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
MPEG2016/m39175
October 2016, Chengdu, China

Source: **DMAG-UPC, CNAG-CRG, BSC, Made of Genes, Dapcom, The Pirbright Institute, Stanford University, HITS gGmbH Heidelberg**

Status: **Proposal**

Title: **GENIFF (GENomic Information File Format), a proposal for a Secure Genomic Information Transport Layer (GITL) based on the ISO Base Media File Format**
Response to the Joint Call for Proposals for Genomic Information Compression and Storage

Authors: **Jaime Delgado, Silvia Llorente, Daniel Naro, Sara Rodríguez-Cubillas, Francesc Tarrés, Luis Torres (Distributed Multimedia Applications Group – Universitat Politècnica de Catalunya),**
Łukasz Roguski (Centro Nacional de Análisis Genómico – Centre de Regulació Genòmica (CNAG – CRG)),
Josep Lluís Gelpí, Dmitry Repchevsky, Romina Royo (Barcelona Supercomputing Center),
Leonor Frías, Oscar Flores (Made of Genes),
Jordi Portell, Francesc Julbe (DAPCOM Data Services),
Paolo Ribeca (The Pirbright Institute),
Mikel Hernaez, Idoia Ochoa, Tsachy Weissman (Stanford University),
Martin Golebiewski (HITS gGmbH, Heidelberg, Germany)

Index

1 Introduction 3
2 Structure of the proposal 3
3 Existing genomic data file formats 3
3.1 Introduction 3
3.2 FASTQ/SAM 3
3.3 Variant Call Format (VCF) 4
4 GENIFF, the GENomic Information File Format 4
4.1 Introduction 4
4.2 ISOBMFF 4
4.3 CARGO 5
4.4 GENIFF description 6
4.5 Proposal of box types and structure 7
4.6 Box description 8
5 Representation and storage of existing genomic data with GENIFF 10
5.1 Introduction 10
5.2 Non-compressed genomic information 10
5.2.1 Representation of FASTQ 10
5.2.2 Representation of SAM 11
5.2.3 Representation of VCF 13
5.3 Compressed genomic information 14
5.3.1 Compressed representation of MPEG FASTQ 14
5.3.2 Compressed representation of MPEG SAM 15
5.4 Selected technologies and their application in GENIFF 17
5.4.1 Application of ORCOM, CBC and QVZ 17
5.4.2 Application of FAPEC 20
6 Metadata associated to genomic information 21
6.1 Introduction 21
6.2 SAM Header 21
6.3 Inclusion of other kind of metadata and semantic-related information in the GITL 24
6.3.1 EBI metadata 24
6.3.2 MIAME metadata 26
6.3.3 Genomic Standards Consortium metadata 26
7 Security and Privacy for Genomic information 28
7.1 Privacy aspects 28
7.2 Use cases 28
7.3 Security and Privacy elements 31
8 Software tools 33
8.1 GENIFF generator 33
8.2 XACML authorization for genomic information access 34
9 Transport coverage matrix 36
10 Acknowledgements 37
11 References 37
Annex I – SAM Header XML schema 39
Annex II - Documentation of metadata to a genomic sequence according to the Minimum Information about any (x) Sequence (MIxS) checklist 42
Annex III – Examples of XACML rules 48
Annex IV - Authorizing access to genomic information based on XACML rules 51
Annex V – GENIFF generator 55
Annex VI – Integration of the two tools 57

1 Introduction

This document provides a response to the Joint Call for Proposals described in [1]. The organizations presenting this proposal are Distributed Multimedia Applications Group from the Universitat Politècnica de Catalunya (DMAG-UPC), Barcelona Supercomputing Center (BSC), Centro Nacional de Análisis Genómico - Centre de Regulació Genòmica (CNAG-CRG), the Spanish companies Made of Genes and DAPCOM Data Services, The Pirbright Institute, Stanford University and HITS gGmbH Heidelberg.

2 Structure of the proposal

The document is structured as follows. First, it gives an insight of the most common formats used to represent and store genomic information. Next, it proposes a format based on the ISO Base Media File Format [2] and adapted according to Compressed ARchiving for GenOmics (CARGO) [3] with specific new boxes for genomic information in order to define a Genomic Information Transport Layer (GITL) to support the Transport requirements defined in [4].

Next, it describes how this proposal fits with the current genomics file formats, like FASTQ or SAM/BAM, among others, showing how they can be modeled using the proposed format. It also describes the relationship with the state of the art compression algorithms for genomics information, especially with the one described in the ORCOM [5] - based proposal [6] for MPEG. Also, the relationship with the FAPEC compression [7] proposed for MPEG in [8] is presented.

Additionally, it describes how the metadata associated to genomic information can be included in the GITL and the different types of metadata considered, like the compression of quality scores described in QVZ [6][9].

Moreover, the document defines privacy aspects to be considered when describing, storing and transmitting genomic information. Several use cases are presented to justify the need of including privacy aspects when representing genomic information. As privacy and security are closely related, a proposal for inclusion of security and privacy aspects in the defined GITL is also described.

Finally, we justify why this proposal covers the Transport requirements defined in [4].

3 Existing genomic data file formats

3.1 Introduction

This section lists some of the existing genomic data file formats that we considered for inclusion in the genomic information file format proposed in this document.

The existing formats are used to show how current genomic information can be structured and included in our proposal for GITL. Nevertheless, the final aim of this proposal is to mainly include the binary files resulting from the MPEG compression algorithm selected.

3.2 FASTQ/SAM

FASTQ [10] and SAM [11] [12] formats were selected for compression by the MPEG working group. Their format and expected usage within MPEG are completely described in documents [13], [14] and [15].

3.3 Variant Call Format (VCF)

Although Variant Call Format (VCF) [16] is not currently included in the compression requirements, it is also considered in this document to show how a complete study of an individual can be represented inside the proposed file format. VCF specifies the format of a text file for storing genomic sequence variations.

4 GENIFF, the GENomic Information File Format

4.1 Introduction

In order to define a file format for the storage and transmission of genomic information, we follow the philosophy behind ISO Base Media File Format (ISOBMFF) [2] combined with concepts coming from Compressed ARchiving for GenOmics (CARGO) [3]. The given name for this file format is GENomic Information File Format (GENIFF).

In the rest of this section we briefly introduce to the ISOBMFF file format. Next, we describe CARGO and finally we define the structure for GENIFF.

4.2 ISOBMFF

In order to define a file format for the storage and transmission of genomic information, we follow the concepts behind ISO Base Media File Format (ISOBMFF) [2].

Figure 1 (extracted from [2]) shows the general structure of an ISO file, where the multimedia tracks are contained in a `moov` (movie) box and the media data is stored in the `mdat` (media data) box. The multimedia information is structured in such a way that allows different possibilities for file creation.

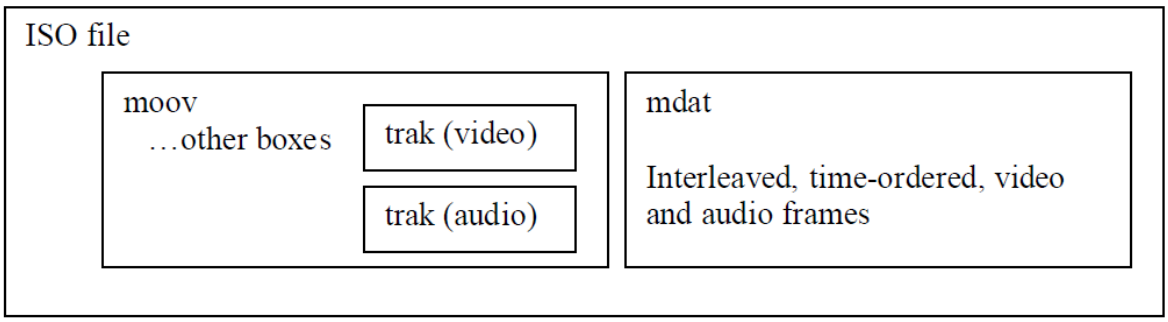


Figure 1 — Simple interchange file

It is also possible, as shown in Figure 2 (extracted from [2]), that the files are not completely self-contained but split into different files. This facilitates supporting the content creation process, where the media files can be edited, deleted, etc. Other file organizations are also possible, to, for example, facilitate streaming of media or local presentation.

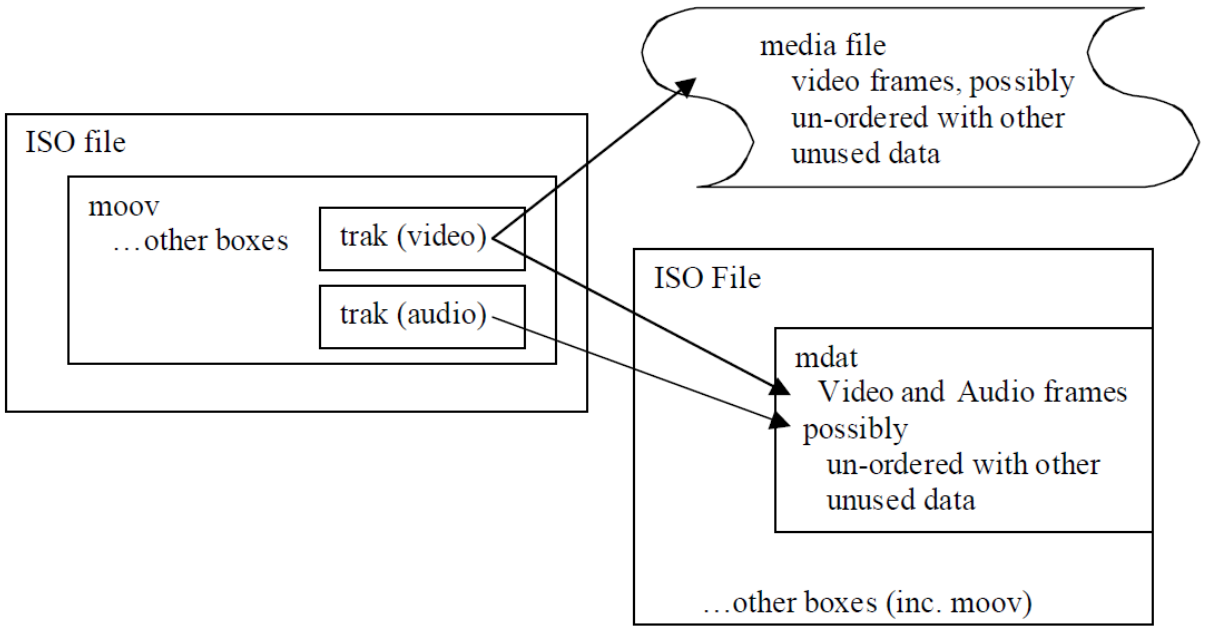


Figure 2 — Content Creation File

Therefore, a file format for genomic information should define specific boxes for supporting this type of information, while keeping the base ideas of the structure of the ISO file for multimedia.

It is worth noting that genomic information does not have a temporal order, as assumed for the content of the ISO file, but a position order, especially when the genomic information is aligned. Nevertheless, streaming of genomic information is an important requirement due to the big amount of information that is currently being generated and analyzed.

4.3 CARGO

CARGO [3] is a high-level framework to automatically generate software systems for the compressed storage of arbitrary types of large genomic data collections. It uses its own container file format to store the data. In CARGO each genomic record type can be precisely-defined in terms of own domain-specific meta-language. It allows the creation of rich data types and most of the commonly used genomic formats can be easily defined. Figure 3 gives an example of definition of SAM records using CARGO metalanguage.

Col	Field	Type	Brief description
1	QNAME	String	Query template NAME
2	FLAG	Int	bitwise FLAG
3	RNAME	String	Reference sequence NAME
4	POS	Int	1-based leftmost mapping POSition
5	MAPQ	Int	MAPping Quality
6	CIGAR	String	CIGAR string
7	RNEXT	String	Ref. name of the mate/next read
8	PNEXT	Int	Position of the mate/next read
9	TLEN	Int	observed Template LENgth
10	SEQ	String	segment SEQuence
11	QUAL	String	ASCII of Phred-scaled base QUALity+33
12	OPT	String	A collection of tab-separated OPTional fields

```
SamRecord = {
  qname = string;
  flag = uint^16;
  rname = string;
  pos = uint^32;
  mapq = uint^8;
  cigar = string;
  next = string;
  pnext = int^32;
  tlen = int^32;
  seq = string;
  qua = string;
  opt = string;
} :
```

Figure 3 – (left) SAM record fields and (right) its corresponding record definition in CARGO

From the supplied record specification CARGO is able to internally decompose the fields of a given record into a number of different *streams*. These streams store the information obtained from the same fields (and of the same type) coming from different records. Each stream is compressed separately using a user-specified codec and it is stored as a collection of blocks inside the CARGO container. The dataset meta-information with record type definition is stored in *dataset area*. The file offsets and indexing information to directly access the streams and records data are also stored in the dataset area. Figure 4 illustrates this concept.

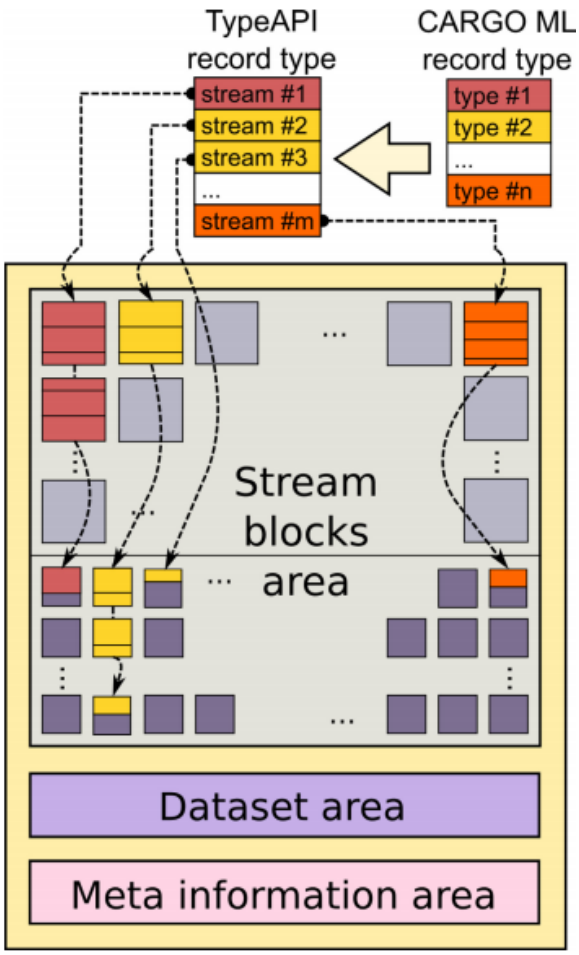


Figure 4 – A conceptual representation of low-level storage in CARGO

Additionally, inside one CARGO container, multiple files and of different genomic types can be stored (as *datasets*). This allows for useful data compaction, storing possibly all the data related with a given study or experiment inside a single file.

4.4 GENIFF description

ISOBMFF and CARGO share a number of similar features and concepts, and hence, we propose a genomic information format based on both formats. By doing so, we allow modeling the hierarchical relationship of the object media data types and the composition of objects into more complex aggregated data types. Therefore, we first model the relevant concepts from CARGO into the ISOBMFF format and then we extend the joint format and concepts, using a bottom-up approach, with the objects hierarchy and relationship explicitly defined in Table 1.

As a starting point, we propose to model the low-level CARGO *stream* concept as ISOBMFF *box* – in the following we will refer to it as *stream box* – , called `gefb`. The stream box will be primarily used for representing and storing the information contained in a specified genomic record field. The information could be either contained in a GENIFF file or in an external file, which is then referenced from a GENIFF file. The `gefb` box represents the genomic record field, which contains the information related to the genomic data *stream*.

A complete set of `gefb` boxes storing the data of predefined genomic record type composes a *genomic dataset data box* (`gedb`). A dataset can represent the genomic file of type FASTA, FASTQ, SAM, VCF, etc.

`gedb` box is accompanied by optional *dataset meta-information box* (`gedh`) (representing optional genomic file-header and also including security and privacy information) and optional *dataset index box* (`gedi`). At this level, a complete study can be represented in a single GENIFF file as a collection of datasets (of possibly different types).

Multiple datasets can be grouped together into a *study* to possibly share the full results of a joint study and/or experiments. Therefore, multiple `gedb` dataset boxes can be included into *genomic study box* (`gesb`). Similarly, as `gedb` box, `gesb` box is accompanied by optional *meta-information box* (`gesh`), also including security and privacy information.

It is worth noting that genomic information does not have a temporal order, as assumed for the content of the ISOBMFF, but some (e.g. SAM or VCF) have position order. A single genomic record can be perceived as a frame in video context, but multiple records can share the same position (*time*). For these formats the ordering by position is preserved, for practical reasons. However, in special cases (e.g. focusing on maximum compression ratio), it may be skipped leading to unordered records (as in FASTQ). Nevertheless, streaming of genomic information is an important requirement due to the big amount of information generated and analyzed.

4.5 Proposal of box types and structure

In this section we present a proposal of box types needed for the representation of genomic information and its relationship. The boxes, defined together with a scheme of its structure, are presented in Table 1.

Table 1 contains the name of the boxes, their relationships (which elements are inside one another and in which order), if they can be repeated or not, and if they are required or not, together with a brief description (see Section 4.5 for a complete description).

Table 1 – Box types and structure

Name					Repeat	Require	Description
ftyp					NO	YES	File type and compatibility.
gesb					To be discussed	YES	Container for all the genomic data - the single study box.
	gesh				NO	NO	Genomic box header for the study box. It contains metadata, including privacy and security information. Note: Considering the use of the meta box defined in ISOBMFF to represent this information.
	gedb				YES	NO	Container for a single genomic dataset equivalent to file of type e.g. FASTA, FASTQ, SAM, VCF, etc.
		gedh			NO	NO	Genomic dataset header, overall information. It contains metadata, including privacy and security information. Note: Considering the use of the meta box defined in ISOBMFF to represent this information.
		gedi			NO	NO	Indexing information for fast dataset access.
		gefb			YES	NO	Genomic records' field box based on stream box.
			gefh		NO	NO	Genomic field header, overall information. It contains metadata, including privacy and security information. Note: Considering the use of the meta box defined in ISOBMFF to represent this information.
			gefi		NO	NO	Genomic information indexing information for direct access.
			gefd		NO	YES	Data information box including buffers, or file pointers on how data is stored internally.
				gdff	NO	NO	Data reference box, pointing directly to file offset(s) of external file.
				gdfd	NO	NO	Direct field data or information about data offsets inside the current file.

It is worth noting that the structure proposed for boxes is rather flexible as it should support different types of genomic information, similar to what ISO file does for multimedia. To show how genomic information files are supported, we give some examples in Section 5.

In particular, we give some details on how specific genomic file formats should be included into this structure to take benefit from the information (and meta-information) expressed by the boxes. Once the compression formats for MPEG are defined we will do the same exercise to describe how these files should be included into the transport format proposed.

4.6 Box description

This section describes the boxes defined in Table 1.

- `ftyp` box defines the type of file it contains as described in [2]. For the purpose of storing genetic information new values for `ftyp` should be defined with a major brand and a minor version.
- `gesb` contains all boxes for a specific study.
- `gesh` genomic header relevant for the entire study such as the tools used. Stores information on the security and privacy (usage rules and information on encryption), and other relevant information to identify the study.
- `gedb` container for a single genomic dataset.
- `gedb/gedh` required header information to correctly interpret the genomic dataset. Stores information on the security and privacy (usage rules and information on encryption), and other relevant information to identify the dataset.
- `gedb/gedi` indexing information specific for the genomic dataset.
- `gefb` container for one stream of data (e.g. uncompressed FASTA, FASTQ, SAM, BAM, VCF, …)
- `gefb/gefh` genomic header specific for the stream. Stores information on the security and privacy (usage rules and information on encryption), and other relevant information to identify the stream.
- `gefb/gefi` indexing information for the specific stream.
- `gefb/gefd` information on how the particular stream is stored.
- `gefb/gefd/gdff` data reference box which points to an external file location storing the stream information.
- `gefb/gefd/gdfd` actual data container of the file storing the relevant information.

Privacy rules defined using XACML should be included into an `xml` box contained in the `gesh`, `gedh`, and `gefh` boxes, depending on which level of information they apply to, the whole study (inside `gesb` box), a specific dataset (inside `gedb` box) or for a stream (inside `gefb` box).

5 Representation and storage of existing genomic data with GENIFF

5.1 Introduction

This section presents some examples of representation of existing genomic data formats inside GENIFF. It points out how other data formats could be included, especially to support compressed formats to be defined by MPEG for genomic information. We have divided the section into non-compressed and compressed genomic information.

Each genomic file format can be represented and stored in two different ways, depending on the specific situation. A genomic file (dataset) can be included in the GENIFF file as the current fixed format file (option 1) or it can be modeled by decomposing its fields' data into separate streams (option 2). We will use both approaches for flexibility and propose the possible compressible solutions in the following sub-sections.

5.2 Non-compressed genomic information

The following sub-sections sketch how non-compressed information could be included in the GENIFF format proposed. For each format, we provide an example of the values the boxes should contain.

5.2.1 Representation of FASTQ

5.2.1.1 A FASTQ file stored inside GENIFF (option 1)

In this example, one FASTQ file represents one dataset. To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`) or at the dataset level (`gedh`). In case there is some indexing information for the FASTQ information, it can be included in the `gefi` or `gedi` (pointing to `gefi`) boxes.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the FASTQ file.
		gedh			Study composed of a FASTQ file. Metadata, including privacy and security information. Even if not supported in FASTQ format, additional information can be included. For example, the information of the machine used for sequencing or experiment specific metadata.
			xacm		XACML rules and protection
		gedi			FASTQ index, if one is decided.
		gefb			Genomic record field box.
			gefh		Stream specific header.
			gefi		FASTQ index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.

5.2.1.2 A FASTQ format decomposed into separate streams stored inside GENIFF (option 2)

In this example, one FASTQ file represents one dataset decomposed into 3 streams: read identifiers, sequence and quality. To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`), at the dataset level (`gedh`) or at the separate stream level (`gefh`). The security measures depends on the required access granularity. In case there is some indexing information for the FASTQ information, it can be included in the `gefi` box (per each stream) or on the dataset level (`gedi`).

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the FASTQ file.
		gedh			Study composed of 3 unique streams composing a FASTQ record data. Metadata, including privacy and security information. Even if not supported in FASTQ format, additional information can be included. For example, the information of the machine used for sequencing or experiment specific metadata.
			xacm		XACML rules and protection
		gedi			Streams index, if one is decided.
		gefb			Genomic record field box #1 used to store read identifiers data.
			gefh		Stream specific header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #2 used to store sequence data.
			gefh		Stream specific header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #3 used to store quality data.
			gefh		Stream specific header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.

5.2.2 Representation of SAM

This section shows how to represent SAM files inside GENIFF. Regarding SAM header information, we have defined an XML schema to formalize its fields. This way, it can be stored in the `gefh` box, following the defined XML schema. The complete XML schema is included in this proposal to facilitate the formalization of SAM header fields to include them into GENIFF, but also for further automatic processing of the SAM header. It is explained in Section 6.2 and can be found in Annex I.

5.2.2.1 A SAM file stored inside GENIFF (option 1)

In this example, one SAM file represents one dataset. To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`) or at the dataset level (`gedh`). In case there is some indexing information for the SAM file, it can be included in the `gedi` box.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the SAM file.
		gedh			Study composed of a unique stream (SAM type). Metadata, including privacy and security information.
			samh		SAM Header modeled in XML.
			xacm		XACML rules and protection.
		gedi			SAM index, if one is decided.
		gefb			Genomic record field box.
			gefh		Stream specific header.
			gefi		Raw SAM index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream (SAM file without the header (records only), which will be re-generated from stored XML -- if needed)

5.2.2.2 A SAM format decomposed into streams stored inside GENIFF (option 2)

In this example, one SAM file represents one dataset decomposed into a number of separate streams (following the official SAM format specification [11][12]). To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`), at the dataset level (`gedh`) or at the separate stream level (`gefh`). In case there is some indexing information for the SAM file, it can be included in the `gefi` and `gedi` boxes.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the SAM file.
		gedh			Study composed of a unique stream - SAM file. Metadata, including privacy and security information.
			samh		SAM Header modeled in XML.
			xacm		XACML rules and protection.
		gedi			Global SAM index, if one is decided, linked to the stream-level indices.
		gefb			Genomic record field box #1 used to store alignment query template name data - SAM field QNAME.
			gefh		Stream specific header.

			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #2 used to store alignment flags data - SAM field FLAG.
			gefh		Stream specific header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
…					
		gefb			Genomic record field box #9 used to store sequence data – SAM field SEQ.
			gefh		Stream specific header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #10 used to store quality scores data - SAM field QUAL.
			gefh		Stream specific header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #11 used to store alignment optional fields data.
			gefh		Stream specific header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.

5.2.3 Representation of VCF

In this example, one VCF file represents one dataset. To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`), at the container level (`gedh`) or at the file (stream) level (`gefh`). In case there is some indexing information for the VCF file, it can be included in the `gefi` box.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the VCF file.
		gedh			Study composed of a unique stream (VCF type). Metadata, including privacy and security information.
		gefb			Genomic record field box.
			gefh		VCF specific header. Access rules and information about protection measures.
			gefi		VCF index, if one is decided.
			gefd		File pointer to how data is stored internally.

				gdfd	Byte array of the current stream

5.3 Compressed genomic information

The following sub-sections sketch how compressed information could be included in the GENIFF format proposed. For each format, we provide an example of the values the boxes should contain. We also include the case where information should be stored using the compression algorithms selected by the MPEG committee. When these algorithms are specified, these sections will be updated accordingly.

5.3.1 Compressed representation of MPEG FASTQ

The following sub-sections describe how FASTQ compressed information could be stored into GENIFF.

5.3.1.1 A FASTQ file stored inside GENIFF (option 1)

If FASTQ file data is to be compressed using a full FASTQ format compressor, then the resulting compressed binary data can be stored in the `gdfd` box. Such approach would apply to FASTQ being compressed by a method, that produces a single file as output. This can be done either by using a general purpose compressor like gzip (currently, the most common way) or format-specific one like DSRC [17], Quip [18] or FAPEC [7] (see Section 5.3.4).

In this example, one FASTQ file represents one dataset. To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`) or at the dataset level (`gedh`).

In this case, it is foreseen that indexing information will be provided by the compression mechanism for FASTQ defined by the MPEG Committee. So, this information has to be included in the `gefi` or `gedi` (pointing to `gefi`) boxes. The details on indexing information will be updated accordingly.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the FASTQ file.
		gedh			Study composed of a unique stream (FASTQ type – together with compression used). Metadata, including privacy and security information.
			xacm		XACML rules and protection.
		gedi			FASTQ index pointer to gefi box.
		gefb			Genomic record field box.
			gefh		Stream specific header.
			gefi		FASTQ index as defined by the MPEG Committee.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.

5.3.1.2 A FASTQ format decomposed into streams and stored inside GENIFF (option 2)

If a FASTQ file is to be compressed by decomposing the data into streams (e.g. identifiers, nucleotide sequences and quality scores), a different specialized codec can be applied per each stream. Thus, the compressed binary data of each stream can be stored in its corresponding `gdfd` box. For example, the sequence data could be compressed by ORCOM [5][6] and the corresponding output stored in `gefb` box #2. Similarly, the quality scores could be compressed for example with QVZ [6][9] and the corresponding output stored in `gefb` box #3.

As another example, the output of full FASTQ format compressors like SCALCE [19] or FQZcomp [20] is also represented as 3 separate files, one per information stream. Therefore, their output could be also possibly stored in GENIFF format using this FASTQ dataset representation method.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the FASTQ file.
		gedh			Study composed of 3 unique streams composing FASTQ record data Metadata, including privacy and security information. Access rules and information about protection measures (if needed)
			xacm		XACML rules and protection
		gedi			Streams index, if one is decided.
		gefb			Genomic record field box #1 used to store read identifiers data.
			gefh		Stream specific header
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #2 used to store compressed sequence data.
			gefh		Stream specific header
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #3 used to store compressed quality data.
			gefh		Stream specific header
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.

5.3.2 Compressed representation of MPEG SAM

The following sub-sections describe how SAM compressed information could be stored into GENIFF.

5.3.2.1 A SAM file stored inside GENIFF

In this example, we consider a compressed SAM file represented as a BAM file, which is the current standard for compressed SAM representation. Note that the compressed data is a single BAM file. To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`), at the container level (`gedh`) or at the file (stream) level (`gefh`). The `gefi`

box contains the indexing information, as defined by the MPEG Committee for the SAM compression. It currently uses the BAI index, to support BAM.

Concerning SAM header information, it is stored in the `gefh` box, following the XML schema defined for SAM header fields and explained in Section 6.2.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the BAM file.
		gedh			Study composed of a unique stream (BAM type – together with compression information). Metadata, including privacy and security information.
			samh		SAM Header modeled using XML.
			xacm		XACML rules and protection.
			enar		Bit array indicating which blocks of the data stored in gdfd are encrypted.
		gedi			Indexing information for unique stream is present at gesb.gedb[0].gefb[0]. This is used to emulate BAI behavior.
		gefb			Genomic record field box.
			gefh		Stream specific header.
			gefi		Currently, BAI index. Later on, the SAM index as defined by the MPEG Committee.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream

5.3.2.2 A SAM format decomposed into streams and stored inside GENIFF

In this example, one SAM file represents one dataset. To control access to the dataset, privacy rules and security information can be included at the study level (`gesh`), at the container level (gedh) or at the file (stream) level (`gefh`). The `gefi` box contains the indexing information per each stream.

If SAM file is to be stored in GENIFF format by decomposing its data streams, the compressed binary data of each stream can be stored in the `gdfd` box. In this way, a different codec can be used to compress each stream. For instance, if the read order is not required (i.e. to achieve maximum compression ratio), sequence data (corresponding SEQ box) can be compressed by ORCOM [5][6]. Alternatively, if the order is required, the sequence data could be compressed with CBC [21][22]. In either case, quality scores per alignments can be stored in corresponding QUAL box using QVZ [6][9].

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the BAM file.

		gedh			Study composed of a unique stream (BAM type). Metadata, including privacy and security information.
		gedi			Indexing information for unique streams is present at each sub-index box at gesb.gedb[0].gefb[i]. This is used to emulate BAI behavior.
		gefb			Genomic record field box #1 used to store alignment query template name data – SAM field QNAME
			gefh		Specific stream header.
			gefi		Stream index.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #2 used to store alignment flags data - SAM field FLAG
			gefh		Specific stream header.
			gefi		Stream index.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
...					
		gefb			Genomic record field box #9 used to store sequence data - SAM field SEQ
			gefh		Specific stream header.
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #10 used to store quality scores data - SAM field QUAL.
			gefh		Specific stream header.
			gefi		Stream index.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #11 used to store alignment optional fields data.
			gefh		Specific stream header.
			gefi		Stream index.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.

5.4 Selected technologies and their application in GENIFF

In previous sections, we have introduced several technologies for the representation of both uncompressed and compressed information for FASTQ and SAM formats. In the next subsections we describe in more detail how selected technologies for compression can be applied to the GENIFF format and the relationship with the compression proposals presented by the authors to the MPEG Committee [6][8].

5.4.1 Application of ORCOM, CBC and QVZ

ORCOM [5][6], CBC [21][22] and QVZ [6][16] algorithms are used to compress DNA sequences, aligned DNA sequences, and quality scores information, respectively. Since they are not strictly tied to any specific genomic file format, they can be used to compress the sequence and quality data both for compressed MPEG FASTQ and MPEG SAM formats. The compressed sequence and quality data can be stored inside the independent boxes (option 2). Alternatively, if a full

specific-format compressor is built based on the mentioned algorithms, the single resulting file can be stored inside GENIFF file (option 1).

5.4.1.1 Compressed representation of MPEG FASTQ

As briefly mentioned in section 5.3.1.2, the ORCOM algorithm can be used to compress DNA sequence information with the resulting binary data stored in `gdfd` box. Analogously, QVZ algorithm can be applied to compress quality scores and storing the resulting binary data into its separate `gdfd` box. The algorithms' parameters will be stored in their streams' corresponding header `gefh` boxes. Since ORCOM and QVZ algorithms are highly parallelizable, in their corresponding `gefi` boxes indexing information can be stored to allow for rapid access to specific compressed blocks of data.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the FASTQ file.
		gedh			Study composed of 3 unique streams composing FASTQ record data Metadata, including privacy and security information. Access rules and information about protection measures (if needed)
			xacm		XACML rules and protection
		gedi			Index to access compressed blocks
		gefb			Genomic record field box #1 used to store read identifiers data.
			gefh		Stream specific header
			gefi		Stream index, if one is decided.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream, a sequence of compressed blocks
		gefb			Genomic record field box #2 used to store sequence data compressed by ORCOM
			gefh		Stream specific header containing ORCOM compression configuration
			gefi		Index to ORCOM compressed blocks
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the ORCOM compressed blocks.
		gefb			Genomic record field box #3 used to store quality data compressed by QVZ
			gefh		Stream specific header containing QVZ configuration
			gefi		Index to QVZ compressed blocks
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the QVZ compressed blocks

5.4.1.2 Compressed representation of MPEG SAM

As briefly mentioned in section 5.3.2.2, QVZ algorithm can be used to compress quality scores information in the alignments (corresponding SAM field: QUAL). Additionally, ORCOM algorithm can be used to compress the sequence (SAM field: SEQ) of the alignments. The

compressed information will be stored in the corresponding `gdfd` boxes, whereas the configuration will be stored in the `gefh` boxes.

As a side note, applying ORCOM compression will result in a reshuffling of the alignments, which implies losing the BAI index compatibility. However, it can be applied to possibly achieve a very high compression ratio at the cost of further need of resorting the alignments and restoring the index.

Alternatively, the compression of the reads can be performed with the algorithm CBC. Note that in this case the order of the reads is preserved. Similarly to the option of using ORCOM, the compressed information will be stored in the `gdfd` box.

Field					Value
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the BAM file.
		gedh			Study composed of a unique stream (BAM type). Metadata, including privacy and security information.
		gedi			Indexing information for unique streams is present at each sub-index box at gesb.gedb[0].gefb[i]. This is used to emulate BAI behavior. However, if ORCOM algorithm is used to compress the sequence data, BAI behavior may not be possible.
		gefb			Genomic record field box #1 used to store alignment query template name data – SAM field QNAME
			gefh		Stream specific header
			gefi		Stream index.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
		gefb			Genomic record field box #2 used to store alignment flags data - SAM field FLAG
			gefh		Stream specific header
			gefi		Stream index.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.
...					
		gefb			Genomic record field box #9 used to store sequence data - SAM field SEQ. Compressed using ORCOM
			gefh		Stream specific header containing ORCOM configuration
			gefi		Index to ORCOM compressed blocks
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the ORCOM compressed blocks
		gefb			Genomic record field box #10 used to store quality scores data - SAM field QUAL. Compressed using QVZ.
			gefh		Stream specific header containing QVZ configuration
			gefi		Index to QVZ compressed blocks
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the QVZ compressed blocks.

		gefb			Genomic record field box #11 used to store alignment optional fields data.
			gefh		Stream specific header
			gefi		Stream index.
			gefd		File pointer to how data is stored internally.
				gdfd	Byte array of the current stream.

5.4.2 Application of FAPEC

In this example, one FAPEC-compressed FASTQ file [7] represents one dataset. Its binary contents (the compressed FASTQ file) are stored in the `gdfd` box.

It contains the several FAPEC compressed (and eventually encrypted) chunks in sequence, without any additional header (i.e., without the global FAPEC file or part headers). Details on the exact FAPEC configuration used for that dataset (exact algorithm, chunk size, error detection and correction options, cryptographic options, etc.) are stored in the `gefh` box. An XML header can be used, as in the BAM example.

As described in [8], each chunk can be decompressed independently, so here it really makes sense to define some kind of index in the `gefi` box. Its exact format is to be defined, but as an initial approach the index could indicate the header of the first sequence contained in each chunk, its position in the binary stream, and the number of sequences contained in that chunk.

To control access to the dataset, privacy rules and security information can also be included at the study level (`gesh`), at the container level (`gedh`) or at the file (stream) level (`gefh`).

Field					**Value**
ftyp					gnif (genomic information file format), version 1.0
gesb					Container for the study.
	gesh				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gedb				Container for the FASTQ-FAPEC file.
		gedh			Study composed of a unique stream (FAPEC-compressed FASTQ type). Metadata, including privacy and security information.
		gefb			Genomic record field box.
			gefh		FASTQ-FAPEC specific header. Access rules and information about protection measures. FAPEC configuration.
			gefi		Index to FAPEC compressed chunks.
			gefd		File pointer to how data is stored internally.
				gdfd	FAPEC byte array of the current stream (sequence of compressed chunks).

6 Metadata associated to genomic information

6.1 Introduction

The term metadata in this document refers to data describing context and origin of the sequence data, as well as information about the environmental context of the biological sources for the sequence information. In more practical terms, metadata comprises the information describing a sampling event and subsequent sequencing efforts leading to the genomic sequence information. Security and privacy information (described in section 7) could be handled together with these metadata.

This section presents how the metadata associated to genomic information can be included in GENIFF. It is worth noting that a complete set of metadata elements needs to be specified. In this current version, several kinds of metadata elements are presented, related to specific genomic formats, like SAM, or to specific use cases. The exact location of the metadata elements needs also to be formalized once the metadata elements are fully introduced.

In the next subsections, as an example, we use the SAM header as a starting point for the definition of metadata elements (see Section 6.2). Furthermore, we point out use cases that could help to identify required metadata elements (see Section 6.3).

6.2 SAM Header

This proposal includes the definition of an XML schema to formalize the fields contained in a SAM file. The objective of providing this schema is twofold. From the one side, it will facilitate the automatic processing of header information. From the other side, it will be easier to include the header fields inside the GENIFF corresponding box.

The complete XML schema can be found in Annex I, but here we present its main features. In Figure 5 we can find the first level elements of the sam_header, header_line, reference_sequence_dictionary, read_groups, programs and comment.

Only header_line is compulsory and all of them can appear at most once except comment, which may appear an undetermined number of times. Comment is of type string.

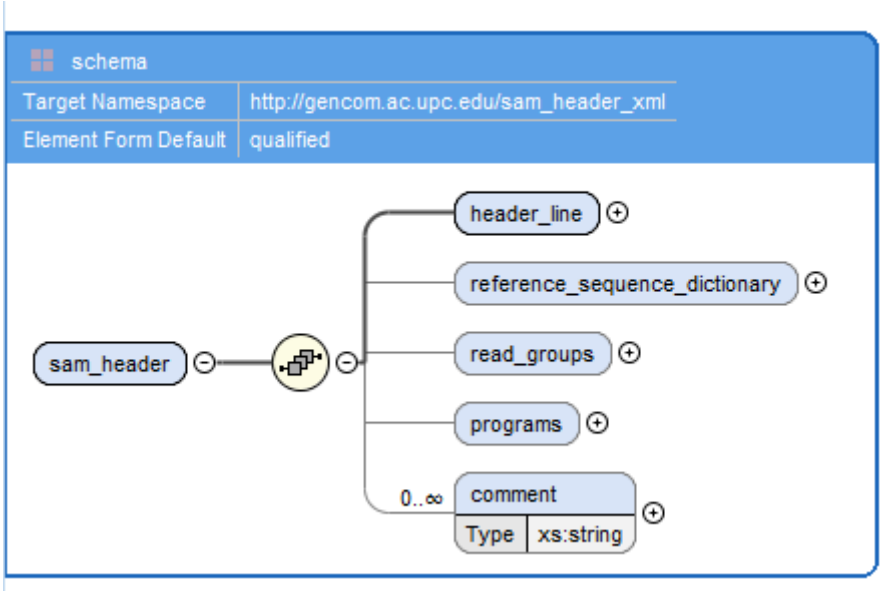


Figure 5 – Elements inside sam_header

The header_line element contains information regarding version, sorting_order and grouping, as shown in Figure 6. sorting_order has a restricted set of possible values: unknown, unsorted, queryname and coordinate. grouping also has some restricted values: none, query and reference.

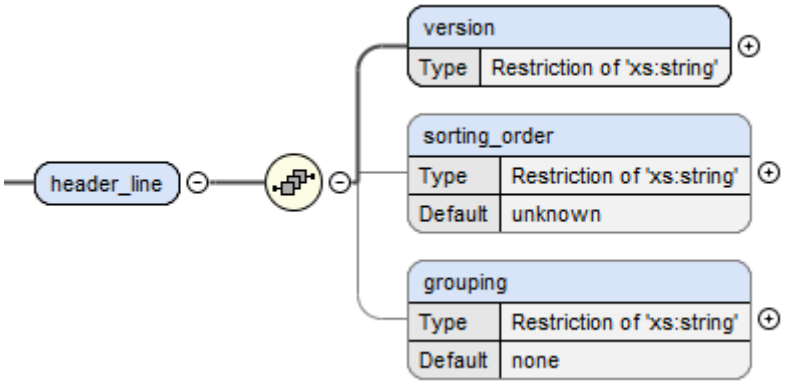


Figure 6 – Elements inside header_line

The reference_sequence_dictionary element shown in Figure 7 contains information about the reference_sequence and a unique_reference_id. The fields contained in reference_sequence are also shown. For more details on its content, see Annex I.

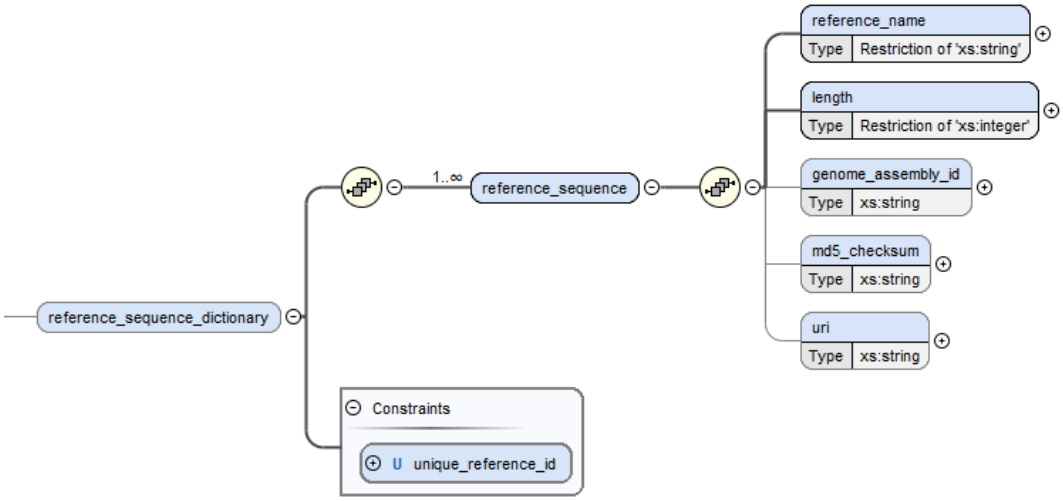


Figure 7 – Elements inside reference_sequence_dictionary

The read_groups element shown in Figure 8 may contain several read_group elements (shown in Figure 9). For more details on its contents, see Annex I.

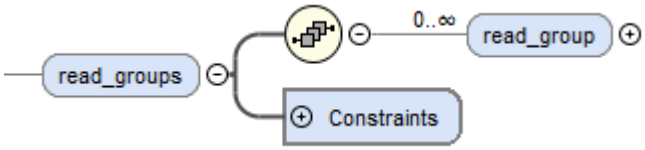


Figure 8 – read_groups elements

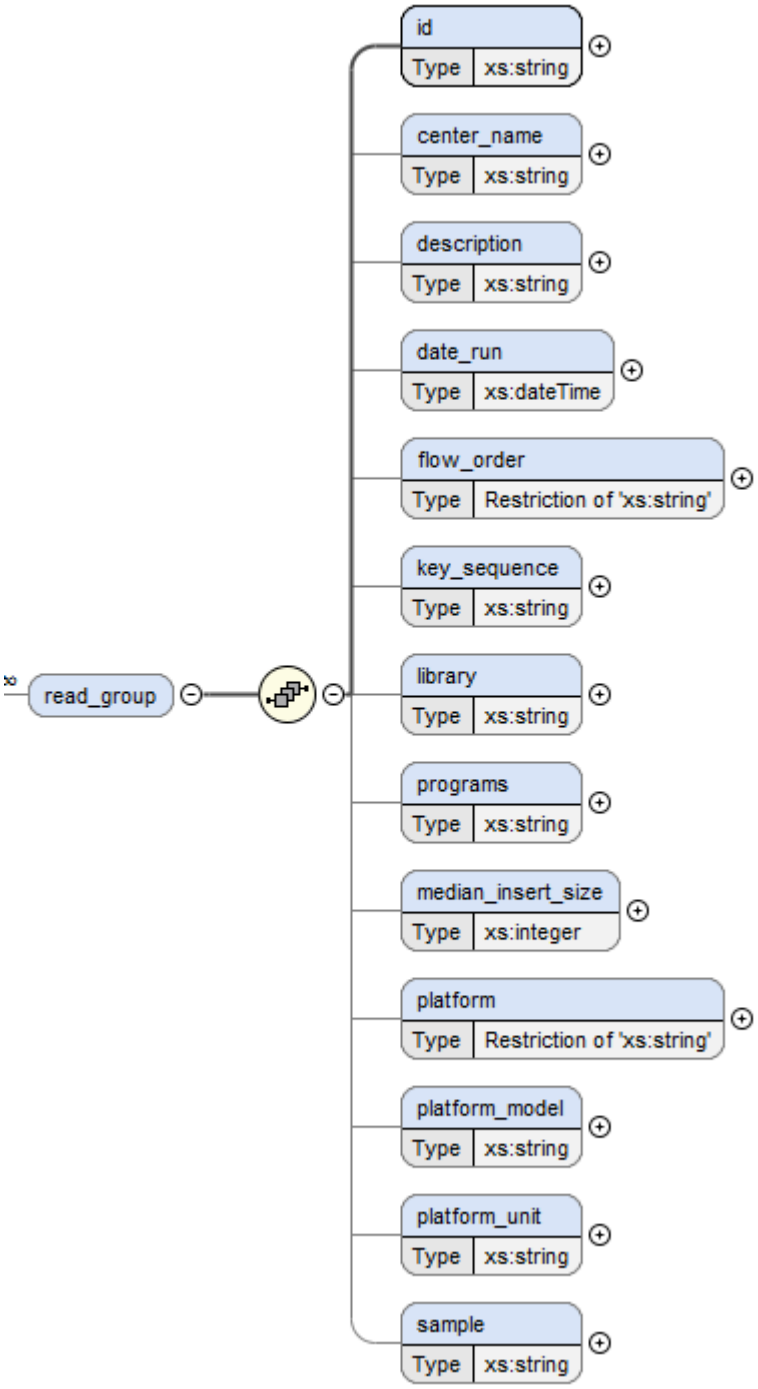


Figure 9 – read_group element complete structure

The programs element shown in Figure 10 may contain several program elements (also shown in Figure 10). For more details on its contents, see Annex I.

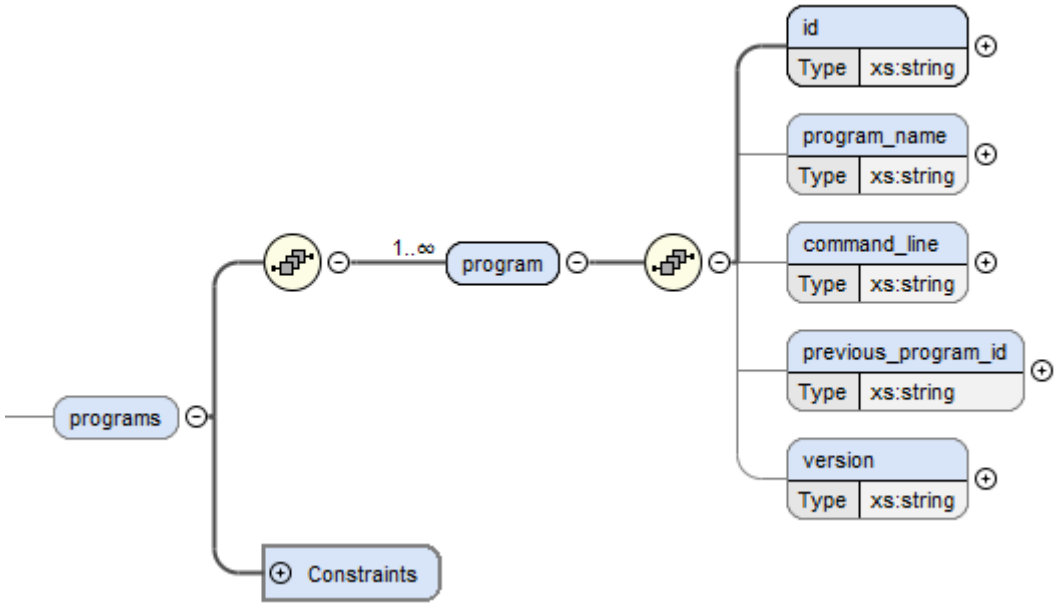


Figure 10 – programs element complete structure

6.3 Inclusion of other kind of metadata and semantic-related information in the GITL

6.3.1 EBI metadata

The European Bioinformatics Institute (EBI) [23] has identified some metadata fields required to publish in a relevant manner genomic data. The metadata format is dictated by the type of content, but the structure they have devised is close to the proposed GENIFF format.

All published information should belong to a Study. Within this study they propose to have multiple datasets each one with their own respective Data Access Committee (DAC) and Data access policy, alongside information about the Analysis. Extending the usage of the header boxes of GENIFF, we are able to integrate this information. For example we have already header information for the study which could at the very least be a pointer to information available online, as proposed by EBI in the case of a publication in the literature. In the case of the Data Access Committee and Data access policy, this information can be included into the XACML.

EBI also contemplates the need to give information on the samples: for example Experimental Factor Ontology references, phenotype traits but also gender and anonymized id. The GENIFF format supports also this specification: the header file can be extended to include this, and in the case of multiple individual the GENIFF format either allows to split in different mono-individual studies, or the `gedi` box could be leveraged to indicate for each dataset to which individual the data belongs to.

Further metadata more closely related to the actual nature of the stream can be placed in the `gedh` box, as shown previously in the case of the SAM file.

More specifically, for each study, the EBI's metadata consists of:
- The center name and its alias.
- A description of the study: its title, its type (based on a controlled vocabulary, if the desired option is not available there should me a request to add the new type), the abstract of the study, the name of the project, the project id and related studies.
- Links to information on the study.
- List of study attributes: each one is composed of a free tag name and a tag value.

The results of the experiments are then stored in datasets with the following metadata:
- Title and description
- Its type (e.g. whole genome sequencing)
- List of the runs belonging to this dataset
- List of the analysis belonging to this dataset
- List of the access policies belonging to this dataset
- List of related resources
- Extra attributes

The runs and the analysis both have their own metadata. Concerning Runs:
- Identification of the run
- Reference to the experiment
- Spot descriptor and platform
- Pointers to the files: filename, file type, quality score system, checksum, etc.
- Related runs and attributes

Concerning Analysis:
- Title, description
- Link to the study
- Link to the samples, and indication of which name they have within the actual file
- Information on the analysis type: for example Reference alignment (with information such as reference sequences used and their name within the BAM file)
- Files: information on the files used for the analysis (checksum, file type, filename,...)
- Analysis attributes

This metadata helps link to other descriptors: for example the samples the analysis link to have their own metadata:
- The center name and its alias
- Taxonomy information on the sample: NCBI taxon id, scientific name and common name
- A description of the sample
- Links to information on the study
- Sample attributes: the sample id, the gender (female, male, unknown), the phenotype (free text, but EFO term recommended) and more attributes if required

Or the experiment:
- The center name and its alias
- Title of the experiment
- Design of the experiment: description of the design, a link to the sample, a description of the library (the library's name, strategy, source and selection)

Finally, the dataset links to the policies:
- Identifier, and title in order to be able to search for the information
- A link to the DAC
- The actual policy: either the text, or the file, or a link
- Extra attributes

EBI requires a list of contact information for DAC (Data Access Committee):
- The center name and its alias
- Identifier, and title in order to be able to search for the information
- A list of contacts, each with the relevant information such as the telephone number, the organization, email and name and who the main contact is
- Links to other resources and extra attributes

By enabling the inclusion of the metadata defined by EBI in the different levels of the GENIFF we would allow the same set of metadata. There are, however, some conceptual differences to take into account. In a GENIFF file, the dataset refers to an aggregation of streams building one content. From the EBI point of view, a dataset is a collection of files (e.g. multiple BAM files, one for each analysis in the dataset). Therefore the metadata proposed for the experiments, runs and analysis is more relevant for the GENIFF's dataset level. For this reason also, the metadata defined for the EBI's dataset is relevant for the GENIFF's study. This does not mean that the EBI's study metadata should be discarded: the information it contains is relevant and should be placed alongside or at least referenced to.

By including EBI's metadata alongside the previously described SAM header, we have more description possibilities: for example a structured way to describe the phenotypes present in the study.

6.3.2 MIAME metadata

Also related to metadata definition for genomic information, we find MIAME [24]. MIAME is the minimum information necessary to be published for a Microarray Experiment. Although this type of information does not coincide with the use intended for GENIFF, some of the MIAME requirements are relevant. For example they require explaining the Experiment Design, with information such as the goal, the experimental factors: by using the EBI's metadata we already cover these requirements. This is also true for the samples of information where MIAME requires information on the samples (such as the species or the gender) and possible actions taken upon them for the preparation (which EBI also contemplates in the description of the library.

Other data required by MIAME is very much tied to the Microarray Experiments. Nevertheless, if the researcher considered it relevant for GENIFF's content too, the EBI metadata allows indicating it too within the samples attributes.

6.3.3 Genomic Standards Consortium metadata

The Genomic Standards Consortium (http://gensc.org/) has defined a standard checklist for 'Minimum Information about any (x) Sequence' (MIxS) to enhance the searchability and traceability of sequence data. Based on this checklist standard, we define a consensus set of metadata items that could be used to document the nature and biological origin of the sequence data and its biological source. Extending the usage of the header boxes of GENIFF, we are able to integrate this metadata information.

Concerning values and units, all dates, times, durations and intervals should be written in ISO 8601 formats.

Except a few cases, strict units are not defined for items in the MIGS/MIMS/MIENS (Minimum Information about a Genome Sequence / Minimum Information about a Metagenomic Sequence/Sample / Minimum Information about an ENvironmental Sequence) checklists, wherever applicable the unit of choice should accompany the value of an item. The units should be in accordance with the The International System of Units (SI).

The table in Annex II describes metadata items that could be documented together with the sequence data. Additional metadata can also be included to further define the nature and environmental context of the biological source and the methodology of the sequencing underlying the data.

7 Security and Privacy for Genomic information

7.1 Privacy aspects

The nature of genomic information makes security and privacy elements a key aspect when providing mechanisms for generating, processing, storing and transmitting it. For this reason, we have identified a number of privacy aspects especially relevant. They are listed below, separated into four categories: Granularity of the information, roles when accessing information, usage and if it is required to inform the owner of the data. The combination of these privacy aspects will lead us to the definition of use cases, like the ones presented in Section 7.2.

Privacy aspects identified:

- Required granularity for genomic information. Inside this category, we can find different levels of granularity: Complete file, one chromosome, range of regions or even one specific position. It also depends on the kind of genomic information being accessed: sequenced or aligned information. Other genomic information formats, like variant/genotype information stored in a variant calling file could be also considered in this category.
- To whom the access to genomic information is given. Here, some roles have been identified: Owner, analyst, genomic information custodian, health care professional or researcher.
- Usage of genomic information. Inside this category, we foresee the definition of different purposes: Full control, research, characterization of individual for a given biological feature, genetic analysis, lineage search, commercial, forensics, etc.
- Provide information to genomic information owner (data sharer). Here, we identify the possibility to inform of the result of the study to the data sharer.

The combination of different privacy aspects gives room to several use cases. Some of them are defined in the next section.

7.2 Use cases

This section presents a list of use cases where the privacy aspects concerning granularity of information, persons with granted access, genomic information usage and if it is required to inform of the results to the data owner are combined.

The use cases are organized into five different types based on the initiator role involved, grouping several aspects. So, first type of use cases groups those where the permission to access the genomic information is given by an individual. The second type includes those use cases where permission is requested by a data analyst. The third type involves healthcare professionals, the fourth type describes those use cases where researchers are involved and there is a fifth type for other kind of use cases. It is worth noting that other groupings are possible, this is just a way of organizing them.

Use cases where the permission is given by an individual:

1.1 An individual wants to share the variant/alignment/sequence information of specific genomic regions and experiment with an analyst, so that he/she performs a genetic analysis.

1.2 Analogously to 1, but combining data of several experiments.

1.3 An individual wants to share the non-aligned reads from a given experiment with an analyst, so that he/she performs a genetic analysis.

1.4 An individual wants to share variant/alignment/sequence information genome wide for a given experiment with an analyst, so that he/she performs a genetic analysis.

1.5 Analogously to 4 but combining data of several experiments.
1.6 An individual wants to share his variant/alignment/sequence genome wide data with a researcher for a given project. He/she wants to be re-contacted back if necessary.
1.7 Analogously to 6, but he/she wants to do this anonymously.
1.8 An individual wants to donate his/her genome for research when he/she dies.
1.9 An individual wants to share access of his whole genome with his trusted healthcare professional for an undetermined amount of time, being able to cancel that access whenever he/she likes.

Use cases where the permission is requested by an analyst:
2.1 An analyst wants to access the variant/alignment/sequence information of specific genomic regions from a given individual and experiment.
2.2 Analogously to 2, but combining data of several experiments.
2.3 An analyst requires access to non-aligned reads from a given individual.
2.4 An analyst wants to access the variant/alignment/sequence information genome wide from a given individual and experiment.
2.5 Analogously to 4 but combining data of several experiments.

Use cases where a healthcare professional is involved:
3.1 An analyst wants to share with a healthcare professional the outcome of a genetic analysis.
3.2 A healthcare professional wants to access the genetic analysis of an individual.

Use cases where the permission is requested by a researcher:
4.1 A researcher wants to access anonymized genomic data for a given project.
4.2 A researcher wants to ask an individual for more experiments to be used in a research study.

Other use cases:
5.1 Paternity test
5.2 Forensic uses

From the above uses cases, several privacy rules can be defined, including information about:
- who is giving the permission to access to some piece of genomic information
- to whom the permission is given
- the time frame for the permission
- the operations permitted
- the purpose
- even if the data sharer has to be informed of the result of the analysis performed.

To define the privacy rules, the eXtensible Access Control Markup Language (XACML) [25] can be used, as already proposed in [26]. Using such a language, it is possible to create rules with the required level of detail. In Section 7.3 gives more details on using this language, how to define rules and how to include them into different genomic information formats. Table 2 shows a summary of the use cases presented.

Table 2 – Use cases summary

Use Case	Role giving permission	Role receiving permission	Role initiative	Object of the permission	Permission	Experiments number	Inform owner?
1.1	Individual	Analyst	Giving	Regions of Variant / Alignment / Sequence	Genetic analysis	One	Yes, results
1.2	Individual	Analyst	Giving	Regions of Variant / Alignment / Sequence	Genetic analysis	Several	Yes, results
1.3	Individual	Analyst	Giving	Non-aligned reads	Genetic analysis	One	Yes, results
1.4	Individual	Analyst	Giving	Complete genome Variant / Alignment / Sequence	Genetic analysis	One	Yes, results
1.5	Individual	Analyst	Giving	Complete genome Variant / Alignment / Sequence	Genetic analysis	Several	Yes, results
1.6	Individual	Researcher	Giving	Complete genome Variant / Alignment / Sequence	Research project	One	Yes, re-contact
1.7	Individual	Researcher	Giving	Complete genome Variant / Alignment / Sequence	Research project	One	No, anonymous
1.8	Individual	Everyone	Giving	Complete genome Variant / Alignment / Sequence	Donation	-	No, it is donated after individual's death
1.9	Individual	Healthcare	Giving	Complete genome Variant / Alignment / Sequence	View	-	-
2.1	Individual	Analyst	Receiving	Regions of Variant / Alignment / Sequence	View	One	-
2.2	Individual	Analyst	Receiving	Regions of Variant / Alignment / Sequence	View	Several	-
2.3	Individual	Analyst	Receiving	Non-aligned reads	View	One	-
2.4	Individual	Analyst	Receiving	Complete genome Variant / Alignment / Sequence	View	One	-
2.5	Individual	Analyst	Receiving	Complete genome Variant / Alignment / Sequence	View	Several	-
3.1	Analyst	Healthcare	Giving	Outcome genetic analysis	Share	One	-
3.2	Individual	Healthcare	Receiving	Genetic analysis	View	One	-
4.1	Owner/Custodian	Researcher	Receiving	Anonymized genomic data	Research project	One/Several	No, anonymous
4.2	Individual	Researcher	Receiving	More experiments	Research project	Several	Yes, request more data
5.1	Individual	Analyst	Giving	Perform paternity test	Paternity test	One	Yes, results
5.2	Owner	Forensics	Receiving	Complete genome	Forensic	One/Several	Unknown

Table 2 summarizes the information contained in each use case, indicating which is the role that has to give the permission, the role receiving it, who has the initiative when requesting permission (giving or receiving), the object of the permission to be given (regions, complete genome, etc.), which permission is requested, the number of experiments involved and if the owner of the genome has to be informed. For the use case 1.9, there is a condition that is that the permission is given for an undetermined amount of time, but the individual can cancel access at any time. It has not been added for clarity.

7.3 Security and Privacy elements

In order to provide security and privacy for the storage and processing of genomic information, several elements should be defined.

Concerning security, multiple strategies have been devised in academia for protecting genomic information. However, based on the intended usage for this file format, we propose to only focus on encrypting the file. By encrypting the content of tags, non-legit access to them is avoided. However, encryption of certain regions should not interfere with the behavior of non-protected content. In this sense, all boxes marked as required or containing a child which is allowed to be read should remain non-encrypted. Not only the genomic data might be encrypted, other fields such as the metadata can be compromised: the SAM header lists the programs which have been used to generate the file or the phenotype data could reveal too much about the sequenced person.

The content of the `gdfd` box is the most relevant for encryption. As explained in Section 3, there are multiple formats available to code its contents. Encrypting this content does not require more than a flag to indicate the encrypted nature of the content, and enough information to obtain the key (and additionally the Initialization Vector, if needed). Currently, repositories of genomic data, allow the download of entire files, e.g. BAM files. Due to the size of the content it would be better not to reencrypt the content for each individual obtaining the file, but rather distributing the content with its original encryption. If the user is interested in the whole content, the XACML rules governing the genomic file will let him know if he can ask for it. If yes, and after clearance of the data's owner, the requester obtains the key through a mechanism to be defined.

Certain compression mechanisms considered in this document already contemplate the need for encryption: e.g. FAPEC [7][8] supports AES-256 encryption, allowing the confidentiality of each chunk included in `gdfd`. For other codifications, an encryption method allowing random-access on block level (e.g. AES-256-CTR) should be preferred to preserve the features offered by certain formats. One of such is the BAM format, where through the usage of the index file, one can seek precisely a specific region without parsing the whole file. However, if the decision is taken to encrypt only certain portions of the content within the `gdfd` box, then other approaches have to be devised. Here follows an example using the BAM format.

When the result of an alignment is codified into BAM, the mid-step byte array (resulting of expressing the SAM content in a binary format) is compressed using independent blocks, each one for a given chunk of the byte array. By encrypting the content of the blocks (the specification refers to this region by CDATA), selective protection can be achieved: for example the encrypted block contains alignments for certain regions which should remain secret. The information we need to maintain in order to enable the decryption is rather small compared to the size of the BAM file. In the example selected by MPEG (ID05 [27]), we need to maintain the state of 248k blocks, for which a bit array of 31k bytes is enough. Through the use of encryption in CTR mode, we maintain the random access features of the original file, reducing even further the drawbacks of having encrypted regions of the file.

However, the structure of the proposed format allows strategies to further reduce the difficulties associated with not having access to given blocks. One of the main drawbacks which comes to mind, is that in the case where the reads where not sorted, the encryption of a block will deny access to reads intended to remain public. Even in the case where the data is sorted by coordinates, one needs to perform certain corrections to recreate a BAM file with the plaintext available. The solution for this is to split the records in multiple BAM files, which are dealt with as the

information for different streams for the one dataset, i.e. the original BAM file. For example, if the task is to encode the content of a BAM file with data from three reference genomes (chr1, chr2, chr3, and unaligned data), we can split this file into four (e.g. BAM_chr1). Each stream is then stored as a stream of the original's file Dataset. With such an encoding, the datasets can have different encryption strategies (either the whole stream or portions of it), and different rules. And is much easier to recover a functional BAM file even if certain streams are protected: all the ones we have information for are decoded and fully functional on their own, and then can be merged together with one of the existing tools. In the case of having the keys to decrypt other regions, the resulting merge will contain more information.

The encryption metadata might need more information in the case where modifications on the file might be possible. If the modifications where to affect the encrypted regions, comparing which blocks varied and which not might allow for inferences on the content.

A special case of modification is the realignment of the data: in this case the data is read and maybe moved elsewhere, possibly going from an encrypted region to an unencrypted region. The correct behavior when performing this operation has to be defined in order to limit the risk of known plaintext attacks.

Concerning privacy, one of the mechanisms foreseen to fulfill with the requirements defined in [4] is the definition of privacy rules. To do so, we propose the use of the XACML language, as presented in [26]. As already explained in Section 6.2, this language gives the possibility of defining privacy rules with a detail. An example of rule can be found in Annex III. There are several reasons for using this kind of rules for the protection of genomic information; some of them are sketched next:

- Usage and access rules storage and location. The size of the rules is negligible when compared with the size of genomic information. In this way, the inclusion of rules inside the genome file format should not be a problem, even if several rules are defined.
- Security and privacy. These rules can be used to both describe security (encryption techniques, access from external systems, etc.) and privacy (who can access, when, for what purpose, who has to be informed, etc.) features associated to the genome. In this way, the security and privacy requirements already identified could be fulfilled by an application/system accessing to the genome file, which should be aware of them. It is worth noting that the rules would permit the description of allowed usages of the genome, which could be checked prior to performing any scientific test to avoid misuses from scientific community as reported in [28].
- Genome search services based on access and usage rules. With the inclusion of rules in genomic information, one can imagine search services based on the permitted uses of the genomes. In this way, the scientific community may have a wider genome base for their studies whilst individuals may provide access to their genome with more confidence.
- Impact on processing of genomic information. The addition of rules should not affect the genomic information processing, although it may govern its usage. They should be included into the formats allowing governance checking before further processing.

8 Software tools

8.1 GENIFF generator

The GENIFF generator application is intended to generate the structure of the ISO BMFF. To do so, the structure of boxes is replicated in a tree structure. This organization can be either programmatically constructed or obtained through the parsing of the file, depending whether the current action is to encode information in a GENIFF file or to decode it.

As the root node of the structure, there is the ISO BMFF file, to which we append child boxes, until the desired structure is completed. There are different classes which inherit from the Box definition, allowing for specialization. For example, the `ftyp` class is one of this child classes intended to encode one type of box: in this case it is the `ftyp` box. Other examples are the `full` box and the `meta` box which are defined to give service to the box with the same names as defined by ISO, or the `gedh` box of GENIFF.

Alongside this specialized boxes, there are other classes which are used for the other cases: for example the `SimpleBox` definition which provides only the functionalities of storing an aggregation of child boxes, or the `CharArrayBox` and `FromFileBox` which allow, respectively, to write the content of a char array in the file, or to copy a portion of a file to the output document.

If encoding data into a GENIFF file, once the structure is defined using the previously described classes, a single call to the root node starts the process of writing the file to the output stream. When encoding, the program reads the task description from a file: which file form the datasets which form the study. It is based on this information that the tree structure is constructed.

If the action performed is to parse a GENIFF file, we use the parse file method from the `BoxFactory` to perform the action. The code does not infer the proper box class to use, but rather uses lookup tables to decide the action to perform. For example, we expect all `gdfd` boxes (the box storing the byte array forming the data) to be rather heavy. This is why we use a `FromFileBox` to model them: we only keep a reference to the file location (i.e. filename, seek point and end point), allowing to copy its content to another file when required without occupying too much memory.

In order to demonstrate the capacities of encrypting BAM, there is another class `BAMFile`, which is used to parse the content, searching which blocks should be encrypted (currently the decision is based on whether or not the block contains a record from a reference sequence which should remain secret). The output of this process is the information stored in the bit array informing on the encrypted status of each block. It is also used for the dedicated `FromBAMFileBox`, which replicates the functionalities of the `FromFileBox`, but expanded to allow the encryption while writing.

When decoding a GENIFF file, once the structure has been parsed, the program outputs the contents of the `gdfd` into files the name of which are given in the dataset header (`gedh`), and if present the indexing contents too.

Annexes V and VI provides more implementation and use details.

8.2 XACML authorization for genomic information access

In order to authorize access to genomic information, we have developed a proof of concept application completely described in Annexes IV and VI.

In this section, we describe the concepts that allow an XACML-based authorization to access genomic information integrated in the current application flow. We have added a new step, as shown in Figure 11, where the XACML-based authorization gives access to the genomic information only when the user has the permission to do so. The permission is expressed with XACML policies, which specify rights and conditions associated with genomic information. An example of XACML policy can be found in Annex III.

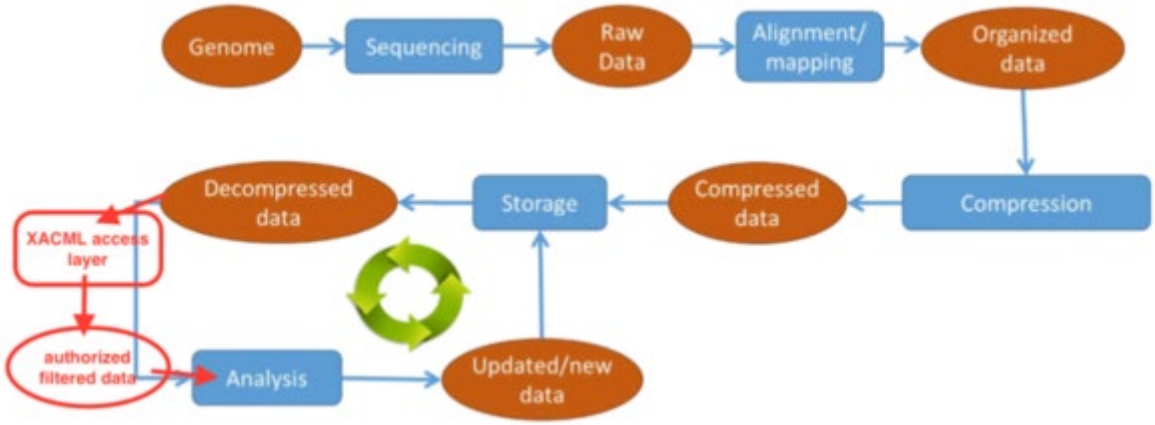
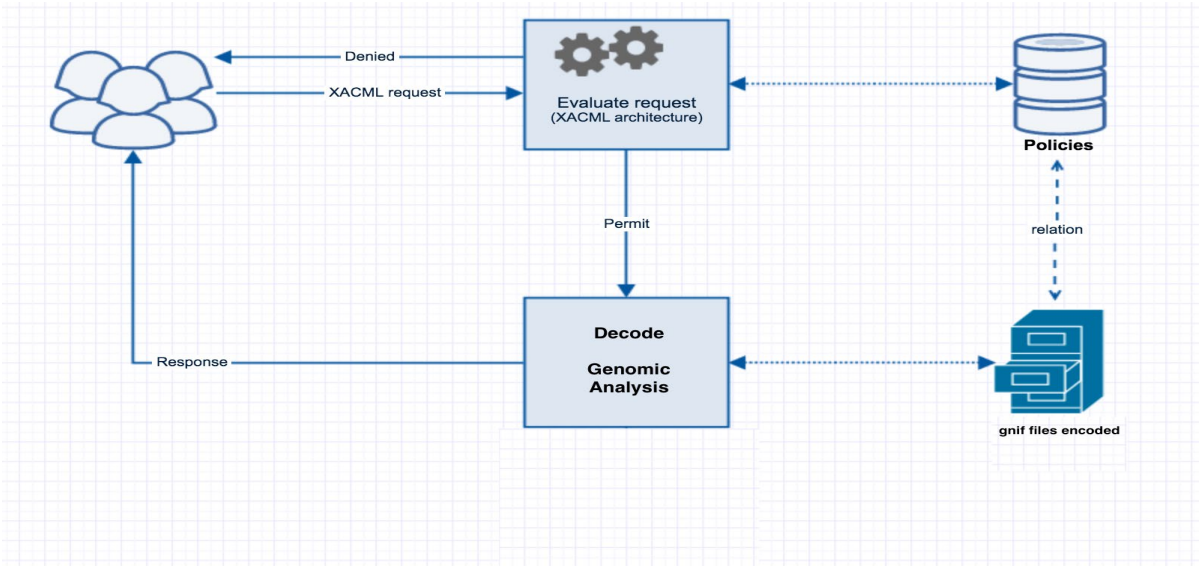


Figure 11 – Complete flow for genomic data, new step added

Figure 12 shows the process of evaluation of access rights:
1. A user sends a request which is intercepted by XACML engine
2. The XACML engine converts the request into a XACML authorization request
3. The XACML engine evaluates the authorization request against the policies that the file requested is configured with.
4. The XACML engine reaches a decision (Permit / Deny / NotApplicable / Indeterminate).
5. In the case that the decision result is equal to "Permit" the file can be decoded an analyzed by



the user.

Figure 12 – Authorization flow proposed

9 Transport coverage matrix

This section describes the coverage matrix defined in [29] for proposals covering Transport requirements.

Req ID	Requirement for transport	Justification	Compliance (Yes/No)
3.1	The compression process shall support the assessment of integrity	Use of CRC or other technique of file checking. Header boxes can support this information.	Y
3.2	The solution shall allow conveying information enabling data protection	The solution provides the possibility of encrypting genomic information whilst providing random access. Moreover, access control rules can be defined by means of XACML and applied to protect access to data at different levels.	Y
3.3	The solution shall allow conveying information enabling accountability and traceability	It is possible to define who has to be informed of accesses done over some genomic information by means of XACML rules.	Y
3.4	The solution shall allow conveying information enabling transparency	XACML rules allow the definition of operations that can be done over some genomic information, like usage restrictions or roles. Combined with the report of usage described in 3.3, we cover this requirement.	Y
3.5	The solutions shall support compressed data streaming.	Compression of the data into a block structure allows sending the data whilst it is being processed, as blocks are independently decompressed and decrypted (if required).	Y

10 Acknowledgements

The work done in this proposal has been partially supported by the Spanish Government under the project: Secure Genomic Information Compression (GenCom, TEC2015-67774-C2-1-R and TEC2015-67774-C2-2-R), the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 305444 (RD-Connect), a fellowship from the Basque Government, a Stanford Graduate Fellowships Program in Science and Engineering, the Stanford Data Science Initiative (SDSI), and an NIH grant with number 1 U01 CA198943-01.

11 References

[1] ISO/IEC JTC 1/SC 29/WG 11 - ISO/TC 276/WG 5 MPEG2016/N16320 - Joint Call for Proposals for Genomic Information Compression and Storage, Geneva, June 2016.
[2] ISO/IEC IS 14496-12 - Information technology - Coding of audiovisual objects - Part 12: ISO base media file format, Fifth edition, December 2015.
[3] Łukasz Roguski, Paolo Ribeca, CARGO: effective format-free compressed storage of genomic information, Nucleic Acids Research (2016), doi: 10.1093/nar/gkw318, http://nar.oxfordjournals.org/content/early/2016/04/29/nar.gkw318.full
[4] ISO/IEC JTC 1/SC 29/WG 11 - ISO/TC 276/WG 5 MPEG2016/N16323 - Requirements on Genomic Information Compression and Storage, Geneva, June 2016.
[5] Szymon Grabowski, Sebastian Deorowicz, Lukasz Roguski, Disk-based compression of data from genome sequencing. Bioinformatics. 2015 May 1; 31 (9):1389-95
[6] MPEG2016/M39176 – A proposal for compression technology based in ORCOM and QVZ, Cheng-du, October 2016.
[7] Portell, J., Villafranca, A. G., and García-Berro, E., Quick outlier-resilient entropy coder for space missions, Journal of Applied Remote Sensing 4, 339–363, 2010.
[8] MPEG2016/M39179 – A proposal for the compression algorithm based in FAPEC, Cheng-du, October 2016.
[9] Greg Malysa, et al. QVZ: lossy compression of quality values. Bioinformatics. 2015 Oct 1; 31(19):3122-9.
[10] Peter J. A. Cock, et al., The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Res. 2010 Apr; 38(6): 1767–1771.
[11] Heng Li, et al., The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009 Aug 15; 25(16): 2078–2079.
[12] Official Sequence Alignment/Map (SAM) Format Specification, https://samtools.github.io/hts-specs/
[13] ISO/IEC JTC 1/SC 29/WG 11 N16145, Database for Evaluation of Genomic Information Compression and Storage, San Diego, USA, February 2016.
[14] ISO/IEC JTC 1/SC 29/WG 11 N15346, Investigation on genomic information compression and storage, Geneva, Switzerland, February 2015.
[15] ISO/IEC JTC 1/SC 29/WG 11 N15527, Genome Compression 101 - Tutorial on Genome Compression and Storage, Warsaw, Poland, June 2015.
[16] Official Variant Call Format (VCF) Format Specification, https://samtools.github.io/hts-specs/
[17] Łukasz Roguski, Sebastian Deorowicz, DSRC 2-Industry-oriented compression of FASTQ files, Bioinformatics. 2014 Aug 1; 30(15):2213-5. doi: 10.1093/bioinformatics/btu208. Epub 2014 Apr 18.
[18] Daniel C. Jones, et al., Compression of next-generation sequencing reads aided by highly efficient de novo assembly, Nucleic Acids Res. 2012 Dec; 40(22): e171. doi: 10.1093/nar/gks754

[19] Faraz Hach, et al., SCALCE: boosting sequence compression algorithms using locally consistent encoding, Bioinformatics. 2012 Dec 1; 28(23): 3051–3057. doi: 10.1093/bioinformatics/bts593
[20] James K. Bonfield and Matthew V. Mahoney, Compression of FASTQ and SAM Format Sequencing Data, PLoS One. 2013; 8(3): e59190. doi: 10.1371/journal.pone.0059190
[21] MPEG2016/M39303 – A proposal for compression technology based in CBC, Cheng-du, October 2016.
[22] I. Ochoa, M. Hernaez and T. Weissman, Aligned genomic data compression via improved modeling, Journal of bioinformatics and computational biology, Vol. 12, No. 6, 2014.
[23] European Bioinformatics Institute (EBI), http://www.ebi.ac.uk/
[24] Brazma A1 et al., Minimum information about a microarray experiment (MIAME)-toward standards for microarray data, Nat Genet. 2001 Dec; 29(4): 365-71. https://www.ncbi.nlm.nih.gov/pubmed/11726920
[25] OASIS Standard, eXtensible Access Control Markup Language (XACML) Version 3.0, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
[26] Jaime Delgado, Silvia Llorente, MPEG2015/M36405 - Some application scenarios for privacy and security requirements on genome usage, compression, transmission and storage, Warsaw, June 2015.
[27] ISO/IEC JTC 1/SC 29/WG 11 - ISO/TC 276/WG 5 N16322/N96 - Database for Evaluation of Genome Compression and Storage, Geneva, 2016.
[28] Sarah Zielinski, “Henrietta Lacks’ ‘Immortal’ Cells”, report on Rebecca Skloot book “The Immortal Life of Henrietta Lacks”, http://www.smithsonianmag.com/science-nature/henrietta-lacks-immortal-cells-6421299/
[29] ISO/IEC JTC 1/SC 29/WG 11 - ISO/TC 276/WG 5 MPEG2016/N16321 - Evaluation Procedure for the Joint Call for Proposals on Genomic Information Compression and Storage, Geneva, June 2016.
[30] Balana implementation. http://svn.wso2.org/repos/wso2/trunk/commons/balana/
[31] Samtools. http://www.htslib.org/
[32] Example of XACML policy used to authorize access specific part of a SAM file, https://github.com/saracubillas/samtools/blob/develop/privacy/policy/XACMLPolicy.xml
[33] Source code of the authorization application, integrated with samtools. https://github.com/saracubillas/samtools/tree/develop/privacy

Annex I – SAM Header XML schema

XML Schema for describing SAM Header fields and one example of use.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://gencom.ac.upc.edu/sam_header_xml"
    xmlns="http://gencom.ac.upc.edu/sam_header_xml"
    xmlns:ns="http://gencom.ac.upc.edu/sam header xml" elementFormDefault="qualified">
    <xs:element name="sam header">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="header_line">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="version">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[0-9]+\.[0-9]+"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="sorting_order"
                                        minOccurs="0" maxOccurs="1" default="unknown">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:enumeration value="unknown"/>
                                        <xs:enumeration value="unsorted"/>
                                        <xs:enumeration value="queryname"/>
                                        <xs:enumeration value="coordinate"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="grouping"
                                        minOccurs="0" maxOccurs="1" default="none">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:enumeration value="none"/>
                                        <xs:enumeration value="query"/>
                                        <xs:enumeration value="reference"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="reference_sequence_dictionary" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="reference sequence" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="reference name">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                    <xs:pattern value="[!-)+-~][!-~]*"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                        <xs:element name="length">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:integer">
                                                    <xs:minInclusive value="1"/>
                                                    <xs:maxInclusive value="4294967295"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                        <xs:element name="genome assembly id"
                                            type="xs:string" minOccurs="0" maxOccurs="1"/>
                                        <xs:element name="md5_checksum" type="xs:string"
                                            minOccurs="0" maxOccurs="1"/>
                                        <xs:element name="uri" type="xs:string"
                                            minOccurs="0" maxOccurs="1"/>
                                    </xs:sequence>
                                </xs:complexType>
```

```
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
            <xs:unique name="unique_reference_id">
                <xs:selector xpath="ns:reference_sequence/ns:reference_name"/>
                <xs:field xpath="."/>
            </xs:unique>
        </xs:element>
        <xs:element name="read_groups" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="read group" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="id" maxOccurs="1" type="xs:string"/>
                                <xs:element name="center_name" minOccurs="0"
                                    maxOccurs="1" type="xs:string"/>
                                <xs:element name="description" minOccurs="0"
                                    maxOccurs="1" type="xs:string"/>
                                <xs:element name="date_run" minOccurs="0"
                                    maxOccurs="1" type="xs:dateTime"/>
                                <xs:element name="flow_order" minOccurs="0"
                                    maxOccurs="1">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:pattern value="\*|[ACMGRSVTWYHKDBN]+"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="key sequence" minOccurs="0"
                                    maxOccurs="1" type="xs:string"/>
                                <xs:element name="library" minOccurs="0" maxOccurs="1"
                                    type="xs:string"/>
                                <xs:element name="programs" minOccurs="0" maxOccurs="1"
                                    type="xs:string"/>
                                <xs:element name="median insert size" minOccurs="0"
                                    maxOccurs="1" type="xs:integer"/>
                                <xs:element name="platform" minOccurs="0"
                                    maxOccurs="1">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:enumeration value="CAPILLARY"/>
                                            <xs:enumeration value="LS454"/>
                                            <xs:enumeration value="ILLUMINA"/>
                                            <xs:enumeration value="SOLID"/>
                                            <xs:enumeration value="HELICOS"/>
                                            <xs:enumeration value="IONTORRENT"/>
                                            <xs:enumeration value="ONT"/>
                                            <xs:enumeration value="PACBIO"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="platform model" minOccurs="0"
                                    maxOccurs="1" type="xs:string"/>
                                <xs:element name="platform_unit" minOccurs="0"
                                    maxOccurs="1" type="xs:string"/>
                                <xs:element name="sample" minOccurs="0" maxOccurs="1"
                                    type="xs:string"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
            <xs:unique name="unique read id">
                <xs:selector xpath="ns:read group/ns:id"/>
                <xs:field xpath="."/>
            </xs:unique>
        </xs:element>
        <xs:element name="programs" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="program" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="id" type="xs:string"/>
                                <xs:element name="program_name" type="xs:string"
```

```
                                        minOccurs="0"/>
                                    <xs:element name="command line" type="xs:string"
                                        minOccurs="0"/>
                                    <xs:element name="previous_program_id" type="xs:string"
                                        minOccurs="0"/>
                                    <xs:element name="version" type="xs:string"
                                        minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
                <xs:unique name="unique program id">
                    <xs:selector xpath="ns:program/ns:id"/>
                    <xs:field xpath="."/>
                </xs:unique>
                <xs:key name="PKProgram">
                    <xs:selector xpath="ns:program/ns:id"/>
                    <xs:field xpath="."/>
                </xs:key>
                <xs:key name="KPreviousProgram">
                    <xs:selector xpath="ns:program/ns:previous_program_id"/>
                    <xs:field xpath="."/>
                </xs:key>
                <xs:keyref name="FKPreviousProgramToProgram" refer="PKProgram">
                    <xs:selector xpath="ns:program/ns:previous_program_id"/>
                    <xs:field xpath="."/>
                </xs:keyref>
            </xs:element>
            <xs:element name="comment" type="xs:string" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Example of SAM Header following the XML Schema.

```
<?xml version="1.0"?>
<sam_header xmlns="http://gencom.ac.upc.edu/sam_header_xml">
        <header line>
                <version>0.0</version>
                <sorting order>coordinate</sorting order>
                <grouping>query</grouping>
        </header_line>
        <reference_sequence_dictionary>
                <reference sequence>
                        <reference name>ref</reference name>
                        <length>4294967295</length>
                </reference_sequence>
                <reference_sequence>
                        <reference_name>ref2</reference_name>
                        <length>47</length>
                </reference sequence>
        </reference sequence dictionary>
        <read_groups>
                <read_group>
                        <id>hola</id>
                        <flow order>*</flow order>
                </read group>
                <read_group>
                        <id>hola2</id>
                        <flow_order>*</flow_order>
                </read group>
        </read groups>
        <programs>
                <program>
                        <id>ref</id>
                </program>
                <program>
                        <id>ref2</id>
                        <previous_program_id>ref</previous_program_id>
                </program>
        </programs>
</sam_header>
```

Annex II - Documentation of metadata to a genomic sequence according to the Minimum Information about any (x) Sequence (MIxS) checklist

The following table describes metadata items that could be documented together with the sequence data. Additional metadata can also be included to further define the nature and environmental context of the biological source and the methodology of the sequencing underlying the data.

Column 1 - **Name**: name of a checklist item (e.g. as it appears in GenBank structured comments)
Column 2 - **Item**: full name of item as it appears in the publication
Column 3 - **Definition**: a description of the item, including links to ontologies and other resources that can be used to fill in values for the item
Column 4 – **Expected value**: examples of values for an item
Column 5 - **Section**: the section of an item
Column 6 - **Value syntax**: the proper syntax for writing the value for a given item

Name	Item	Definition	Expected value	Section	Value syntax
submitted_to_insdc	submitted to insdc	Depending on the study (large-scale e.g. done with next generation sequencing technology, or small-scale) sequences have to be submitted to SRA (Sequence Read Archive), DRA (DDBJ Read Archive) or via the classical Webin/Sequin systems to Genbank, ENA and DDBJ. Although this field is mandatory, it is meant as a self-test field, therefore it is not necessary to include this field in contextual data submitted to databases	boolean	investigation	{boolean}
investigation_type	investigation type	Nucleic Acid Sequence Report is the root element of all MIGS/MIMS compliant reports as standardized by Genomic Standards Consortium. This field is either eukaryote,bacteria,virus,plasmid,organelle, metagenome,mimarks-survey, or mimarks-specimen	eukaryote, bacteria_archaea, plasmid, virus, organelle, metagenome,mimarks-survey or mimarks-specimen	investigation	[eukaryote|bacteria_archaea|plasmid|virus|organelle|metagenome|mimarks-survey|mimarks-specimen]
project_name	project name	Name of the project within which the sequencing was organized		investigation	{text}
experimental_factor	experimental factor	Experimental factors are essentially the variable aspects of an experiment design which can be used to describe an experiment, or set of experiments, in an increasingly detailed manner. This field accepts ontology terms from Experimental Factor Ontology (EFO) and/or Ontology for Biomedical Investigations (OBI). For a browser of EFO (v 2.43) terms, please see http://purl.bioontology.org/ontology/EFO; for a browser of OBI (v 2013-10-25) terms please see http://purl.bioontology.org/ontology/OBI	text or EFO and/or OBI	investigation	{term|text}
lat_lon	geographic location (latitude and longitude)	The geographical origin of the sample as defined by latitude and longitude. The values should be reported in decimal degrees and in WGS84 system	decimal degrees	environment	{float} {float}

depth	geographic location (depth)	Please refer to the definitions of depth in the environmental packages	-	environment	-
alt_elev	geographic location (altitude/elevation)	Please refer to the definitions of either altitude or elevation in the environmental packages	-	environment	-
geo_loc_name	geographic location (country and/or sea,region)	The geographical origin of the sample as defined by the country or sea name followed by specific region name. Country or sea names should be chosen from the INSDC country list (http://insdc.org/country.html), or the GAZ ontology (v 1.512) (http://purl.bioontology.org/ontology/GAZ)	country or sea name (INSDC or GAZ):region(GAZ) :specific location name	environment	{term}:{term}:{text}
collection_date	collection date	The time of sampling, either as an instance (single point in time) or interval. In case no exact time is available, the date/time can be right truncated i.e. all of these are valid times: 2008-01-23T19:23:10+00:00; 2008-01-23T19:23:10; 2008-01-23; 2008-01; 2008; Except: 2008-01; 2008 all are ISO8601 compliant	date and time	environment	{timestamp}
env_biome	environment (biome)	Biomes are defined based on factors such as plant structures, leaf types, plant spacing, and other factors like climate. Biome should be treated as the descriptor of the broad ecological context of a sample. Examples include: desert, taiga, deciduous woodland, or coral reef. EnvO (v 2013-06-14) terms can be found via the link: www.environmentontology.org/Browse-EnvO	EnvO	environment	{term}
env_feature	environment (feature)	Environmental feature level includes geographic environmental features. Compared to biome, feature is a descriptor of the more local environment. Examples include: harbor, cliff, or lake. EnvO (v 2013-06-14) terms can be found via the link: www.environmentontology.org/Browse-EnvO	EnvO	environment	{term}
env_material	environment (material)	The environmental material level refers to the material that was displaced by the sample, or material in which a sample was embedded, prior to the sampling event. Environmental material terms are generally mass nouns. Examples include: air, soil, or water. EnvO (v 2013-06-14) terms can be found via the link: www.environmentontology.org/Browse-EnvO	EnvO	environment	{term}
env_package	environmental package	MIGS/MIMS/MIMARKS extension for reporting of measurements and observations obtained from one or more of the environments where the sample was obtained. All environmental packages listed here are further defined in separate subtables. By giving the name of the environmental package, a selection of fields can be made from the subtables and can be reported	CV	migs/mims/mimarks extension	[air|built environment|host-associated|human-associated|human-skin|human-oral|human-gut|human-vaginal|microbial mat/biofilm|misc environment|plant-associated|sediment|soil|wastewater/sludge|water]

subspecf_gen_lin	subspecific genetic lineage	This should provide further information about the genetic distinctness of this lineage by recording additional information i.e biovar, serovar, serotype, biovar, or any relevant genetic typing schemes like Group I plasmid. It can also contain alternative taxonomic information	genetic lineage below lowest rank of NCBI taxonomy, which is subspecies, e.g. serovar, biotype, ecotype	nucleic acid sequence source	{text}
ploidy	ploidy	The ploidy level of the genome (e.g. allopolyploid, haploid, diploid, triploid, tetraploid). It has implications for the downstream study of duplicated gene and regions of the genomes (and perhaps for difficulties in assembly). For terms, please select terms listed under class ploidy (PATO:001374) of Phenotypic Quality Ontology (PATO), and for a browser of PATO (v 2013-10-28) please refer to http://purl.bioontology.org/ontology/PATO	PATO	nucleic acid sequence source	{term}
num_replicons	number of replicons	Reports the number of replicons in a nuclear genome of eukaryotes, in the genome of a bacterium or archaea or the number of segments in a segmented virus. Always applied to the haploid chromosome count of a eukaryote	for eukaryotes and bacteria: chromosomes (haploid count); for viruses: segments	nucleic acid sequence source	{integer}
extrachrom_elements	extrachromosomal elements	Do plasmids exist of significant phenotypic consequence (e.g. ones that determine virulence or antibiotic resistance). Megaplasmids? Other plasmids (borrelia has 15+ plasmids)	number of extrachromosmal elements	nucleic acid sequence source	{integer}
estimated_size	estimated size	The estimated size of the genome prior to sequencing. Of particular importance in the sequencing of (eukaryotic) genome which could remain in draft form for a long or unspecified period.	number of base pairs	nucleic acid sequence source	{integer} bp
ref_biomaterial	reference for biomaterial	primary publication if isolated before genome publication; otherwise, primary genome report	PMID, DOI or URL	nucleic acid sequence source	{PMID|DOI|URL}
source_mat_id	source material identifiers	A unique identifier assigned to a material sample (as defined by http://rs.tdwg.org/dwc/terms/materialSampleID, and as opposed to a particular digital record of a material sample) used for extracting nucleic acids, and subsequent sequencing. The identifier can refer either to the original material collected or to any derived sub-samples. The INSDC qualifiers /specimen_voucher, /bio_material, or /culture_collection may or may not share the same value as the source_mat_id field. For instance, the /specimen_voucher qualifier and source_mat_id may both contain 'UAM:Herps:14' , referring to both the specimen voucher and sampled tissue with the same identifier. However, the /culture_collection qualifier may refer to a value from an initial culture (e.g. ATCC:11775) while source_mat_id would refer to an identifier from some derived culture from which the nucleic	for cultures of microorganisms: identifiers for two culture collections; for other material a unique arbitrary identifer	nucleic acid sequence source	{text}

		acids were extracted (e.g. xatc123 or ark:/2154/R2).			
Pathogenicity	known pathogenicity	To what is the entity pathogenic.	CV	nucleic acid sequence source	{term}
biotic_relationship	observed biotic relationship	Is it free-living or in a host and if the latter what type of relationship is observed	enumeration	nucleic acid sequence source	[free living|parasite|commensal|symbiont]
specific_host	specific host	If there is a host involved, please provide its taxid (or environmental if not actually isolated from the dead or alive host - i.e. pathogen could be isolated from a swipe of a bench etc) and report whether it is a laboratory or natural host). From this we can calculate any number of groupings of hosts (e.g. animal vs plant, all fish hosts, etc)	host taxid, unknown, environmental	nucleic acid sequence source	
host_spec_range	host specificity or range	The NCBI taxonomy identifier of the specific host if it is known	NCBI taxid	nucleic acid sequence source	{integer}
health_disease_stat	health or disease status of specific host at time of collection	Health or disease status of specific host at time of collection. This field accepts PATO (v 2013-10-28) terms, for a browser please see http://purl.bioontology.org/ontology/PATO	PATO	nucleic acid sequence source	{term}
trophic_level	trophic level	Trophic levels are the feeding position in a food chain. Microbes can be a range of producers (e.g. chemolithotroph)	enumeration	nucleic acid sequence source	[autotroph| carboxydotroph| chemoautotroph| chemoheterotroph| chemolithoautotroph| chemolithotroph| chemoorganoheterotroph| chemoorganotroph| chemosynthetic| chemotroph| copiotroph| diazotroph| facultative| autotroph| heterotroph| lithoautotroph| lithoheterotroph| lithotroph| methanotroph| methylotroph| mixotroph| obligate| chemoautolithotroph| oligotroph| organoheterotroph| organotroph| photoautotroph| photoheterotroph| photolithoautotroph| photolithotroph| photosynthetic| phototroph]
propagation	propagation	This field is specific to different taxa. For phages: lytic/lysogenic, for plasmids: incompatibility group (Note: there is the strong opinion to name phage propagation obligately lytic or temperate, therefore we also give this choice	for virus: lytic, lysogenic, temperate, obligately lytic; for plasmid: incompatibility group; for eukaryote: asexual, sexual) [CV	nucleic acid sequence source	{term}
encoded_traits	encoded traits	Should include key traits like antibiotic resistance or xenobiotic degradation phenotypes for plasmids, converting genes for phage	for plasmid: antibiotic resistance; for phage: converting genes	nucleic acid sequence source	{text}
rel_to_oxygen	relationship to oxygen	Is this organism an aerobe, anaerobe? Please note that aerobic and anaerobic are valid	enumeration	nucleic acid sequence source	[aerobe|anaerobe|facultative|microaerophilic|microanaerobe|obligate

		descriptors for microbial environments			aerobe|obligate anaerobe]
isol_growth_condt	isolation and growth condition	Publication reference in the form of pubmed ID (pmid), digital object identifier (doi) or url for isolation and growth condition specifications of the organism/material	PMID,DOI or URL	nucleic acid sequence source	{PMID|DOI|URL}
samp_collect_device	sample collection device or method	The method or device employed for collecting the sample	type name	nucleic acid sequence source	{text}
samp_mat_process	sample material processing	Any processing applied to the sample during or after retrieving the sample from environment. This field accepts OBI, for a browser of OBI (v 2013-10-25) terms please see http://purl.bioontology.org/ontology/OBI	text or OBI	nucleic acid sequence source	{text|term}
samp_size	amount or size of sample collected	Amount or size of sample (volume, mass or area) that was collected	measurement value	nucleic acid sequence source	{float} {unit}
nucl_acid_ext	nucleic acid extraction	Link to a literature reference, electronic resource or a standard operating procedure (SOP)	PMID, DOI or URL	sequencing	{PMID|DOI|URL}
nucl_acid_amp	nucleic acid amplification	Link to a literature reference, electronic resource or a standard operating procedure (SOP)	reference to amplification method; clean-up method	sequencing	{PMID|DOI|URL}
lib_size	library size	Total number of clones in the library prepared for the project	number of clones	sequencing	{integer}
lib_reads_seqd	library reads sequenced	Total number of clones sequenced from the library	number of reads sequenced	sequencing	{integer}
lib_const_meth	library construction method	Library construction method used for clone libraries	library construction method	sequencing	{text}
lib_vector	library vector	Cloning vector type(s) used in construction of libraries	vector	sequencing	{text}
lib_screen	library screening strategy	Specific enrichment or screening methods applied before and/or after creating clone libraries	screening strategy name	sequencing	{text}
target_gene	target gene	Targeted gene or locus name for marker gene studies	gene name	sequencing	{text}
target_subfragment	target subfragment	Name of subfragment of a gene or locus. Important to e.g. identify special regions on marker genes like V6 on 16S rRNA	gene fragment name	sequencing	{text}
pcr_primers	pcr primers	PCR primers that were used to amplify the sequence of the targeted gene, locus or subfragment. This field should contain all the primers used for a single PCR reaction if multiple forward or reverse primers are present in a single PCR reaction. The primer sequence should be reported in uppercase letters	FWD: forward primer sequence REV:reverse primer sequence	sequencing	FWD:{dna} REV:{dna}
mid	multiplex identifiers	Molecular barcodes, called Multiplex Identifiers (MIDs), that are used to specifically tag unique samples in a sequencing run. Sequence should be reported in uppercase letters	multiplex identifier sequence	sequencing	{dna}
adapters	adapters	Adapters provide priming sequences for both amplification and sequencing of the sample-library fragments. Both adapters should be reported; in uppercase letters	adapter A and B sequence	sequencing	{dna},{dna}
pcr_cond	pcr conditions	Description of reaction conditions and components for PCR in the form of 'initial	initial denaturation:degrees_minutes; annealing:degrees	sequencing	initial denaturation:degrees_minutes; annealing:degrees_min

		denaturation:94degC_1.5min; annealing=...'	_minutes; elongation: degrees_minutes; final elongation:degrees_minutes; total cycles		utes; elongation: degrees_minutes; final elongation:degrees_minutes; total cycles
seq_meth	sequencing method	Sequencing method used; e.g. Sanger, pyrosequencing, ABI-solid		sequencing	{text}
seq_quality_check	sequence quality check	Indicate if the sequence has been called by automatic systems (none) or undergone a manual editing procedure (e.g. by inspecting the raw data or chromatograms). Applied only for sequences that are not submitted to SRA,ENA or DRA	none or manually edited	sequencing	[none|manually edited]
chimera_check	chimera check	A chimeric sequence, or chimera for short, is a sequence comprised of two or more phylogenetically distinct parent sequences. Chimeras are usually PCR artifacts thought to occur when a prematurely terminated amplicon reanneals to a foreign DNA strand and is copied to completion in the following PCR cycles. The point at which the chimeric sequence changes from one parent to the next is called the breakpoint or conversion point	name and version of software	sequencing	{text} {text}
assembly	assembly	How was the assembly done (e.g. with a text based assembler like phrap or a flowgram assembler); estimated error rate associated with the finished sequences (e.g. error rate of 1 in 1000 bp); and the method of calculation	assembly method; estimated error rate; method of calculation	sequencing	{text};{text};{text}
assembly_name	assembly name	Name/version of the assembly provided by the submitter that is used in the genome browsers and in the community	name and version of assemby	sequencing	{text} {text}
finishing_strategy	finishing strategy	Was the genome project intended to produce a complete or draft genome, Coverage, the fold coverage of the sequencing expressed as 2x, 3x, 18x etc, and how many contigs were produced for the genome	status; coverage; number of contigs	sequencing	[complete|draft];{integer};{integer}
annot_source	annotation source	For cases where annotation was provided by a community jamboree or model organism database rather than by a specific submitter	annotation source description	sequencing	{text}
url	relevant electronic resources		URL	sequencing	{URL}
sop	relevant standard operating procedures	Standard operating procedures used in assembly and/or annotation of genomes, metagenomes or environmental sequences	reference to SOP	sequencing	{PMID|DOI|URL}

Annex III – Examples of XACML rules

XACML policy containing two rules. The first rule defines that a physician can access to genomic information provided that an e-mail is sent to the patient. The second one defines that access to chromosome 20 inside a genomic information file is only permitted to the main researcher of a research project and that an e-mail has to be sent when it happens.

```
<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:isdcm: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">
                            examples/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">
                            examples/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:lab6: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>
    </Rule>
</Policy>
```

Annex IV - Authorizing access to genomic information based on XACML rules

This section presents an example of use of XACML rules for authorizing operations over SAM files. It consists of an application prototype of an XACML driven authorization for genomic data manipulation.

The application is shipped with an Open Source implementation of XACML called Balana which can be found in [30], which allows us to check access control to genomic files.

The application is a proof of concept and consists of a "privacy layer" that checks if the user requesting a read or manipulate operation over a SAM file is able to do it according to the rules defined. In order to use the application, it is required to install Samtools [31].

Figure 13 shows the flow of the application. The user sends an Evaluate request to our application, which checks if there is any policy authorizing the user. If it permits the operation, then the SAM file is accessed and the requested information is given to the user. If not, then the application notifies the user that the operation is denied.

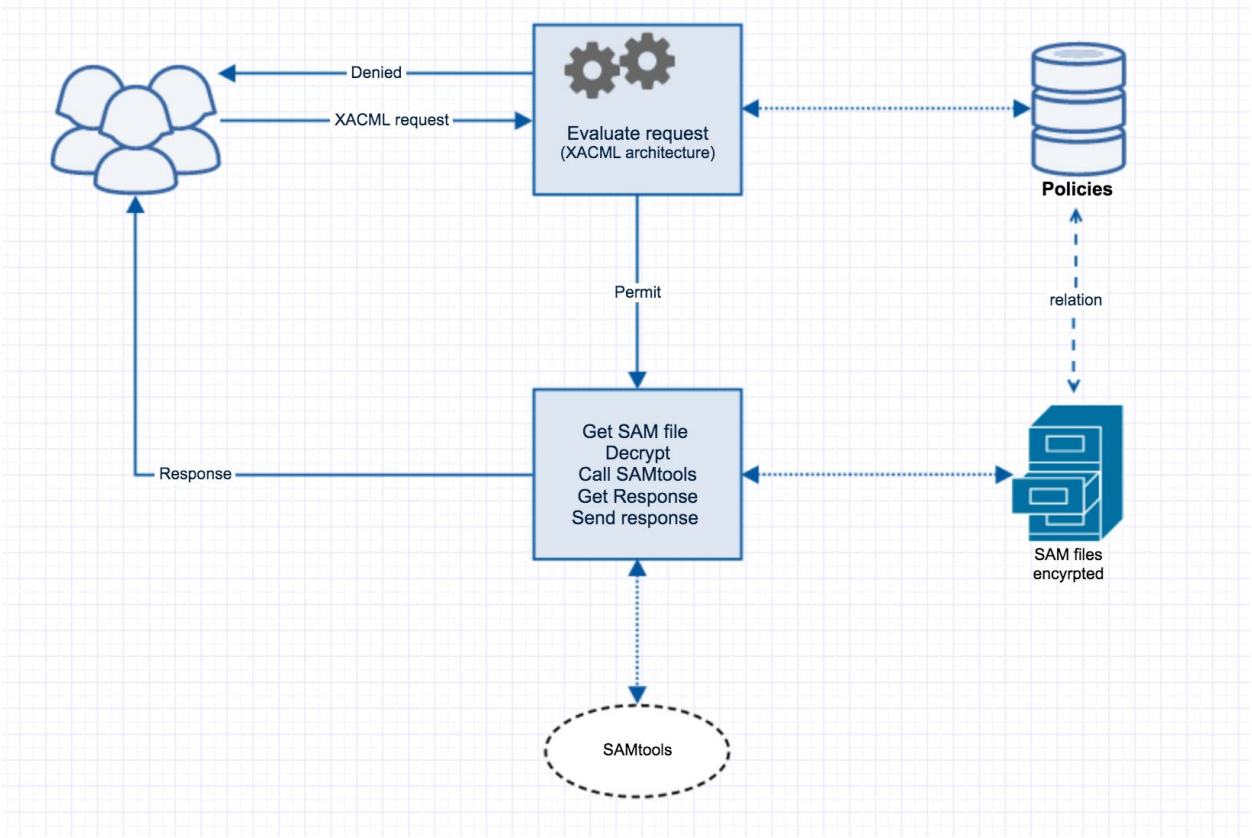


Figure 13 – Authorization flow

For the moment, there are two operations allowed:
1) View a sam file. It generates a .bam file.
2) View a chromosome of a sam file. It generates a .bam file, creating the index file (BAI) and then it extracts the data for the specific region requested.

To check the operation, the following rules have been defined:
1) 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.
2) 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.

The XACML policy used can be found here in Annex III of this document. Alternatively, it can be found in [32]. The complete code of the application can be found in [33]. To try out the script run: java -jar privacy.jar

The file used in this example, toy.sam, provided by Samtools, can be downloaded from: https://github.com/saracubillas/samtools/blob/develop/privacy/examples/toy.sam

Figure 14 below shows two different scenarios where the request is authorized:

```
Enter User name : John Doe
Enter role : physician

You can select one of following action:

[1] View - Convert SAM to BAM   [2] View Chromosome - extract the data for specific region
Enter action Id : 1
Enter file path : /Users/sara/Documents/MIRI/thesis/samtools-1.3.1/examples/toy.sam
```

```
Enter User name : JaneDoe
Enter role : researcher

You can select one of following action:

[1] View - Convert SAM to BAM   [2] View Chromosome - extract the data for specific region
Enter action Id : 2
Enter file path : /Users/sara/Documents/MIRI/thesis/samtools-1.3.1/examples/toy.sam
Enter chromosome reference : ref2
```

Figure 14 – Example of execution

The XACML model supports and encourages the separation of the access decision from the point of use. Policy Enforcement Points, abbreviated as PEPs, are the endpoints in the XACML Reference Architecture where authorization questions are formulated and their resulting decisions are enforced. An authorization question comprises of attributes grouped into subject, resource, action, environment and possibly more custom categories, and the corresponding values for each such attribute.

The image below shows how the user will enter the data needed to create the XACML request in the sample application:
- a subject
- an action
- a resource
- a role

The code to evaluate the request using Balana and show the result to the user is shown in Figure

```
private static String evaluateRequest(String request) {
    Balana balana = new Balana();
    String response = balana.evaluateRequest(request);
    return XACMLparser.getDecision(response);
}
```

15.

Figure 15 – Construct and show evaluation request

The Policy Decision Point (PDP), the component in the XACML architecture responsible for processing the access control requests from the PEP, responds to such a question with a single

```
if (decision.equals("Permit")){
    System.out.println("\n" + userName + " is authorized to perform this action\n\n");
    executeAction(resource, actionName);
} else {
    System.out.println("\n" + userName + " is NOT authorized to perform this action\n");
}
```

decision, which typically is a Permit or a Deny. Upon receiving the response, if the decision is Permit an action it is executed.

The code snippet to construct the samtools command after authorization is shown in Figure 16.

```java
package application.service;

public class ViewChromosome extends Command {

    @Override
    public void execute(String file) {
        String fileWithoutRef = file.substring(0, file.lastIndexOf('#'));
        String fileWithoutExtn = fileWithoutRef.substring(0, fileWithoutRef.lastIndexOf('.'));
        String chromosomeRef = file.substring(file.lastIndexOf("#") + 1);

        viewSAM(fileWithoutRef);
        IndexBam(fileWithoutExtn);
        String command = "samtools view -h " + fileWithoutExtn + ".bam " + chromosomeRef;

        String output = executeCommand(command);
        System.out.println(output);
        System.out.println("\n========OUTPUT======");
    }

    private void IndexBam(String fileWithoutExtn) {
        Command Index = new Index();
        Index.execute(fileWithoutExtn + ".bam");
    }

    private void viewSAM(String file) {
        Command ViewSAM = new ViewSAM();
        ViewSAM.execute(file);
    }
}
```

Figure 16 – Create samtools command

Annex V – GENIFF generator

In the current version of this tool, the user specifies in an xml file the data to be encoded, as shown in the following XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
        <study>
                <dataset>
                        <stream type=”BAM”>
                                <source>File path for input</source>
                                <encryption>Partial</encryption>
                        </stream>
                </dataset>
                <dataset>
                        <stream type=”BAM”>
                                <source>File path for input</source>
                        </stream>
                </dataset>
        </study>
<root>
```

By providing as parameters “encode” the path of the output file and the path of this xml file, the tool will generate the defined encoding. As an example, currently the “decode” method will recreate the input files at their original location (and appending “new” to the file name). An example of both commands can be found in Figure 17.

```
gencom@gencom-VirtualBox:~$ ./writingGenISOBMFF encode  test_output.geniff config.xml
gencom@gencom-VirtualBox:~$ ./writingGenISOBMFF decode test_output.geniff
```

Figure 17 – Command line commands for encoding and decoding GENIFF files

Figure 18 shows the relationship between the boxes defined in GENIFF and the C++ classes that form part of the GENIFF generator application.

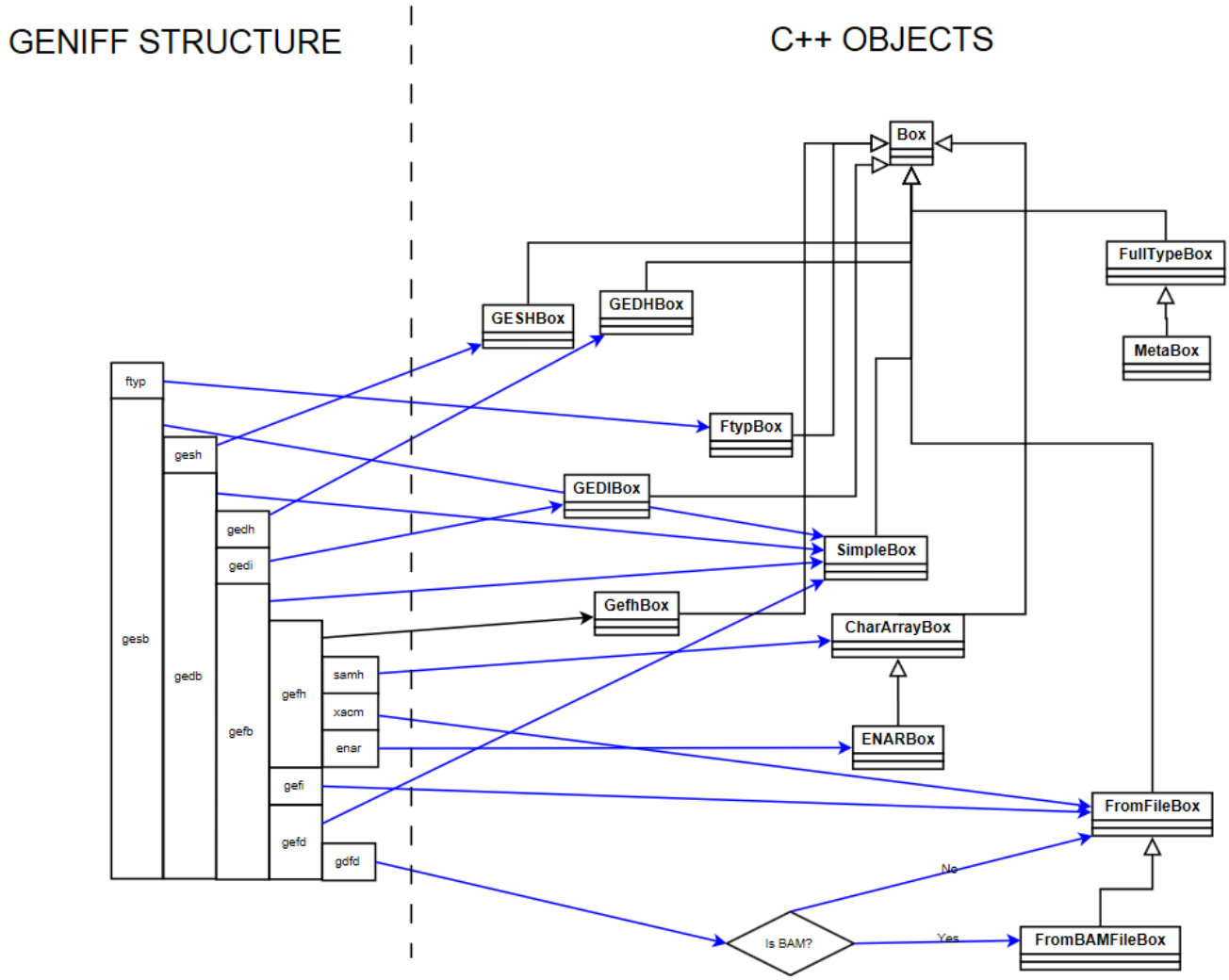


Figure 18 – Relationship between C++ classes and GENIFF structure

Annex VI – Integration of the two tools

By integrating the tools described in Section 8 and Annexes IV and V, we take advantage of both XACML and GENIFF to give a privacy aware access to the resources. GENIFF provides a way to store the XACML definitions within the dataset. By extracting this region of the file, we have enough information to evaluate the rules and decide if the request should be granted or not.

We extend our previous software to perform this task. The first step is to add a parameter option to the GENIFF tool in order to specify which element of the dataset should be extracted. In this case, this specific element is the XACML file.

We show in Figure 19 a placeholder request to obtain the policy contents from a GENIFF file.

```
private static void initBalana() {
    // using file based policy repository. so set the policy location as system property
    String policyLocation = null;
    String configPath = null;
    try {
       // policyLocation = getPolicyFromGniff(resourceName);
```

Figure 19 – Extracting XACML rules from GENIFF file

As previously explained, the rules are evaluated through the use of the Balana library. If the response is positive, then the original file is extracted from the GENIFF container, and decrypted if necessary, allowing further access through another tool such as samtools, as we have seen before. The specific command which is issued is the one for which we evaluated the policy as the first step (see Figure 20).

```
        if (decision.equals("Permit")){
            System.out.println("\n" + userName + " is authorized to perform this action\n\n");
            //resource = decode(resource);
            executeAction(resource, actionName);
        } else {
            System.out.println("\n" + userName + " is NOT authorized to perform this action\n");
        }

    }
```

Figure 20 – Show evaluation result

Finally, figure 21 shows a revised authorization flow.

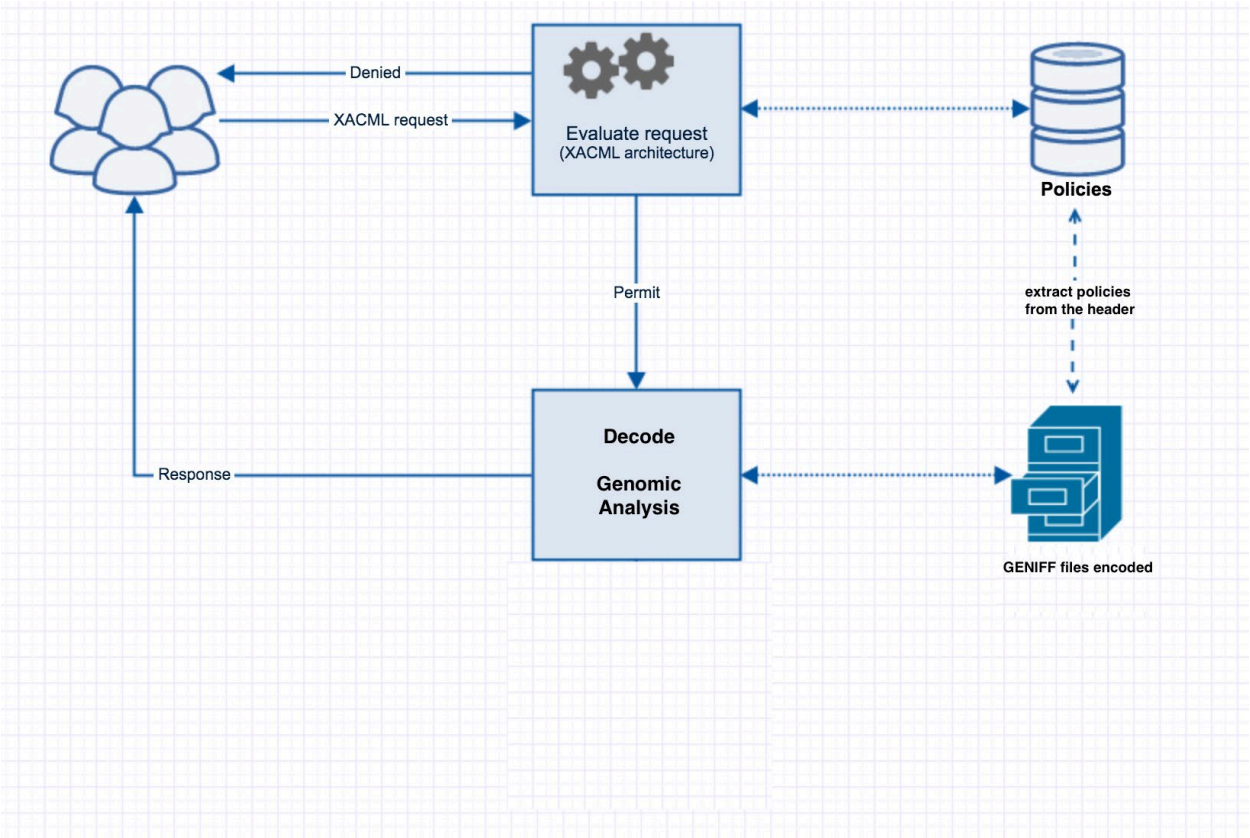


Figure 21 – Revised authorization flow

Thanks to this integration we can now take full advantage of GENIFF's features of placing together data and access rules.