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/m40494
April 2017, Hobart, Australia

Source: DMAG-UPC, EPFL, BSC, Made of Genes
Status: Proposal
Title: Genomic Information Representation. Proposal for Part 3 on Protection, Application Programming Interfaces and Metadata
Authors: Jaime Delgado, Silvia Llorente, Daniel Naro (Distributed Multimedia Applications Group – Universitat Politècnica de Catalunya),
Claudio Alberti (EPFL),
Josep Lluís Gelpí, Dmitry Repchevsky, Romina Royo (Barcelona Supercomputing Center),
Leonor Frías, Oscar Flores (Made of Genes)

Index

1 Introduction

This document is a first proposal for Part 3 of the new MPEG Genomic Information Representation (MPEG-GIR) standard. This part is devoted to the description of 1) Protection information associated to genomic information, 2) Application Programming Interfaces (API's) to access, manipulate and convert genomic information expressed in MPEG Genomic standard to/from other Genomic Information Representations (GIR), and 3) Metadata.

The document defines the data structure of the protection information present in the protection boxes identified in Part 1 [1] of this standard. Next, it proposes several operations being part of the Application Programming Interfaces (API's) needed to access to, manipulate and convert, among other things, genomic information. Finally, it describes the metadata required at the different levels of information identified in [1]. This information has to be stored in the metadata boxes identified in Part 1 [1].

2 Definition of the *protection* data structure

This section presents the specific definition of the protection boxes identified in Part 1 [1], which are named `DG_protection`, `DT_protection` and `DS_protection`, corresponding to the levels they belong to, namely **dataset group**, **dataset** and **descriptor stream**, respectively.

These three types of protection boxes follow the same basic idea. Each box contains an XML document, which defines the different confidentiality, integrity, authentication and privacy rules elements that have to be applied to each level. The overall structure is as follows:

```
<protection_box>
      <Encryptions>
            <EncryptedData></EncryptedData>
            <EncryptedData></EncryptedData>
      </Encryptions>
      <Privacy_rules></Privacy_rules>
      <Signatures>
            <Signature></Signature>
            <Signature></Signature>
      </Signatures>
</protection_box>
```

The next subsections describe the meaning of each element. A complete XML Schema is provided in Annex I.

2.1 Encryption

This section describes how the different elements represented inside the MPEG genomic format are encrypted. First, `gen_info` elements encryption is presented. Then, the encryption of stream blocks is shown.

2.1.1 Encryption of `gen_info` elements

At each level, the protection box conveys the information on how its sibling boxes and the protection boxes of a layer below are encrypted. This information is represented as an *XML encryption* element, as specified in [2]. The data reference of the XML Encryption tag is used to both indicate the location of the cipher text and its function. In other words, if a `gen_info` element

is encrypted, its content is replaced with the cipher text. These references are constructed using a URI syntax, as described in the Authenticity section.

2.1.2 Encryption of descriptor streams blocks

In the descriptor stream protection box, we additionally have the option to specify the encryption of the specific blocks of data contained by the descriptor stream. In this case, the information is conveyed in a new XML tag: `<encryption-blocks>`.

This tag contains a `KeyInfo` tag as specified in [2], an `IV` element in Base64 and an `encryptionActive` tag represented in Base64, which encodes a bit array where each bit indicates if the corresponding block is encrypted or not. The corresponding type definition in the XML Schema is as follows:

```
<xs:complexType name="encryption-blocksType">
    <xs:sequence>
      <xs:element ref="xd:KeyInfo"
                  xmlns:xd="http://www.w3.org/2000/09/xmldsig#"/>
      <xs:element type="xs:base64Binary" name="iv"/>
      <xs:element type="xs:base64Binary" name="encryption-active"/>
    </xs:sequence>
  </xs:complexType>
```

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

The policy of which blocks to encode is left to the will of the user: each block can be encrypted on its own, independently of the other blocks.

In order to make this selection as precise as possible, the user might have to tune the random access parameters in order to meet the requirements. For example reducing the number of cases where read pairs are stored together, and adapting the size of the boxes in order to limit the cases where a read is stored in a block with a different policy.

The following figure shows how reducing the threshold for storing random pairs together can be adapted to respect the privacy settings. A white background rectangle represents an unprotected block, whereas a red background stands for a protected one. A green line represents a read stored as plaintext, and red as cipher, a dashed line connects the two reads of a same pair. By reducing the threshold in the second case, we ensure that the second read in the third pair is properly stored in a protected block.

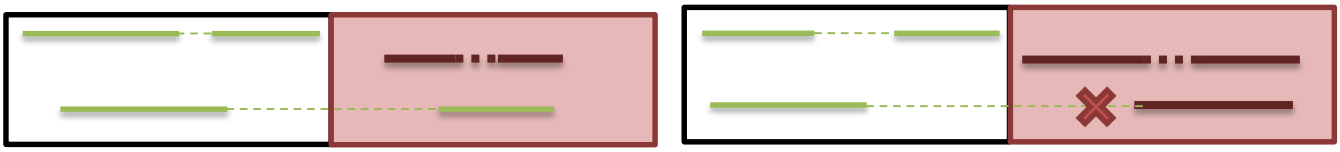

Figure 2.1 - Modifying threshold for read pairs for protection

In the following case, all reads were in one encrypted block: public access reads (in green) were stored in this encrypted block, although only those base pairs between the dashed lines should be

protected. By adapting the size and creating new blocks, we can keep access to more open access reads.

Figure 2.1 - Modifying blocks range for protection

2.2 Privacy rules

At each level, the privacy rules indicate which rules are at hand for sibling boxes and the protection boxes of a layer below, which are identified with the URIs listed in the Authenticity section. The protection box has to be viewable by any user who has to have access to some information stored in that layer or at a layer below.

In the last level, the descriptor stream level, the rules can specify regions of the genome by specifying one of the label Ids. As such, the regions where the rules apply can be updated independently from the privacy rules.

The privacy rules tag should be already compliant with the XACML specification [3]: by exporting this tag as the root element of a new document, the privacy handling application should have a valid policy document. The only modification that could take place is the conversion from the label centric types of rules, to a new resource description model to be defined. This new description model should contain the indexing attributes needed to resolve the privacy rules when the user submits a random access query based on the indexing.

Here follows a conceptual example of converting label based privacy rules, to genomic coordinates based ones.

	Labels	Rules
Original	Label 1: Ref1 20-30 Ref2 500-600 Label2: Ref3 50-60	Physician: access to label1and label2 Researcher: access to label1 Default: no access
After interpretation	Label 1: Ref1 20-30 Ref2 500-600 Label2: Ref3 50-60	Physician: access to Ref1 20-30, Ref2 500-600, ref3 50-60 Researcher: access to ref3 50-60 Default: no access

2.3 Authentication

2.3.1 General case

At each level, the protection box might include the authentication information. This includes signing of a subset of the sibling boxes having no `gen_info` children, and a signature for a subset of protection boxes of one level below. If the signed `gen_info` is encrypted, the signature uses the cipher text as input. The content that can be signed is as follows:

Protection box at level	Signs the content of
Dataset group (dgcn)	• Datasets group header (dghd) • Reference genome (rfgn) • Dataset group's metadata (dgmd) • All Dataset protection within the dataset group (dtpr)
Dataset (dtcn)	• Dataset header (dthd) • Master index table (mitb) • Parameters sets (pars) • Dataset's metadata (dtmd) • All Descriptors stream protection within the dataset (dspr)
Descriptors stream (dscn)	• Descriptors stream header (dshd) • Local index table (litb) • Descriptors stream's metadata (dsmd)

Each signature is provided as an XML detached signature [4]. No canonicalization of the data is performed: the input of the authentication algorithm is the byte stream as stored on the storage medium following the standard. The reference URI's are constructed as follows:

Protection box at level	Content to point to	URI construction:
Dataset group (dgcn)	dghd	<file URI>/datasetgroup/{id}/header
	rfgn	<file URI>/datasetgroup/{id}/refgen
	dgmd	<file URI>/datasetgroup/{id}/metadata
	dtpr	<file URI>/datasetgroup/{id}/dataset/{d_id}/protection
Dataset (dtcn)	dthd	<file URI>/datasetgroup/{id}/dataset/{d_id}/header
	mitb	<file URI>/datasetgroup/{id}/dataset/{d_id}/mitb
	pars	<file URI>/datasetgroup/{id}/dataset/{d_id}/pars
	dtmd	<file URI>/datasetgroup/{id}/dataset/{d_id}/metadata
	dspr	<dataset URI>/destream/{id}/protection
Descriptors stream (dscn)	dshd	<dataset URI>/destream/{id}/header
	litb	<dataset URI>/destream/{id}/litb
	dsmd	<dataset URI>/destream/{id}/metadata

The content to sign for each box corresponds to the payload in each `gen_info` structure. In other words, the V in the KLV structure.

Each element can be signed multiple times, i.e. there can be multiple signatures pointing to the same resource, if, for example, some content has to be signed by multiple parties.

2.3.2 Authenticity of the dataset group protection box

In order to guarantee the authenticity of the protection box at the dataset group level, an enveloped signature can be provided, which will be located within the protection tag.

3 Privacy rules over genomic information: some concepts

In [5] and [6], several use cases for using genomic information were described. To complete the specification of privacy rules over genomic information, some concepts introduced in those use cases are required. The concepts are separated into five different categories: *Roles*, *Actions*, *Objects*, *Conditions* and *Notifications*, this last one indicating which information will be send and to whom when an action is performed. For example, the genome owner will receive a notification when someone has accessed the records. Table 3.1 presents the values currently identified for each category. [NOTE: This table needs to be completed].

These concepts should then be used to define privacy rules. For this purpose, they need to be included inside XACML rules. Therefore, they should be defined in a vocabulary, with an XML schema or another formal mechanism, external to the rules themselves.

Annex II provides some examples of XACML privacy rules and how use of genomic information is authorized according to these rules.

Table 3.1. Concepts extracted from genomic use cases

Roles	Actions	Objects	Conditions	Notifications
Analyst	View	Regions of Variant / Alignment / Sequence	Number of times	To owner: data viewed
Researcher	View Chromosome	Complete genome Variant / Alignment / Sequence	Time period	To owner: data modified
Healthcare	View Region	Non-aligned reads	Country / Organization	To owner: data viewed, re-contact information
Owner	Genetic Analysis	Outcome genetic analysis statistic	Conclusions publishable	Yes, request more data
Custodian	Research Project	Outcome genetic analysis pileup	Samples publishable	To research project: data available
Forensics	Donation	Outcome genetic analysis variants		No, anonymous
Individual	Share	Genetic analysis		No, it is donated after individual's death
	Paternity test	Anonymized genomic data		Unknown
	Forensic	Result paternity test		
	Metadata Manipulation	Complete genome		
	Personalized Medicine			
	Personalized Treatment			
	Workflow			
	Modification (related to workflow operations)			
	More experiments			

4 Application Programing Interfaces (API's) for MPEG-GIR

4.1 API definition

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

The operations provided by this API may affect to different aspects of genomic information and its associated metadata, protection information and other fields contained at each level. By level we understand File, Dataset Group, Dataset, Descriptor Stream and Block (including those for storage and streaming), as described in Part 1 of this standard [1]. They may include functionalities such as providing access, performing modifications, authorizing operations or integrity verification. Each level has to specifically define the functionality of each operation.

Table 4.1 shows an initial classification of the different kind of operations defined in the API. Each operation category may contain different operations, depending on the information available for each level of genomic information.

Table 4.1 – Operations classification

Category	Description
Access	Gives the requested information to the user.
Modification	Changes the information indicated by the user.
Authorization	Checks that the user has permission to perform an operation.
Verification	Checks the integrity of some information indicated by the user.
Conversion	Converts some information from / to MPEG-GIR to other existing GIR formats.
Beacon-like	Provides information about MPEG-GIR in the form of beacons (statistical, appearance, etc.) [7].

Tables 4.2 to 4.4 briefly describe the operations considered in different categories. Specifically, table 4.2 lists access operations, table 4.3 lists modification operations and table 4.4 lists the rest of foreseen operations, indicating which category they belong.

Table 4.2 – Access Category Operations

Operation name	Brief description
GetHeader	Returns the content of the complete header of the corresponding level.
GetHeaderField	Returns the content of a specific header field of the corresponding level.
GetMetadata	Returns the content of the complete metadata element of the corresponding level.
GetMetadataField	Returns the content of a specific metadata field of the corresponding level.
GetProtection	Returns the content of the complete protection element of the corresponding level.
GetProtectionField	Returns the content of a specific protection field of the corresponding level.
GetLabel	Returns the content of a specific label inside a dataset.
GetDatasetGroup	Returns the content of a specific dataset group.
GetDataset	Returns the content of a specific dataset.
GetData	Returns the content of a level, that is, the payload. It can be filtered by positions and reference sequence.
isSetField	Checks if a field has a value in the corresponding level, in order to access such field using one of the access methods.
ListMetadata	Lists all the metadata contained in a file.

ListMetadataField	Lists all the values of a metadata field contained in a file.
ListProtection	Lists all the protection information contained in a file.
ListProtectionField	Lists all the values of a protection field contained in a file.
ListLabel	List labels inside a dataset.
SearchMetadata	Searches for some value inside the metadata contained in the file.
SearchMetadataField	Searches for some value inside a specific field of the metadata contained in the file.
SearchProtection	Searches for some value inside the protection contained in the file.
SearchProtectionField	Searches for some value inside a specific field of the protection contained in the file.
SearchLabel	Searches for some value inside labels in a dataset.
StreamData	Send stored data using streaming.

Table 4.3 – Modification Category Operations

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

Table 4.4 – Other Category Operations

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

Table 4.5 contains the mapping matrix between operations and levels, indicating which operation is available at each level.

Table 4.5 – Operation matrix

Level / Operation	File	Datasets Group	Dataset	Descriptor Stream	Block	Transport Block	Packet
GetHeader	x	x	x	x			
GetHeaderField	x	x	x	x			
GetMetadata		x	x	x			

GetMetadataField		x	x	x			
GetProtection		x	x	x			
GetProtectionField		x	x	x			
GetLabel			x				
GetDatasetGroup	x						
GetDataset		x					
GetData					x	x	x
AddHeaderField	x	x	x	x			
AddMetadata		x	x	x			
AddMetadataField		x	x	x			
AddProtection		x	x	x			
AddProtectionField		x	x	x			
AddLabel			x				
AddData					x	x	x
UpdateHeader	x	x	x	x			
UpdateHeaderField	x	x	x	x			
UpdateMetadata		x	x	x			
UpdateMetadataField		x	x	x			
UpdateProtection		x	x	x			
UpdateProtectionField		x	x	x			
UpdateLabel			x				
UpdateData					x	x	x
isSetField	x	x	x	x	x	x	x
ListMetadata		x	x	x			
ListMetadataField		x	x	x			
ListProtection		x	x	x			

ListProtectionField		x	x	x			
ListLabel			x				
SearchMetadata		x	x	x			
SearchMetadataField		x	x	x			
SearchProtection		x	x	x			
SearchProtectionField		x	x	x			
SearchLabel			x				
Authorize	x	x	x	x			
Verify	x	x	x	x	x	x	x
ConvertTo		x	x	x			
ConvertFrom		x	x	x			
StreamData		x	x	x	x		
Beacon		x					

Table 4.6 contains some examples of operations for the Datasets Group level. For each operation, we provide its name, its URL for a hypothetical REST-based service and a brief description of what it should do. Access operations are currently mapped to the GET HTTP method [8] and modification operations (add and update) are mapped to the POST HTTP method. The use of the PUT HTTP method for modification operations needs further discussion.

Table 4.6 – Some examples of operations for the Datasets Group level

Operation name	URL (for REST-based API's)	Description
getDatasetsGroupHeader	GET /datasetgroup/{id}/header	Returns the content of the datasets group header for the dataset group identified by {id}.
addDatasetsGroupHeader	POST /datasetgroup/{id}/header	Adds the content of the datasets group header for the dataset group identified by {id}.
updateDatasetsGroupHeader	POST /datasetgroup/{id}/header	Updates the content of the datasets group header for the dataset group identified by {id}.
getDatasetsGroupHeader Field	GET /datasetgroup/{id}/hfield={field_name}	Returns a field identified by field_name from the datasets group header.
updateDatasetsGroupHeader Field	POST /datasetgroup/{id}/hfield={field_name}	Updates a field identified by field_name in the datasets group header.
getDatasetsGroup Metadata	GET /datasetgroup/{id}/metadata	Returns the content of the datasets group metadata for the dataset group identified by {id}.
addDatasetsGroupMetadata	POST /datasetgroup/{id}/metadata	Adds the content of the datasets group metadata for the dataset group identified by {id}.

updateDatasetsGroupMetadata	POST /datasetgroup/{id}/metadata	Updates the content of the datasets group metadata for the dataset group identified by {id}.
getDatasetsGroupMetadataField	GET /datasetgroup/{id}/mfield={field_name}	Returns a field identified by field_name from the datasets group metadata.
addDatasetsGroupMetadataField	POST /datasetgroup/{id}/mfield={field_name}	Adds a field identified by field_name in the datasets group metadata.
updateDatasetsGroupMetadataField	POST /datasetgroup/{id}/mfield={field_name}	Updates a field identified by field_name in the datasets group metadata.
getDatasetsGroupProtection	GET /datasetgroup/{id}/protection	Returns the content of the datasets group protection for the dataset group identified by {id}.
addDatasetsGroupProtection	POST /datasetgroup/{id}/protection	Adds the content of the datasets group protection for the dataset group identified by {id}.
updateDatasetsGroupProtection	POST /datasetgroup/{id}/protection	Updates the content of the datasets group protection for the dataset group identified by {id}.
getDatasetsGroupProtectionField	GET /datasetgroup/{id}/pfield={field_name}	Returns a field identified by field_name from the datasets group protection.
addDatasetsGroupProtectionField	POST /datasetgroup/{id}/pfield={field_name}	Adds a field identified by field_name in the datasets group protection.
updateDatasetsGroupProtectionField	POST /datasetgroup/{id}/pfield={field_name}	Updates a field identified by field_name in the datasets group protection.
streamDatasetsGroup	GET /datasetgroup/{id}/stream	Streams the data contained in the datasets group identified by id.

5 Metadata

5.1 Background

Input contribution [5] 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 objective is to propose metadata fields with the same expressiveness then EGA's metadata, and the ability to use the relevant attributes listed in MIAME or the Genomic Standards Consortium (http://gensc.org/).
In the case of EBI's metadata, there are different foreign keys in use to reference metadata, but this information is not needed in the case of a file format where the relationships are implicit in the structure of the file. Additionally, we should avoid having to repeat redundant information: for example if all samples in a study undergo the same experiments we should not repeat it unnecessary.
The proposed metadata schema aims at:

- being modular (at creation time it can be decided to add more or less description, from almost none to as complete as EGA's metadata and MIAME/GENSC attributes)
- use the file format to avoid using foreign keys
- avoid repetitions: the dataset's metadata inherits the dataset groups metadata, and can correct it if needed.

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 introduces a first proposal to be discussed.

5.2 Introduction

There are different usages for the genomic data, and each one requires a different set of description values. We define a structure that is adaptable to different requirements. It is specified in XML, where the elements are meant to be redefined (or, rather, "specialized") as new needs are uncovered. The structure is to be maintained across different versions of metadata. In order to achieve this, the metadata format should leverage 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.

[NOTE: 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 genetic 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 centres	List of type of project centres	Yes
Description	String	No
Samples	List of type of samples	Yes

If a given use case also 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 centres	List of type project centres	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 Tables 1 and 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: Basic sample's metadata

Field name	Field type	Mandatory
Id	URI	Yes

However, in the context of the Genome Wide Association Study (GWAS), 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 centres	List of type project centres	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 label to region information

Field name	Field type	Mandatory
Title	String	No
Type	Controlled vocabulary	No
Abstract	String	No
Project centres	List of type project centres	No
Description	String	No
Samples	List of type sample	No
LabelToRegion	Label id and ontology term	Yes

5.5 Examples

This section provides some examples to illustrate the definition of different metadata for different levels of genomic information, as introduce above.

5.5.1 Example 1

Let us imagine 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 study	
Project centres	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 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 study
Project centres	UPC
Samples	<table><tr><td>Id</td><td>Patient A</td></tr><tr><td>IsCase</td><td>False</td></tr></table><table><tr><td>Id</td><td>Patient B</td></tr><tr><td>IsCase</td><td>True</td></tr></table>

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

Field name	Field value
Type	Whole Genome Sequencing
Samples	<table><tr><td>Id</td><td>Patient A</td></tr><tr><td>IsCase</td><td>False</td></tr></table>

And the second one with:

Field name	Field value
Type	Whole Genome Sequencing
Samples	<table><tr><td>Id</td><td>Patient B</td></tr><tr><td>IsCase</td><td>True</td></tr></table>

6 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).

7 References

[1] ISO/IEC JTC 1/SC 29/WG 11 / N16722 - ISO/TC 276/WG 5 / N151 – Text of ISO/IEC NNNNN-1WD 1, Transport and Storage of Genomic Information, Geneva, January 2017.
[2] W3C Recommendation, XML Encryption Syntax and Processing, 10 December 2002. https://www.w3.org/TR/xmlenc-core/
[3] OASIS Standard, eXtensible Access Control Markup Language (XACML) Version 3.0, 22 January 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
[4] W3C Recommendation, XML Signature Syntax and Processing (Second Edition), 10 June 2008. https://www.w3.org/TR/xmldsig-core/
[5] 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.
[6] ISO/IEC JTC 1/SC 29/WG 11 / N16725 - ISO/TC 276/WG 5 / N154 – Use cases for an efficient genomic information representation, Geneva, January 2017.
[7] Genomic Alliance for Genomics and Health, Beacon project, http://ga4gh.org/#/beacon, 2017.
[8] Internet Engineering Task Force, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, https://tools.ietf.org/html/rfc7231, June 2014.
[9] Global Alliance for Genomics and Health, GA4GH Genomics API, https://ga4gh-schemas.readthedocs.io/en/latest/, 2017.
[10] Ecma International, Standard ECMA 404, The JSON Data Interchange Format, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, October 2013.

Annex I – Protection boxes XML Schemas

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

I.1 Dataset group protection box XML schema

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

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

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

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

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

</xs:schema>
```

I.2 Dataset protection box XML schema

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

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

I.3 Descriptor stream protection box XML schema

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

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

Annex II – Examples of XACML rules

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

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

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

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

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

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

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

```
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
                    <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>
```

II.1 XACML authorization for genomic information access

This section describes 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 II.1, 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.

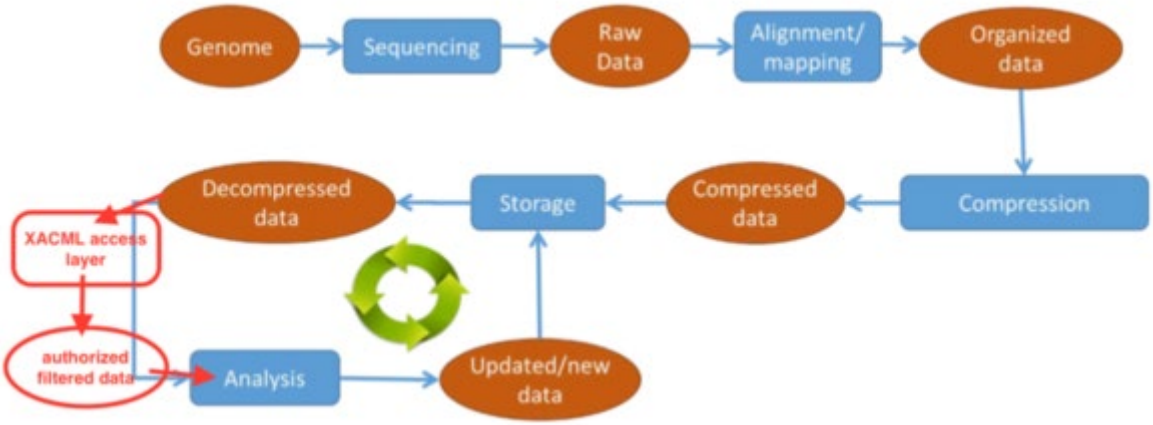

Figure II.1 - Complete flow for genomic data, new step added

Figure II.2 shows the process of evaluation of access rights:

1. A user sends a request, which is intercepted by the XACML engine
2. The XACML engine converts the request into the 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.

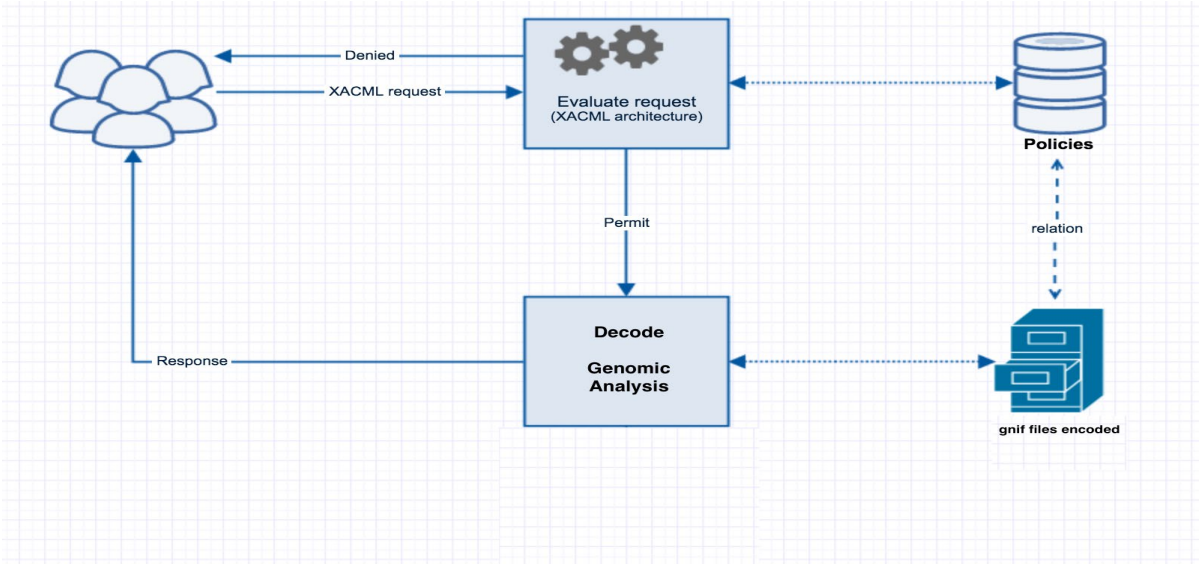

Figure II.2 – Authorization flow proposed

Annex III – Complete list of operations for GA4GH

III.1 Summary of the format for operations defined in GA4GH API

In order to define the API in MPEG-GIR, other APIs concerning genomic information have been analyzed. This section summarizes the one defined by GA4GH [9]. All requests and responses are defined using JSON [10].

Table III.1 summarizes the different kind of operations defined in the GA4GH API. They define operations to get one element by id, a set of elements by id (in this case, the data type set is already defined), search elements, search sets of elements and a specific operation to list references (it is the only one with this structure).

For the operations accessing to one element by id, a common HTTP command line structure is defined, using the GET HTTP command, the name of the element being requested, which could be a reference, variant, metadata, etc. To identify the id, it is appended to the name of the element. The URL used for these operations is /{element_name}/{id}.

The operations for accessing to a set of elements by id are quite similar to the ones for elements. The only difference is that they refer to a set of elements instead of only one element. So, the HTTP command and URL are almost the same as for one element.

The search operations use the POST HTTP command to perform the request. Again, the operations for searching an element or a set of elements only differ on if they refer to an element name or a set of elements. The URL used is /{element_or_set_name}/search.

Finally, there is one list operation, used to list reference bases. The form of this operation is POST /listreferencebases. No more listing operations are defined in the API, although the authors indicate at some points of the API that further operations are required, as it is still work in progress. Table III.2 contains the complete list of operations up to January 2017.

Table III.1 – Summary of operations

Operation type	HTTP command line	Comments
Get one element	GET /{element_name}/{id}	Gets an element by id. Example: Response GetIndividual (request)
Get a set of elements	GET /{element_set_name}/{set_id}	Gets a set of elements by id. Example: Response GetReadGroupSet (request)
Search elements	POST /{element_name}/search	Gets a list of elements accessible through the API. Example: Response SearchIndividuals (request)
Search a set of elements	POST /{element_set_name}/search	Gets a list of set elements matching the search criteria. Example: Response SearchReadsGroupSets (request)
List references	POST /listreferencebases	Lists Reference bases by ID and optional range. Example: Response ListReferenceBases(request)

Table III.2 shows a brief description of the operations to be supported.

Level	Object types	Related Operations in Service
common	GAException	No associated service.
common	Position	No associated service.
common	ExternalIdentifier	No associated service.
assay_metadata	Experiment	No associated service.
assay_metadata	Analysis	No associated service.
bio_metadata	Individual	GetIndividual (request) - GET /individuals/{id} Gets an Individual by id. SearchIndividuals (request) - POST /individuals/search Gets a list of Individuals accessible through the API.
bio_metadata	Biosample	GetBiosample (request) - GET /biosamples/{id} Gets a Biosample by ID. SearchBiosamples (request) - POST /biosamples/search Gets biosamples accessible through the API.
metadata	OntologyTerm	No operation in API.
metadata	Dataset	GetDataset(request) - GET /datasets/{dataset_id} Gets a Dataset by ID. SearchDatasets (request) – POST /datasets/search Gets a list of Dataset matching the search criteria.
metadata	Program	No operation in API.
read	ReadStats	No operation in API.
read	ReadGroup	Operations for GroupSets (below).
read	ReadGroupSet	GetReadGroupSet (request) - GET /readgroupsets/{read_group_set_id} Gets a ReadGroupSet by ID. SearchReadsGroupSets (request) - POST /readgroupsets/search Gets a list of ReadGroupSet matching the search criteria.
read	LinearAlignment	No operation in API.
read	ReadAlignment	SearchReads (request) - POST /reads/search Gets a list of ReadAlignments for one or more ReadGroups.
read	CigarUnit	No operation in API.
references	Reference	GetReference(request) - GET /references/{reference_id} Gets a Reference by ID. ListReferenceBases(request) - POST /listreferencebases Lists Reference bases by ID and optional range. SearchReferences(request) - POST /references/search Gets a list of Reference matching the search criteria.
references	ReferenceSet	GetReferenceSet(request) - GET /referencesets/{reference_set_id} Gets a ReferenceSet by ID. SearchReferenceSets(request) - POST /referencesets/search Gets a list of ReferenceSet matching the search criteria.
variant	VariantSetMetadata	No operation in API.
variant	VariantSet	GetVariantSet(request) - GET /variantsets/{variant_set_id} Gets a VariantSet by ID. SearchVariantSets(request) - POST /variantsets/search

		Gets a list of VariantSet matching the search criteria.
variant	CallSet	GetCallSet(request) - GET /callsets/{id} Gets a CallSet by ID. SearchCallSets(request) - POST /callsets/search Gets a list of call sets matching the search criteria.
variant	Call	No operation in API.
variant	Variant	GetVariant(request) - GET /variants/{id} Gets a Variant by ID. SearchVariants(request) - POST /variants/search Gets a list of Variant matching the search criteria.
rna_quantification	RnaQuantificationSet	GetRnaQuantificationSet(request) - GET /rnaquantificationsets/{id} Gets a RnaQuantificationSet by ID. SearchRnaQuantificationSets(request) - POST /rnaquantificationsets/search Gets a list of RnaQuantificationSet matching the search criteria.
rna_quantification	RnaQuantification	GetRnaQuantification(request) - GET /rnaquantifications/{id} Gets a RnaQuantification by ID. SearchRnaQuantifications(request) - POST /rnaquantifications/search Gets a list of RnaQuantification matching the search criteria.
rna_quantification	ExpressionLevel	GetExpressionLevel(request) - GET /expressionlevels/{id} Gets a ExpressionLevel by ID. SearchExpressionLevels(request) - POST /expressionlevels/search Gets a list of ExpressionLevels matching the search criteria.
allele_annotations	AnalysisResult	Under construction.
allele_annotations	AlleleLocation	Under construction.
allele_annotations	VariantAnnotationSet	GetVariantAnnotationSet(request) – GET /variantannotationsets/{variant_annotation_set_id} Gets a VariantAnnotationSet by ID. SearchVariantAnnotationSets(request) - POST /variantannotationsets/search Returns a list of available variant annotation sets.
allele_annotations	HGVSAnnotation	Under construction.
allele_annotations	TranscriptEffect	Under construction.
allele_annotations	VariantAnnotation	SearchVariantAnnotation(request) - POST /variantannotations/search Returns a list of variant annotation matching the search criteria. Under construction.
sequence_annotations	Attributes	No operation in API.
sequence_annotations	AttributeValue	No operation in API.
sequence_annotations	AttributeValueList	No operation in API.
sequence_annotations	Feature	GetFeature(request) – GET /features/{id} Gets a Feature by ID. SearchFeatures(request) – POST /features/search

		Gets a list of Feature matching the search criteria.
sequence_annotations	FeatureSet	GetFeatureSet(request) – GET /featuresets/{id} Gets a FeatureSet by ID. SearchFeatureSets(request) – POST /featuresets/search Gets a list of FeatureSet matching the search criteria.
struct	Struct	No associated service.
struct	Value	No associated service.
struct	ListValue	No associated service.
genotype_phenotype	PhenotypeAssociationSet	SearchPhenotypeAssociationSets(request) – POST /phenotypeassociationsets/search Gets a list of association sets accessible through the API.
genotype_phenotype	EnvironmentalContext	No operation in API.
genotype_phenotype	PhenotypeInstance	SearchPhenotype(request) – POST /phenotypes/search Gets a list of phenotypes accessible through the API.
genotype_phenotype	Evidence	No operation in API.
genotype_phenotype	FeaturePhenotypeAssociation	SearchPhenotypeAssociations(request) - POST /featurephenotypeassociations/search Gets a list of genotype-phenotype associations accessible through the API.

III.1.1 Example of request and response for API operations

SearchReadsRequest (request for operation searchReads)

Fields:

- read_group_ids (string) – The ReadGroups to search. At least one id must be specified.
- reference_id (string) – The reference to query. Leaving blank returns results from all references, including unmapped reads - this could be very large.
- start (long) – The start position (0-based) of this query. If a reference is specified, this defaults to 0. Genomic positions are non-negative integers less than reference length. Requests spanning the join of circular genomes are represented as two requests one on each side of the join (position 0).
- end (long) – The end position (0-based, exclusive) of this query. If a reference is specified, this defaults to the reference's length.
- page_size (integer) – Specifies the maximum number of results to return in a single page. If unspecified, a system default will be used.
- page_token (string) – The continuation token, which is used to page through large result sets. To get the next page of results, set this parameter to the value of next_page_token from the previous response.

SearchReadsResponse (response for operation searchReads)

Fields:

- alignments (list of ReadAlignment) – The list of matching alignment messages, sorted by position. Unmapped reads, which have no position, are returned last.

- next_page_token (string) – The continuation token, which is used to page through large result sets. Provide this value in a subsequent request to return the next page of results. This field will be empty if there aren't any additional results.

III.1.2 Definition of variant and its serialization in JSON

Definition	JSON Serialization

message Variant {
  string id = 1;
  string variant_set_id = 2;
  repeated string names = 3;
  int64 created = 4;
  int64 updated = 5;
  string reference_name = 6;
  int64 start = 7;
  int64 end = 8;
  string reference_bases = 9;
  repeated string alternate_bases = 10;
  map<string, google.protobuf.ListValue> info = 11;
  repeated Call calls = 12;
}
	{
	  "id": "gv79384-3200-11",
	  "variantSetId": "vs-44-1",
	  "names": [
	  "rs110",
	  "Victoria"
	  ],
	  "created": 1446842841,
	  "updated": null,
	  "start": 1000,
	  "end": 1001,
	  "referenceBases": "A",
	  "alternateBases": [
	  "C",
	  "CTATCTT"
	  ],
	  "info": {
	  "variantfacts": ["is_interesting", "is_long"],
	  "numberOfPapers": ["11"]
	  },
	}

III.1.3 Objects defined by GA4GH and their relationship