsHeader Field	POST /datasetgroup/{id}/dataset/{did}/hfield={field_name}	Updates a field identified by field_name in the datasets header.
updateDatasetMetadata	POST /datasetgroup/{id}/dataset/{did}/metadata	Updates the content of the datasets group metadata for the dataset identified by {did}.
updateDatasetMetadataField	POST /datasetgroup/{id}/dataset/{did}/mfield={field_name}	Updates a field identified by field_name in the datasets metadata.
updateDatasetProtection	POST /datasetgroup/{id}/dataset/{did}/protection	Updates the content of the dataset protection for the dataset identified by {did}.
updateDatasetProtectionFiel d	POST /datasetgroup/{id}/dataset/{did}/pfield={field_name}	Updates a field identified by field_name in the datasets protection.
updateLabel	POST /datasetgroup/{id}/dataset/{did}/lid={lid}	Updates the definition of the label identified by id {lid}.
isSetField	GET /datasetgroup/{id}/dataset/{did}/hfield={field_name}	Returns true if field is set, false otherwise
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
listProtection	GET /datasetgroup/{id}/dataset/{did}/protection	Returns a list of active protection fields
listProtectionField	GET /datasetgroup/{id}/dataset/{did}/pfield={field_name}	Returns a list of active values in the protection field

listLabel	GET /datasetgroup/{id}/dataset/{did}/label	Returns a list of active labels
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}
searchProtection	GET /datasetgroup/id/ dataset/{did}/search_protection?{search_criteria}	Returns a list of protection fields matching the provided {search_criteria}
searchProtectionField	GET /datasetgroup/id/ dataset/{did}/search_protectionField?{search_criteria}	Returns a list of protection values matching the provided {search_criteria}
searchLabel	GET /datasetgroup/id/ dataset/{did}/search_label{search_criteria}	Returns a list of labels matching the provided {search_criteria}
Authorize	POST /datasetgroup/{id}/dataset/{did}	Posts a XACML request, receives a XACML resolution.
Verify	GET /datasetgroup/{id}/dataset/{did}	Returns the list of signatures which could be verified.
convertTo	GET /datsetgroup/{id}/dataset/{did}?formatId={fid}	Returns as many files as necessary to convert the datset's content to the selected format.
convertFrom	POST /datasetgroup/{id}/dataset/{did}	Creates a dataset, or replaces the dataset with the content of the POSTED file.
streamDatasets	GET /datasetgroup/{id}/dataset/{did}/stream	Streams the data contained in the datasets group identified by id.

Table 7.9 – Operations for the Descriptor stream level

Operation name	URL (for REST-based API's)	Description
getStreamHeader	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/header	Returns the content of the stream's header for the dataset identified by {did}.
getStreamHeaderField	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/hfield={field_name}	Returns the content of the stream's header field with name {field_name}
getStreamMetadata	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/metadata	Returns the content of the stream's metadata.
getStream MetadataField	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/mfield={field_name}	Returns the field identified by field_name from the stream's metadata.
getStreamProtection	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/protection	Returns the content of the stream protection for the stream identified by {sid}.
getStreamGroupProtectionField	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/pfield={field_name}	Returns a field identified by field_name from the streams protection.
addHeaderField	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/hfield={field_name}	Adds a field identified by field_name to the stream header (field has to be marked as optional in Part 1).

addStreamMetadata	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/metadata	Adds the content of the stream's metadata for the dataset identified by {sid}.
addstreamMetadataField	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/mfield={field_name}	Adds a field identified by field_name in the streams metadata.
addStreamProtection	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/protection	Adds the content of the stream protection for the stream identified by {sid}.
addStreamProtectionField	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/pfield={field_name}	Adds a field identified by field_name in the stream's protection.
updateStreamHeader	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/header	Updates the content of the stream header for the dataset identified by {sid}.
updateStreamHeader Field	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/hfield={field_name}	Updates a field identified by field_name in the streams header.
updateStreamMetadata	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/metadata	Updates the content of the stream's metadata for the dataset identified by {sid}.
updateStreamMetadataField	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/mfield={field_name}	Updates a field identified by field_name in the stream's metadata.
updateStreamProtection	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/protection	Updates the content of the stream protection for the stream identified by {sid}.
updateStreamProtectionFiel d	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/pfield={field_name}	Updates a field identified by field_name in the stream's protection.
isSetField	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/hfield={field_name}	Returns true if field is set, false otherwise
listMedatadata	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/metadata	Returns a list of active metadata fields
listMetadataField	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/ mfield={field_name}	Returns a list of active values in the metadata field
listProtection	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/protection	Returns a list of active protection fields
listProtectionField	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/pfield={field_name}	Returns a list of active values in the protection field
searchMetadata	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/search_metadata?{search_criteria}	Returns a list of metadata fields matching the provided {search_criteria}
searchMetadataField	GET /datasetgroup/{id}/ dataset/{did}/ stream/{sid}/search_metadataField?{search_criteria}	Returns a list of values in a metadata field matching the provided {search_criteria}
searchProtection	GET /datasetgroup/id/ dataset/{did}/ stream/{sid}/search_protection?{search_ criteria}	Returns a list of protection fields matching the provided {search_criteria}
searchProtectionField	GET /datasetgroup/id/ dataset/{did}/ stream/{sid}/search_protectionField?{search_criteria}	Returns a list of protection values matching the provided {search_criteria}

Authorize	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}	Posts a XACML request, receives a XACML resolution.
Verify	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}	Returns the list of signatures which could be verified.
convertTo	GET /datsetgroup/{id}/dataset/{did}/strean/{sid}?formatId={fid}	Returns as many files as necessary to convert the stream's content to the selected format. The selected format has to be able to store only the information contained in that stream.
convertFrom	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}	Creates a stream, or replaces the stream with the content of the POSTED file. The POSTED file cannot contain data not compatible with the stream type
streamStream	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/stream	Streams the data contained in the stream identified by sid.

Table 7.10 – Operations for the Block level

Operation name	URL (for REST-based API's)	Description
getData	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/block/{bid}	Returns the content of block
addData	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/block/{bid}	Append the provided content to the selected block
updateData	POST /datasetgroup/{id}/dataset/{did}/stream/{sid}/block/{bid}/header	Updates the content of the block.
Verify	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}	Returns the list of signatures which could be verified.
streamBlock	GET /datasetgroup/{id}/dataset/{did}/stream/{sid}/block/{bid}/stream	Streams the data contained in the block identified by bid.

Table 7.11 – Operations for the Transport block level

Operation name	URL (for REST-based API's)	Description
getData	GET /	Returns the content of block
addData	POST /	Append the provided content to the selected block
updateData	POST /	Updates the content of the block.
Verify	GET /verify	Returns the list of signatures which could be verified.

Table 7.12 – Operations on packets

Operation name	URL (for REST-based API's)	Description
getData	GET /	Returns the content of packet
addData	POST /add	Append the provided content to the packet

updateData	POST /	Updates the content of packet
Verify	GET /verify	Returns the list of signatures which could be verified.

10.1.2 C-like API

[Note: To be discussed if we need this together with the REST API]

11 References

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

[2] EGA, "EGA metadata's schema," [Online]. Available: ftp://ftp.sra.ebi.ac.uk/meta/xsd/sra_1_5.

[3] NCBI, "Metadata's schema," [Online]. Available: https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=xml_schemas.

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

12 Annexes

Annex I – Protection boxes XML Schemas

This annex describes the XML schemas corresponding to the protection elements associated to Dataset group, dataset and descriptor stream.

I.1 Dataset group protection box 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>
```

I.2 Dataset protection box 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>
```

I.3 Descriptor stream protection box XML schema

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema       attributeFormDefault="unqualified"       elementFormDefault="qualified"
targetNamespace="http://www.w3.org/2001/04/xmlenc#"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element              name="EncryptedData"              type="xe:EncryptedDataType"
xmlns:xe="http://www.w3.org/2001/04/xmlenc#"/>
  <xs:complexType name="EncryptionMethodType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:anyURI" name="Algorithm" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="EncryptedDataType">
    <xs:sequence>
      <xs:element         type="xe:EncryptionMethodType"         name="EncryptionMethod"
xmlns:xe="http://www.w3.org/2001/04/xmlenc#"/>
      <xs:element ref="xd:KeyInfo" xmlns:xd="http://www.w3.org/2000/09/xmldsig#"/>
      <xs:element               type="xe:CipherDataType"               name="CipherData"
xmlns:xe="http://www.w3.org/2001/04/xmlenc#"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CipherReferenceType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="URI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="CipherDataType">
    <xs:sequence>
      <xs:element          type="xe:CipherReferenceType"          name="CipherReference"
xmlns:xe="http://www.w3.org/2001/04/xmlenc#"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Annex II – Examples of XACML rules

This annex describes some examples of XACML policies that may be included into a MPEG-G genomic file.

It is a XACML policy containing several rules, exemplifying two different approaches. On the one hand, the case where by default we deny access to, and the rules indicate the exceptions to this. These rules are the ones defined for roles “physician” and “researcher” (the first role is allowed to access the whole dataset, the later only the chromosome 2). On the other hand, we have the case where the default is to grant access, while the rules provide the exceptions. For an example of this, refer to the rules applying to the role “doctor”, where the access to chromosome 2 is denied.

```
<Policy
    xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
                        http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
    PolicyId="urn:genomeaccescontrol:policyid:2"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable"
    Version="1.0">
    <Description> Policy rules sample</Description>
    <PolicyDefaults>
        <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
    </PolicyDefaults>
    <Target/>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleSAM" Effect="Permit">
        <Description> A physician may view the genomic information file
            for which he or she is the designated primary care
            physician, provided an email is sent to the patient</Description>
        <Target>
            <AnyOf>
                <AllOf>
                    <!-- Which kind of user: physician -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                           physician
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:role" AttributeId="role"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                    <!-- Which resource -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            toy.sam
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                    <!-- Which action  -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            VIEW
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                </AllOf>
            </AnyOf>
        </Target>

        <Condition>
```

```
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
                    <ApplyFunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:count" AttributeId="countView"
                            DataType="http://www.w3.org/2001/XMLSchema#integer"/>
                    </Apply>
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
                        4
                    </AttributeValue>
                </Apply>
            </Apply>
        </Condition>
    </Rule>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleSAMChromosome" Effect="Permit">
        <Description>A researcher may view chromosome 20 of a genomic information
            file if he is the responsible of the study,
            provided an email is sent to the data sharer </Description>
        <Target>
            <AnyOf>
                <AllOf>
                    <!-- Which kind of user: researcher -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            researcher
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:role" AttributeId="role"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                    <!-- Which resource -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            toy.sam#ref2
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                    <!-- Which action  -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            VIEWCHROMOSOME
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>
                </AllOf>
            </AnyOf>
        </Target>

        <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
                    <ApplyFunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:count" AttributeId="countView"
                            DataType="http://www.w3.org/2001/XMLSchema#integer"/>
                    </Apply>
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
                        4
                    </AttributeValue>
                </Apply>
            </Apply>
        </Condition>
    </Rule>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleSAMChromosomeDeny" Effect="Deny">
        <Description>A doctor cannot view chromosome 2 </Description>
        <Target>
            <AnyOf>
                <AllOf>
                    <!-- Which kind of user: researcher -->
```

```
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            doctor
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:role" AttributeId="role"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                    <!-- Which resource -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            file.sam#ref2
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                    <!-- Which action  -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            VIEWCHROMOSOME
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                </AllOf>
            </AnyOf>
        </Target>

        <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">

                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
                    <ApplyFunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:count" AttributeId="countView"
                            DataType="http://www.w3.org/2001/XMLSchema#integer"/>
                    </Apply>
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
                        4
                    </AttributeValue>
                </Apply>
            </Apply>
        </Condition>
    </Rule>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleSAMChromosomeALL" Effect="Permit">
        <Description>A doctor may view all genomic information,
            provided an email is sent to the data sharer </Description>
        <Target>
            <AnyOf>
                <AllOf>
                    <!-- Which kind of user: doctor -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            doctor
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:role" AttributeId="role"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>

                    <!-- Which resource -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            file.sam*
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>
```

```
                    <!-- Which action  -->
                    <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                            VIEWCHROMOSOME
                        </AttributeValue>
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Match>
                </AllOf>
            </AnyOf>
        </Target>

        <Condition>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
                    <ApplyFunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
                        <AttributeDesignator MustBePresent="false"
                            Category="urn:oasis:names:tc:xacml:3.0:count" AttributeId="countView"
                            DataType="http://www.w3.org/2001/XMLSchema#integer"/>
                    </Apply>
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
                        4
                    </AttributeValue>
                </Apply>
            </Apply>
        </Condition>
    </Rule>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:genomeaccescontrol:FinalRule" Effect="Deny"/>
    <ObligationExpressions>
        <ObligationExpression ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
            FulfillOn="Permit">
            <AttributeAssignmentExpression
                AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
                <AttributeSelector
                    MustBePresent="true"
                    Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                    Path="patient-email"
                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </AttributeAssignmentExpression>
            <AttributeAssignmentExpression
                AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
                    >Your genomic information has been accessed by:</AttributeValue>
            </AttributeAssignmentExpression>
            <AttributeAssignmentExpression
                AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
                <AttributeDesignator
                    MustBePresent="false"
                    Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </AttributeAssignmentExpression>
        </ObligationExpression>
    </ObligationExpressions>
</Policy>
```

Annex III – Examples of metadata

Example of a possible datasets group metadata.

```
<datasetGroupMetadata>
  <title>Human male and female with glaucoma</title>
  <type>Whole Genome Sequencing</type>
  <Samples>
    <Sample>
      <TaxonId>9096</TaxonId>
      <Title>Human female with glaucoma</Title>
    </Sample>
    <Sample>
      <TaxonId>9096</TaxonId>
      <Title>Human male with glaucoma</Title>
    </Sample>
  </Samples>
</ datasetGroupMetadata>
```

Example of a possible dataset metadata element, belonging to the previously described dataset group. The sample element specifies for which of the two samples from the datasets group, the current dataset holds information.

```
<datasetMetadata>
  <Samples>
    <Sample>
      <TaxonId>9096</TaxonId>
      <Title>Human female with glaucoma</Title>
    </Sample>
  </Samples>
</ datasetMetadata>
```