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

ISO/IEC JTC 1/SC 29/WG 11
MPEG2017/m39940
January 2017, Geneva, Switzerland

Source:	**DMAG-UPC, CNAG-CRG, BSC, Made of Genes**
Status:	**Proposal**
Title:	**GENIFF v2**
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)**

Index

1 Introduction

This document describes the changes of GENIFF, from its original version, to version v2. The changes have been introduced in preparation for the Core Experiment CE 4 Genomic Access Abstract Layer described in [1].

GENIFF v2 is proposed by several organizations, including 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), and the Spanish company Made of Genes.

2 Structure of the document

This document revises some of the features of the GENIFF format presented in [2] in order to provide better results for CE 4. Section 3 proposes an improvement of the format for representing genomic information. Next, section 4 describes how the proposed format is able to handle current genomic information following FASTQ or SAM/BAM, among others.

Moreover, the document revises the Metadata (section 5) and Privacy (section 6) aspects of the initial proposal.

The development of the core experiment itself is described in another document [3].

3 GENIFF, the GENomic Information File Format

3.1 Introduction

In order to define a file format for the storage and transmission of genomic information, we follow the structure Type, Length and Value (TLV) for encoding the information.

The given name for this file format is GENomic Information File Format (GENIFF).

In the rest of the section, we briefly describe the features of the format and its formal representation.

3.2 GENIFF description

In order to be able to define the required fields for genomic information, we need to define a formal representation for them. The selected representation is TLV (Type, Length, Value), as we want to represent fields without a predefined length.

type	length	value
8 bytes	8 bytes	Variable length

Figure 1 – Type, length and value fields and corresponding length

The definition of such a field is:

```
class gen_info
{
      char type [8];
      uint64_t length;   // Search for a 64 bytes length integer value
      uint8_t value [];  // Undetermined length value
}
```

The type field consists of 8 bytes indicating the name of the field using ASCII characters. The length field is an integer value of 64 bits representing the size of the whole `gen_info` field in bytes. The value field contains the data and its size is the total size indicated in the length value less 16 bytes corresponding bytes used by the type and length fields.

The value will be interpreted according to the type: the byte array is defined for each, as a concatenation of native types (fixed length fields), characters sequences (with the length prepended), or other fields defined with the same TLV notation. See specification for each type for further details.

The information contained inside a GENIFF file follows a hierarchical structure, going from the more general concept of a genomic study, to the most basic concept of the raw genomic information. 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, to represent the genomic information, we use concepts like *stream* as defined in CARGO [4]. The objects hierarchy and relationship are defined in Table 1.

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

A complete set of `genrecor` fields storing the data of predefined genomic record type composes a *genomic dataset data* (`gendatas`, for genomic dataset). A dataset can represent the genomic file of type FASTA, FASTQ, SAM, VCF, etc.

`gendatas` field is accompanied by optional *dataset meta-information* (`datahead`) (representing optional genomic file-header and also including security and privacy information) and optional *dataset index* (`dataindx`). 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 `gendatas` dataset fields can be included into *genomic study* (`genstudy`, for genomic study). Similarly, as `gendatas` field, `genstudy` is accompanied by optional *meta-information field* (`studyhea`, for genomic study header), also including security and privacy information.

It is worth noting that genomic information does not have a temporal order, 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.

3.3 Proposal of field types and structure

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

Table 1 contains the name of the fields, 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 3.4 for a complete description).

Table 1 – Field types and structure

Name					**Repeat**	**Require**	**Description**
filetype					NO	YES	File type and compatibility.
genstudy					To be discussed	YES	Container for all the genomic data - the single study field.
	studyhea				NO	NO	Genomic field header for the study field. It contains metadata, including privacy and security information.
	gendatas				YES	NO	Container for a single genomic dataset equivalent to file of type e.g. FASTA, FASTQ, SAM, VCF, etc.
		datahead			NO	NO	Genomic dataset header, overall information. It contains metadata, including privacy and security information.
		dataindx			NO	NO	Indexing information for fast dataset access.
		genrecor			YES	NO	Genomic records' field field based on stream field.
			recohead		NO	NO	Genomic field header, overall information. It contains metadata, including privacy and security information.
			recoindx		NO	NO	Genomic information indexing information for direct access.
			genofile		NO	YES	Data information field including buffers, or file pointers on how data is stored internally.
				extefile	NO	NO	Data reference field, pointing directly to file offset(s) of external file.
				intefile	NO	NO	Direct field data or information about data offsets inside the current file.

It is worth noting that the structure proposed for fields is rather flexible, as it should support different types of genomic information. To show how genomic information files are supported, we give some examples in Section 4.

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 fields. 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. Table 2 shows the correspondence with the names defined in GENIFF v1 [2] for further reference.

Table 2 – Field names equivalence

GENIFF v1	GENIFF v2
ftyp	filetype
gesb	genstudy
gesh	studyhea
gedb	gendatas
gedh	datahead
gedi	dataindx
gefb	genrecor
gefh	recohead
gefi	recoindx
gefd	genofile
gdff	extefile
gdfd	intefile

3.4 Fields description

This section describes the fields defined in Table 1. They are as follows:

- `filetype` field defines the type of file it contains according to the genomic information supportd.
- `genstudy` contains all fields for a specific study.
- `studyhea` 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.
- `gendatas` container for a single genomic dataset.
- `gendatas/datahead` 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.
- `gendatas/dataindx` indexing information specific for the genomic dataset.
- `genrecor` container for one stream of data (e.g. uncompressed FASTA, FASTQ, SAM, BAM, VCF, …)
- `genrecor/recohead` 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.

- `genrecor/recoindx` indexing information for the specific stream.
- `genrecor/recofile` information on how the particular stream is stored.
- `genrecor/recofile/extefile` data reference field which points to an external file location storing the stream information.
- `genrecor/recofile/intefile` actual data container of the file storing the relevant information.

Privacy rules defined using XACML should be included into an `xml` field contained in the `studyhea`, `datahea` and `recohead` fields, depending on which level of information they apply to, the whole study (inside `genstudy` field), a specific dataset (inside `genrecor` field) or for a stream (inside `genrecor` field).

3.5 Proposal of Application Programming Interface (API) for managing GENIFF fields

In order to facilitate management of the GENIFF format and the fields it contains, we propose the definition of an Application Programming Interface (API), which could be implemented locally or remotely.

The API should provide access to the different levels provided by the GENIFF format. Table 3 shows a brief description of the operations to be supported.

Table 3 – API methods

Level	Method name	Brief description
Complete file (GENIFF)	getStudyHeader	Returns the contents of the genomic study header.
Complete file (GENIFF)	setStudyHeader	Updates the contents of the genomic study header.
Complete file (GENIFF)	getStudyDataset	Returns the contents of a specific dataset inside the study.
Complete file (GENIFF)	setStudyDataset	Updates the contents of a specific dataset inside the study.
Complete file (GENIFF)	getStudyHeaderField	Returns the contents of a field inside genomic study header.
Complete file (GENIFF)	setStudyHeaderField	Updates the contents of a field inside genomic study header.
Dataset	getDatasetHeader	Returns the contents of the genomic dataset header.
Dataset	setDatasetHeader	Updates the contents of the genomic dataset header.
Dataset	getDatasetHeaderField	Returns the contents of a field inside genomic dataset header.
Dataset	setDatasetHeaderField	Updates the contents of a field inside genomic dataset header.
Dataset	getDatasetIndex	Returns the contents of the genomic dataset index.
Dataset	setDatasetIndex	Updates the contents of the genomic dataset index.
Dataset	getDatasetRecord	Returns the contents of a specific record inside the dataset.
Dataset	setDatasetRecord	Updates the contents of a specific record inside the dataset.
Record	getRecordHeader	Returns the contents of the genomic record header.
Record	setRecordHeader	Updates the contents of the genomic record header.
Record	getRecordHeaderField	Returns the contents of a field genomic record header.
Record	setRecordHeaderField	Updates the contents of a field genomic record header.
Record	getRecordIndex	Returns the contents of the genomic record index.

Record	setRecordIndex	Updates the contents of the genomic record index.
Record	getRecordValueInt	Returns the value of a genomic record stored internally.
Record	setRecordValueInt	Updates the value of a genomic record stored internally.
Record	getRecordValueExt	Returns the value of the genomic record stored externally.
Record	setRecordValueExt	Updates the value of the genomic record stored externally.

4 Representation and storage of existing genomic data with GENIFF

4.1 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.

4.2 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 fields should contain.

4.2.1 Representation of FASTQ

4.2.1.1 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 (`studyhea`) or at the dataset level (`datahead`). In case there is some indexing information for the FASTQ information, it can be included in the `recoindx` or `dataindx` (pointing to `recoindx`) fields.

Field					Value
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the FASTQ file.
		datahead			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.
			xacmrule		XACML rules and protection
		dataindx			FASTQ index, if one is decided.
		genrecor			Genomic record field.
			recohead		Stream specific header.
			recoindx		FASTQ index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.

4.2.1.2 *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 (`studyhea`), at the dataset level (`datahead`) or at the separate stream level (`recohead`). 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 `recoindx` field (per each stream) or on the dataset level (`dataindx`).

Field					Value
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the FASTQ file.
		datahead			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.
			xacmrule		XACML rules and protection
		dataindx			Streams index, if one is decided.
		genrecor			Genomic record field #1 used to store read identifiers data.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #2 used to store sequence data.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #3 used to store quality data.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.

4.2.2 Representation of SAM

This section shows how to represent SAM files inside GENIFF. Regarding SAM header information, we defined an XML schema in [2] to facilitate its processing.

4.2.2.1 *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 (`studyhea`) or at the dataset level (`datahead`). In case there is some indexing information for the SAM file, it can be included in the `dataindx` field.

Field					**Value**
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the SAM file.
		datahead			Study composed of a unique stream (SAM type). Metadata, including privacy and security information.
			samheade		SAM Header modeled in XML.
			xacmrule		XACML rules and protection.
		dataindx			SAM index, if one is decided.
		genrecor			Genomic record field.
			recohead		Stream specific header.
			recoindx		Raw SAM index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream (SAM file without the header (records only), which will be re-generated from stored XML -- if needed)

4.2.2.2 *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 [5] [6]). To control access to the dataset, privacy rules and security information can be included at the study level (`studyhea`), at the dataset level (`datahead`) or at the separate stream level (`recohead`). In case there is some indexing information for the SAM file, it can be included in the `recoindx` and `dataindx` fields.

Field					**Value**
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the SAM file.
		datahead			Study composed of a unique stream - SAM file. Metadata, including privacy and security information.
			samheade		SAM Header modeled in XML.
			xacmrule		XACML rules and protection.

		dataindx			Global SAM index, if one is decided, linked to the stream-level indices.
		genrecor			Genomic record field #1 used to store alignment query template name data - SAM field QNAME.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #2 used to store alignment flags data - SAM field FLAG.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
…					
		genrecor			Genomic record field #9 used to store sequence data – SAM field SEQ.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #10 used to store quality scores data - SAM field QUAL.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #11 used to store alignment optional fields data.
			recohead		Stream specific header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.

4.3 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 fields 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.

4.3.1 Compressed representation of FASTQ

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

4.3.1.1 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 `intefile` field. Such approach would apply to FASTQ being compressed by a method that produces a single file as output.

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 (`studyhea`) or at the dataset level (`datahead`).

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 `recoindx` or `dataindx` (pointing to `recoindx`) fields. The details on indexing information will be updated accordingly.

Field					Value
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the FASTQ file.
		datahead			Study composed of a unique stream (FASTQ type – together with compression used). Metadata, including privacy and security information.
			xacmrule		XACML rules and protection.
		dataindx			FASTQ index pointer to recoindx field.
		genrecor			Genomic record field.
			recohead		Stream specific header.
			recoindx		FASTQ index as defined by the MPEG Committee.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.

4.3.1.2 *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 `intefile` field.

Field					Value
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the FASTQ file.
		datahead			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)

			xacmrule		XACML rules and protection
		dataindx			Streams index, if one is decided.
		genrecor			Genomic record field #1 used to store read identifiers data.
			recohead		Stream specific header
			recoindx		File pointer to how data is stored internally.
			genofile		Byte array of the current stream.
		genrecor			Genomic record field #2 used to store compressed sequence data.
			recohead		Stream specific header
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #3 used to store compressed quality data.
			recohead		Stream specific header
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.

4.3.2 Compressed representation of SAM

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

4.3.2.1 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 (`studyhea`), at the container level (`datahead`) or at the file (stream) level (`recohead`). The `recoindx` field 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 `recohead` field, following the XML schema defined for SAM header fields in [2].

Field					Value
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the BAM file.
		datahead			Study composed of a unique stream (BAM type – together with compression information). Metadata, including privacy and security information.
			samheade		SAM Header modeled using XML.
			xacmrule		XACML rules and protection.

			enarbloc		Bit array indicating which blocks of the data stored in intefile are encrypted.
		dataindx			Indexing information for unique stream is present at genstudy.gendatas[0].genrecor[0]. This is used to emulate BAI behavior.
		genrecor			Genomic record field.
			recohead		Stream specific header.
			recoindx		Currently, BAI index. Later on, the SAM index as defined by the MPEG Committee.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream

4.3.2.2 A SAM file decomposed into BAM streams inside GENIFF

In order to simplify access to different sub entities within the SAM file, we divide its content in multiple files, each one compressed with BAM and self-contained. In this example, we show the structure for such a file containing one dataset that corresponds to the original dataset, but composed of multiple BAM-encoded streams (for example, one stream per reference sequence). This allows, for example, a first step into random access, or the ability to stream the well-defined regions.

Field					**Value**
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the BAM file.
		datahead			Study encoded in multiple BAM streams (BAM multistream type). Metadata, including privacy and security information.
			samheade		SAM Header modeled using XML.
			xacmrule		XACML rules and protection.
		dataindx			Left empty
		genrecor			Genomic record field #1 (e.g. stores data for Chromosome 1)
			recohead		Specific stream header. Information such as stream name, salt used if encrypted
			recoindx		Stream index. (BAI format)
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #2 (e.g. stores data for Chromosome 2)
			recohead		Specific stream header. Information such as stream name, salt used if encrypted
			recoindx		Stream index. (BAI format)
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #3 (e.g. stores data for Chromosome 3)

			recohead		Specific stream header. Specific stream header. Information such as stream name, salt used if encrypted
			recoindx		Stream index. (BAI format)
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #4 (e.g. stores data for Chromosome 4)
			recohead		Specific stream header. Specific stream header. Information such as stream name, salt used if encrypted
			recoindx		Stream index. (BAI format)
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #5 (e.g. stores data for Chromosome 5)
			recohead		Specific stream header. Specific stream header. Information such as stream name, salt used if encrypted
			recoindx		Stream index. (BAI format)
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.

4.3.2.3 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 (`studyhea`), at the container level (`datahead`) or at the file (stream) level (`recohead`). The `recoindx` field 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 `intefile` field. In this way, a different codec can be used to compress each stream.

Field					**Value**
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for the study.
	studyhea				The header indicates that the information is for study X.Y. Privacy rules at study level. Other metadata relevant for the study.
	gendatas				Container for the BAM file.
		datahead			Study composed of a unique stream (BAM type). Metadata, including privacy and security information.
		dataindx			Indexing information for unique streams is present at each sub-index field at genstudy.gendatas[0].genrecor[i]. This is used to emulate BAI behavior.
		genrecor			Genomic record field #1 used to store alignment query template name data – SAM field QNAME
			recohead		Specific stream header.
			recoindx		Stream index.

			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #2 used to store alignment flags data - SAM field FLAG
			recohead		Specific stream header.
			recoindx		Stream index.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
...					
		genrecor			Genomic record field #9 used to store sequence data - SAM field SEQ
			recohead		Specific stream header.
			recoindx		Stream index, if one is decided.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #10 used to store quality scores data - SAM field QUAL.
			recohead		Specific stream header.
			recoindx		Stream index.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.
		genrecor			Genomic record field #11 used to store alignment optional fields data.
			recohead		Specific stream header.
			recoindx		Stream index.
			genofile		File pointer to how data is stored internally.
				intefile	Byte array of the current stream.

4.3.3 Genomic information represented using CARGO

In this example, genomic information represented using CARGO is stored inside GENIFF. CARGO uses three different types of blocks:
1) container blocks
2) stream blocks
3) superblocks

The relation between different types of blocks is as follows. The genomic record data, after information decoupling and optional data transformations, is stored in streams. Each stream stores data of specific type, e.g. sequences can be stored in one stream, quality values in another. The data inside streams is split into separate blocks of configurable size, called *stream blocks*. The stream blocks are further stored in *container blocks,* which can be also of configurable size. Therefore, less than one or multiple stream blocks can be stored inside one container block. Additionally, information about stream blocks from different streams can be compacted into *superblocks* (and of configurable size), which define independent portions of input genomic data. This way, each superblock can contain data from N genomic records and can be compressed or decompressed in parallel to provide efficient data streaming.

The representation of this information inside GENIFF is presented below. Some new fields are added to the GENIFF format in order to support CARGO data structures.

Field					Value
filetype					gnif (genomic information file format), version 1.0
genstudy					Container for all the genomic data. One CARGO container is stored per study. A CARGO container as a set of 3 files including: - *.cargo-meta – container meta information - *.cargo-dataset – information about stored datasets - *.cargo-streams – streams data composing datasets
	studyhea				CARGO container header stored as container meta information area (*.cargo-meta file). Contains essential container information and its configuration (e.g., container blocks configuration and their sizes). It may also include information about security access restrictions and privacy measures at a global container scope.
	gendlist				Contains information about the dataset(s) residing in the container. Information is stored in the container dataset area (*.cargo-dataset file). The information about each dataset is stored in *.cargo-dataset file, one entry per each dataset.
	gendatas				Genomic dataset.
		datahead			Genomic dataset header. Contains information about the dataset, its genomic data type, of which streams it is composed and which compression methods are used. It may also include information about security access restrictions and privacy measures to define restricted access to: - streams or their parts - aligned regions (by additionally limiting the index for querying)
		dataindx			Dataset index for fast dataset access and query capabilities. Indexing information are generated by user-specified key-generation function during dataset storing/compression – this can be chromosome:position (as in SAM) or any, arbitrary one. Possible configurable granularity of indexing (e.g. calculating and storing index per input block of N records).
		datapart			Genomic independent data parts descriptors. They correspond to CARGO superblocks. Defines boundaries of independent parts of genomic dataset (defined as collections of stream blocks), for better access granularity. This is an auxiliary information that can help in indexing and parallel processing of the genomic data.
			parhead		Genomic independent data part header. It may contain auxiliary information to help with indexing (e. g. it stores the key).
		genrecor			CARGO stream. The information inside stream is stored as a collection of stream blocks.
			recohead		Stream header. It may also include information about restricted access and encryption technique used.
			recoindx		No additional sub-index per stream. All indexing is stored in dataindx.

					To improve the granularity of indexing and data access, it can be implemented here.
			sblk		Stream blocks composing the stream
				sbhead	Stream block header. Contains essential block information as its size. Possible configuration of its size to adapt for required data processing workloads (also associated with used compression method per stream). Can contain information (own copy, duplicated) about used compression and encryption methods.
				sbdata	Stream block binary content (compressed and encrypted).

5 Metadata associated to genomic information

5.1 Background

The GENIFF proposal [2] already introduced a deep analysis of metadata to be included in a Genome Information File Format. That document presented several alternatives, examples and analysis of options.
The next step is to propose specific metadata elements and how to manage them in the format. The approach we have chosen is to start from something rather basic and then improve it gradually. This section 5 introduces a first proposal to be discussed.

5.2 Introduction

We start with the fact that there are already different usages for the genomic data, and each one requires a different set of description values. We want to define a structure that is adaptable to different requirements. In order to do so, we propose a structure, specified in XML, where the elements are meant to be redefined (or, rather, "specialized") as we uncover new needs. We also want to ensure a certain structure to be maintained across the different versions of metadata. In order to achieve this, we propose here a metadata format leveraging on the polymorphism offered by XSD, i.e. it is possible to add new fields where desired, but ensuring that the new definition is compatible with the previous ones.

However, the presented approach requires further analysis, since other alternatives could be also feasible. Examples of alternative approaches could be to duplicate values if needed for datasets and studies, or to "aggregate" information from datasets in order to get values for studies.

5.3 Study metadata

We first describe the metadata needed for a GENIFF study. Certain fields are mandatory, such as the Title, or the Type of study, while other fields can be added in order to adapt to the needs using the previously mentioned polymorphism. Table 1 presents a study metadata box with what might be the minimal set of description fields.

Table 1: Base study's metadata

Field name	Field type	Mandatory
Title	String	Yes
Type	Controlled vocabulary	Yes
Abstract	String	Yes
Project centers	List of type project center	Yes
Description	String	No
Samples	List of type sample	Yes

If a given use case requires a description for the experiments, the previous metadata schema can be extended to what is presented in Table 2.

Table 2: Study's metadata extended with experiments

Field name	Field type	Mandatory
Title	String	Yes
Type	Controlled vocabulary	Yes
Abstract	String	Yes

Project centers	List of type project center	Yes
Description	String	No
Samples	List of type sample	Yes
Experiments	List of type experiment	Yes

From a technical point of view, the metadata shown in Table 2 would be defined in an XSD schema extending the description of Table 1's schema.

As we can see from Table 1 and Table 2, certain fields can be described with basic types, but other field types require more complex descriptions, such as the type sample. As with the entire metadata, the requirements of these fields depend on the usages. If we take the example of blood sample with only a barcode as identifier, we can define a basic sample metadata, as in Table 3.

Table 3: Base sample's metadata

Field name	Field type	Mandatory
Id	URI	Yes

However, in the context of the Genome Wide Association Study, we will require extra parameters. In the most basic case, it might be enough with an indication whether the sample belongs to the case or the control (see Table 4).

Table 4: Sample's metadata extended

Field name	Field type	Mandatory
Id	URI	Yes
Is_Case	Boolean	Yes

Using these extensions mechanisms, the requirements specified within the Minimum Information about any (x) Sequence (MIxS) could be integrated at will, although it is now straightforward to define a new extension with fields such as the *lat_lon*, the experimental_factor, or the *env_biome*, among many other.

5.4 Dataset metadata

The dataset metadata is meant to overwrite those fields whose values differ from the ones indicated at the study level (i.e., the new value is a specialization of the value at the study level). For example, we might have datasets for patient A, B and C, therefore we set in the study's metadata a list of samples representing A, B and C, but each dataset corrects ("specializes") the value to the relevant value (either A, B or C). Therefore, the base set of fields in the dataset metadata is the same as for the study, but the fields are not mandatory, as per default we consider them equal to the values indicated in the study (see Table 5).

Table 5: Base dataset's metadata

Field name	Field type	Mandatory
Title	String	No
Type	Controlled vocabulary	No
Abstract	String	No
Project centers	List of type project center	No
Description	String	No
Samples	List of type sample	No

As in the case of the Study, the definition might be extended to include new sections if required. For example, in order to enable better privacy control over the data, one such field could be a map indicating for each stream to which section of the genome's compression it participates, as in Table 6.

Table 6: Dataset's metadata extended with stream to region information

Field name	Field type	Mandatory
Title	String	No
Type	Controlled vocabulary	No
Abstract	String	No
Project centers	List of type project center	No
Description	String	No
Samples	List of type sample	No
StreamToRegion	Map GENIFF stream identifier to DNA region identifier	Yes

5.5 Examples

5.5.1 Example 1

Let us imagine that we are in the case of a study with only one dataset. Then, the metadata would be that of Table 7.

Table 7: Example 1, study metadata

Field name	Field value
Title	Example study 1
Type	Whole Genome Sequencing
Abstract	An example GENIFF study
Project centers	UPC
Samples	Id Patient A

As all values are shared between the study and the dataset, the dataset's metadata is left empty.

5.5.2 Example 2

Let us imagine that we are in the case of a study with two datasets, one case and one control. Then, the metadata would be that of Table 8.

Table 8: Example 2, study metadata

Field name	Field value
Title	Example study 2
Type	Metagenomics
Abstract	Another example GENIFF study
Project centers	UPC
Samples	Id Patient A IsCase False

	Id Patient B IsCase True

The first dataset would then correct the information with the following metadata:

Field name	**Field value**
Type	Whole Genome Sequencing
Samples	Id Patient A IsCase False

And the second with:

Field name	**Field value**
Type	Whole Genome Sequencing
Samples	Id Patient B IsCase True

6 Security and Privacy for Genomic information

This section revises the security and privacy elements applying to GENIFF. The privacy elements and use cases presented in [2] still apply in this document.

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 GENIFF fields marked as required or containing a child which is allowed to be read should remain non-encrypted. In addition, metadata should be protected by means of encryption and/or signature. See [7] for an analysis on how digital signature can be applied to genomic information headers.

The content of the `intefile` GENIFF field is the most relevant for encryption. As explained in Sections 3 and 4, 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 files, it would be better not to re-encrypt 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.

Due to the requirements over the format, 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 `intefile` GENIFF field, then other approaches have to be devised. Here follows an example using the BAM format.

When the result of an alignment is encoded 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 that 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 for the Core Experiment [1], we need to maintain the state of 248k blocks, for which a bit array of 31k bytes is enough. Using encryption in counter (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.

Regarding privacy aspects, when developing the new version of GENIFF tools, we have kept the same approach (see [2]) but defined new privacy rules in eXtensible Access Control Markup Language (XACML) [8], which exemplify two approaches. On the one hand, we have the case where by default we deny access, and the rules indicate the exceptions to this. The reader will see this for the rules affecting the roles "physician" and "researcher" (the first role is allowed to access the whole dataset, the later only the chromosome 2). On the other hand, we have the case where the default is to grant access, while the rules provide the exceptions. For an example of this, refer to the rules applying to the role "doctor", where the access to chromosome 2 is denied. These rules can be found in Annex I.

7 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) and the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 305444 (RD-Connect).

8 References

[1] ISO/IEC JTC 1/SC 29/WG 11 / N16526 - ISO/TC 276/WG 5 / N120 – Core Experiments on Genomic Information Representation, Chengdu, October 2016.
[2] Jaime Delgado, Silvia Llorente et al., ISO/IEC JTC 1/SC 29/WG 11 M39175, GENIFF (GENomic Information File Format), a proposal for a Secure Genomic Information Transport Layer (GITL) based on the ISO Base Media File Format, Chengdu, China, October 2016.
[3] Jaime Delgado, Silvia Llorente et al., ISO/IEC JTC 1/SC 29/WG 11 M39939, Genomic Information Compression and Storage CE4: DMAG-UPC's GENIFF v.2 implementation, December 2016.
[4] 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
[5] Heng Li, et al., The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009 Aug 15; 25(16): 2078–2079.
[6] Official Sequence Alignment / Map (SAM) Format Specification, https://samtools.github.io/hts-specs/
[7] Daniel Naro, Jaime Delgado and Silvia Llorente, ISO/IEC JTC 1/SC 29/WG 11 M39944, GENIFF (GENomic Information File Format) v2: Security and signature issues, January 2017.
[8] 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

Annex I – Examples of XACML rules

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

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

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

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

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

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

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

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

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

```
                        Category="urn:oasis:names:tc:xacml:3.0:role" AttributeId="role"
                        DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </Match>

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

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

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

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

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

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

                <!-- Which action  -->
                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                        VIEWCHROMOSOME
```

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

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