**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 N17080**

**July 2017, Torino, IT**

|  |  |
| --- | --- |
| **Source** | **ISO/IEC JTC 1/SC 29/WG 11** |
| **Status** | **Approved** |
| **Title** | **Genomic Information Representation APIs** |

**Contents**

[1 Introduction 4](#_Toc489207722)

[2 Scope 4](#_Toc489207723)

[3 Normative references 4](#_Toc489207724)

[4 Terms and definitions 4](#_Toc489207725)

[5 Character encoding 5](#_Toc489207726)

[6 SAM interoperability 5](#_Toc489207727)

[6.1 SAM Header 5](#_Toc489207728)

[6.1.1 HD field 5](#_Toc489207729)

[6.1.2 SQ section 6](#_Toc489207730)

[6.1.3 Read Group (RG) 7](#_Toc489207731)

[6.1.4 Program Records (PG) 7](#_Toc489207732)

[6.1.5 Comments (CO) 8](#_Toc489207733)

[6.2 Auxiliary fields mapping 8](#_Toc489207734)

[6.2.1 SAM auxiliary fields 8](#_Toc489207735)

[6.2.2 User defined fields 12](#_Toc489207736)

[6.3 Transcoding to/from SAM 12](#_Toc489207737)

[6.3.1 SAM Flags 12](#_Toc489207738)

[6.3.2 Unmapped mate with PNEXT value 13](#_Toc489207739)

[6.3.3 Duplicate records 13](#_Toc489207740)

[6.3.4 SAM headers errors 13](#_Toc489207741)

[6.3.5 Mapping position error 14](#_Toc489207742)

[7 Supported FASTA format 14](#_Toc489207743)

[8 Protection 14](#_Toc489207744)

[8.1 Encryption 14](#_Toc489207745)

[8.1.1 *xenc:EncryptedData* 14](#_Toc489207746)

[8.1.2 Encryption-blocks 15](#_Toc489207747)

[8.2 Privacy Rules 16](#_Toc489207748)

[8.3 Digital Signature 16](#_Toc489207749)

[8.3.1 General case 16](#_Toc489207750)

[8.3.2 Authenticity of the dataset group protection box 17](#_Toc489207751)

[9 APIs 17](#_Toc489207752)

[9.1 API definition 17](#_Toc489207753)

[9.1.1 REST API 21](#_Toc489207754)

[9.1.2 C-like API 27](#_Toc489207755)

[10 Bibliography 27](#_Toc489207756)

[11 Annexes 28](#_Toc489207757)

[Annex I – Protection boxes XML Schemas 28](#_Toc489207758)

[I.1 Dataset group protection box XML schema 28](#_Toc489207759)

[I.2 Dataset protection box XML schema 28](#_Toc489207760)

[I.3 Descriptor stream protection box XML schema 29](#_Toc489207761)

[Annex II – Examples of XACML rules 30](#_Toc489207762)

# 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,
* controlled access to genomic information coded in compliance with Part 1 (N17075) and Part 2 (N117076) can be implemented and enforced,
* interfaces to access genomic information coded in compliance with Part 1 (N17075) and Part 2 (N17076).

# Scope

# Normative references

# 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 |
|  |  |
|  |  |

# Character encoding

This specification utilizes UTF8 character encoding.

# 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[];

}

## 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.

### 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 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 |
| 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 |
|  |  |  |  |

### 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 |

### 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 |

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

### Comments (CO)

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

## 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 |

### 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, BAQi = Qi - (BQi - 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 associated to the read according to the Part 2 of this standard.* | R2 |
| **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/=/XCIGAR 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 |

### 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’.

## 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.

### 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 |

#### 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.

### 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.

### 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 CCCFFFFFHHGHHJJJJJJJJJIJJJJJJJJJJJJJJJJJJJJIJHHHHEHFFFFFFEDEDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD> 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 CCCFFFFFHHGHHJJJJJJJJJIJJJJJJJJJJJJJJJJJJJJIJHHHHEHFFFFFFEDEDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD> 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:49C51HSQ1004:134: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 |

### 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. XXX].

|  |
| --- |
| 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:- |

### 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 to be added].

# 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 |
| … | … |
| … | … | … |

# 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 to be added] for the concrete structure.

## Encryption

### *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 to be added]. 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. 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.

### Encryption-blocks

The XML tag <encryption-blocks> indicates which blocks are encrypted and using which key. Refer to [ref to be added] 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 to be added], 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.

## Privacy Rules

The privacy rules tag has to be a valid policy element according to the XACML specification [XXX]: 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.]

## Digital Signature

### 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 to be added]. 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 to be added]. 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.

### 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.

# APIs

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

## 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.) [7]. |

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 |  |  |  |  |  |

### 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 to be added] 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 |
| updateDatasetsGroupHeaderField | 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}. |
| getDatasetsGroupHeaderField | GET /datasetgroup/{id}/hfield={field\_name} | Returns a field identified by field\_name from the datasets group header. |
| updateDatasetsGroupHeaderField | POST /datasetgroup/{id}/hfield={field\_name} | Updates a field identified by field\_name in the datasets group header. |
| getDatasetsGroupMetadata | 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}. |
| getDatasetsGroupMetadataField | GET /datasetgroup/{id}/mfield={field\_name} | Returns a field identified by field\_name from the datasets group metadata. |
| addDatasetsGroupMetadataField | 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}. |
| getDatasetsGroupProtectionField | GET /datasetgroup/{id}/pfield={field\_name} | Returns a field identified by field\_name from the datasets group protection. |
| addDatasetsGroupProtectionField | 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. |
| getDatasetMetadataField | 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}. |
| updateDatasetsHeaderField | 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}. |
| updateDatasetProtectionField | 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. |
| getStreamMetadataField | 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}. |
| updateStreamHeaderField | 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}. |
| updateStreamProtectionField | 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. |

### C-like API

***To be discussed if we need this together with the REST API***

# 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]  | S. D. Kahn, "On the Future of Genomic Data," *Science,* vol. 331, pp. 728-729, 2011.  |
| [3]  | Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron and G. E. Robinson, "Big Data: Astronomical or Genomical?," *PLOS Biology,* 2015.  |
| [4]  | ISO/IEC JTC 1/SC 29/WG 11 - ISO/TC 276/WG 5, "N16323/N97 - Requirements for Genomic Information Compression and Storage," Geneva, 2016.  |

# 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 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"> <Apply FunctionId="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"> <Apply FunctionId="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"> <Apply FunctionId="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"> <Apply FunctionId="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> |