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/m39941
January 2017, Geneva, Switzerland

Source: **DMAG-UPC, CNAG-CRG, BSC**
Status: **Proposal**
Title: **User's Guide for DMAG-UPC's GENIFF v.2 software used for Genome information CEs**
Authors: **Daniel Naro, Jaime Delgado, Silvia Llorente, Sara Rodríguez-Cubillas (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)

Index

1 Introduction

This is the User's Guide for the GENIFF v.2 software used for the Core Experiment CE4 on Genomic Information Compression and Storage. CE4 is described in output document N16526. The software is distributed to those interested in running the experiments.

2 Creating the execution environment

The project has been written on Linux and not tested on other platforms. There is nothing limiting to this platform, if not, maybe, the availability of the libraries. To execute the code we suggest creating a virtual environment. The following instructions summarize the installation process. The described process is tested on Ubuntu 16.04.

```
sudo apt-get install libxml++2.6-dev libcrypto++-dev openjdk-8-jdk
#Download the binary writingGENISOBMFF and ExtendHeader.jar, execute with
./writingGENISOBMFF
```

Please note that there is a known issue when the java library used when compiling does not match the shared library of the system when executing. In order to address this issue please check that it is indeed the libjvm.so provided by openjdk-8-jdk which is used.

3 Using the tool

There are two main tools: first, a command line program which allows the basic operations of creating a GENIFF instance and retrieving information from such files. The other tool is a Server providing access to the file's content through an HTTP access.

3.1 Command line utility

The first step when using this utility will be to create a GENIFF instance. In order to do so, the program requires a description of the content. An example of such description/configuration is provided in the next section of this document. The command to perform such an action is:

`./writingGENISOBMFF` encode *filename_output configuration_file*

In order to perform an extraction of the content within a GENIFF file, we have different options to make the selection of extracted content broader or rather more precise.

The broadest selection is the one with no parameter at all: this will create in the current directory copies of the different files (data files, index files, XACML rules,…) for all datasets and all streams within the datasets.

`./writingGENISOBMFF` decode *input_geniff_file_name*

The first step to reduce the output is to specify the name of the dataset. The output of this command is the same as previously, but just for the selected dataset.

`./writingGENISOBMFF` decode *input_geniff_file_name name_dataset*

One can further reduce the output by also specifying the name of the stream.

`./writingGENISOBMFF` decode *input_geniff_file_name name_dataset name_stream*

Analogously to the previously described methods, the type of extracted content can also be specified:

`./writingGENISOBMFF` decode *input_geniff_file_name [XACML|DATA|HEADER|INDEX]*
`./writingGENISOBMFF` *decode input_geniff_file_name [XACML|DATA|HEADER|INDEX] name_dataset*
`./writingGENISOBMFF` *decode input_geniff_file_name [DATA|INDEX] name_dataset name_stream*

Finally there are two extra commands. Both output an xml file through the standard output, which needs to be redirected to the desired location. The first command allows describing the content: it specifies the file's hierarchy (study, dataset, streams) and important parameters for each (such as encryption salts) alongside information of byte offset within the file to allow the retrieval of information by the server.

`./writingGENISOBMFF` *decode input_geniff_file_name DESCRIPTION*

The second is intended for debugging purposes/understanding the GENIFF's structure: it returns also a XML file, but here the tags match the GENIFF's boxes. The user will find here an indication of all the parameters stored within each file (including name, type and value), and information allowing to position the box in a hexadecimal tool for example. The command for this is:

`./writingGENISOBMFF` *decode input_geniff_file_name DEBUG*

3.2 Configuration of the encoding

As previously stated, when encoding a file we need to provide a configuration. One example of configuration follows. In this example we have five datasets. The first three store BAM information. More specifically:

- Dataset 1: stores the information in the form of multiple streams. Each of this streams has be obtained from the MPEG's input Id05, by extracting with samtools the information in order to recover a file with the information aligned on only one reference sequence (namely NC_007605 and hs37d5). Note also that the encryption instruction for the dataset is overwritten at the stream level.
- Dataset 3: In this case the encryption is performed at the BAM's block level. For each compression block containing one reference aligned on the reference sequence 1, its information is stored as encrypted content. Other blocks are left unmodified.

Datasets 4 and 5 encode a FASTQ file compressed as gzip. The first one is encrypted, the second one is not.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <config version="2" java_class_path="place here the absolute path to
ExtendHeader.jar">
   <study>
      <dataset type="SAM" compression="BAM_multi_stream" ID="1"
encryption="NONE" header_filename="9827_2_49.bam.header">
         <XACML>XACMLPolicy.xml</XACML>
         <stream encryption="ENTIRE" name="NC_007605">
            <source>9827_2_49_NC_007605.bam</source>
            <index>9827_2_49_NC_007605.bam.bai</index>
         </stream>
```

```
        <stream encryption="NONE" name="hs37d5">
          <source>9827_2_49_hs37d5.bam</source>
          <index>9827_2_49_hs37d5.bam.bai</index>
        </stream>
      </dataset>
      <dataset type="SAM" compression="BAM" ID="2" encryption="ENTIRE">
        <stream>
          <source>wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam</source>
          <index>wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam.bai</index>
        </stream>
      </dataset>
      <dataset type="SAM" compression="BAM" ID="3"
encryption="BLOCK_LEVEL">
        <stream>
          <source>wgEncodeCshlShortRnaSeqGm12878CellShortAln2.bam</source>

<index>wgEncodeCshlShortRnaSeqGm12878CellShortAln2.bam.bai</index>
        </stream>
      </dataset>
      <dataset type="FASTQ" compression="GZ" ID="4" encryption="ENTIRE">
        <stream>

<source>wgEncodeRikenCageK562CytosolPapRawDataRep1.fastq.gz</source>
        </stream>
      </dataset>
      <dataset type="FASTQ" compression="GZ" ID="5" encryption="NONE">
        <stream>

<source>wgEncodeRikenCageK562CytosolPapRawDataRep2.fastq.gz</source>
        </stream>
      </dataset>
   </study>
</root>
```

3.3 Analysing the file

In order to understand the content of the file, one can use the previously introduced DEBUG option. Here after we comment on interesting regions of its output.

For example for the first stream in the first dataset, we can find its relevant information at the location /file/genstudy/gendatas[1]/genrecor[1] (expressed as XPath). The information stored there is the following one:

```
<genrecor content_size="54977" content_starts_at="69917">
          <recohead content_size="46544" content_starts_at="69933">
            <flags represented_as="uint8_t" value="0"/>
            <ID represented_as="uint64" value="0"/>
            <filename_size represented_as="uint8_t" value="23"/>
```

```
            <filename represented_as="cstring"
value="9827_2_49_NC_007605.bam"/>
            <name_size represented_as="uint8_t" value="9"/>
            <name represented_as="cstring" value="NC_007605"/>
            <datasetType represented_as="uint8_t" value="1"/>
            <extra_data_length represented_as="uint32" value="0"/>
            <extra_data represented_as="char_array" value=""/>
            <FASTQ_compression represented_as="uint8_t" value="0"/>
            <SAM_compression represented_as="uint8_t" value="2"/>
            <ENCRYPTION_type represented_as="uint8_t" value="1"/>
            <salt_length represented_as="uint8_t" value="16"/>
            <salt represented_as="unsigned_char_array"
value="6cf18e27686aaf5dff20482230bae278"/>
            <samhsamh content_size="46460" content_starts_at="70017"
represented_as="CharArrayBox"/>
          </recohead>
          <genofile content_size="7521" content_starts_at="116477">
            <intefile content_size="7505" content_starts_at="116493"/>
          </genofile>
          <recoindx content_size="896" content_starts_at="123998"/>
        </genrecor>
```

The recohead box stores all the header information for the stream (from which we can highlight the encryption type set to one, the values of the salt used to generate the key for the encryption, or information such as the name of the stream). The stream is then composed of the genofile box, which contains an intefile box which contains the actual data. In other words, we now know that starting at position 7505, and for the next 116493 bytes, we have the encrypted content of our 9827_2_49_NC_007605.bam file.

Next we introduce the analogous section, but for the stream of the last dataset. This information can be found at /file/genstudy/gendatas[5]/genrecor (again expressed as an XPath).

```
        <genrecor content_size="844437526" content_starts_at="1268791724">
          <recohead content_size="34" content_starts_at="1268791740">
            <flags represented_as="uint8_t" value="128"/>
            <salt_length represented_as="uint8_t" value="16"/>
            <salt represented_as="unsigned_char_array"
value="4f45cf85a0b6fef8dc7125 5a52f1649"/>
          </recohead>
          <genofile content_size="844437476" content_starts_at="1268791774">
            <intefile content_size="844437460" content_starts_at="1268791790"/>
          </genofile>
        </genrecor>
```

The most striking difference is the length of the recohead. This can be explained by the value of the parameter flags: its first bit is set to one, indicating that the configuration is inherited from the dataset.

3.4 Assessing overheads

In order to assess the impact of the introduced overheads, one can use a configuration parameter which will deactivate the generation of all those regions of content with the actual information (such as the data of the streams, the indices, the privacy rules). The resulting file will contain the overhead introduced by GENIFF: box structure, and box parameters. In order to use this functionality, simply introduce the attribute debug and the value "DEBUG_NO_PAYLOAD". The resulting config tag is:

```
<config version="2" debug="DEBUG_NO_PAYLOAD" java_class_path="place here the absolut path to GENIFF_tool/JavaClasses/ExtendHeader.jar">
```

3.5 Preparing to serve the content

Before being able to serve the content, we need to obtain a description of it. As previously introduced this is obtain with the command DESCRIPTION.

```
<?xml version="1.0" encoding="UTF-8"?>
<file>
  <study>
    <dataset FASTQ_compression="" SAM_compression="BAM_multi_stream" encryption="NONE" name="test"
type="SAM">
      <xacml filename="Id05.geniff" from="55942" to="69885"/>
      <header filename="Id05.geniff" from="164" name="test" to="55926"/>
      <stream data_type="1" encryption="1" filename="Id05.geniff" from="116493"
name="9827_2_49_NC_007605.bam" name_bis="NC_007605" salt="6CF18E27686AAF5DFF20482230BAE278"
to="123982">
        <index filename="Id05.geniff" from="123998" to="124878"/>
      </stream>
      <stream data_type="1" encryption="0" filename="Id05.geniff" from="171448"
name="9827_2_49_hs37d5.bam" name_bis="hs37d5" to="304662520">
        <index filename="Id05.geniff" from="304662536" to="304759712"/>
      </stream>
    </dataset>
    <dataset FASTQ_compression="" SAM_compression="BAM" encryption="ENTIRE"
name="wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam" type="SAM">
      <xacml filename="Id05.geniff" from="304763950" to="304777893"/>
      <header filename="Id05.geniff" from="304759827"
name="wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam" to="304763934"/>
      <stream data_type="255" encryption="255" filename="Id05.geniff" from="304777991"
name="mono_stream" name_bis="mono" salt="90F2F0E854C8C483E32899253AA35BD4" to="359647692">
        <index filename="Id05.geniff" from="359647708" to="364552708"/>
      </stream>
    </dataset>
    <dataset FASTQ_compression="" SAM_compression="BAM" encryption="BLOCK_LEVEL"
name="wgEncodeCshlShortRnaSeqGm12878CellShortAln2.bam" type="SAM">
      <xacml filename="Id05.geniff" from="364556947" to="364570890"/>
      <header filename="Id05.geniff" from="364552824"
name="wgEncodeCshlShortRnaSeqGm12878CellShortAln2.bam" to="364556931"/>
      <stream data_type="255" encryption="255" filename="Id05.geniff" from="364579225"
name="mono_stream" name_bis="mono" salt="37D99B7CD3AF0AB67F0D43DF6BE03702" to="419448926">
        <index filename="Id05.geniff" from="419448942" to="424353942"/>
      </stream>
    </dataset>
    <dataset FASTQ_compression="GZ" SAM_compression="" encryption="ENTIRE"
name="wgEncodeRikenCageK562CytosolPapRawDataRep1.fastq.gz" type="FASTQ">
      <stream data_type="255" encryption="255" filename="Id05.geniff" from="424354144"
name="mono_stream" name_bis="mono" salt="384DFB8ADBE3117D4820BA4455058D62" to="1268791588"/>
    </dataset>
    <dataset FASTQ_compression="GZ" SAM_compression="" encryption="NONE"
name="wgEncodeRikenCageK562CytosolPapRawDataRep2.fastq.gz" type="FASTQ">
      <stream data_type="255" encryption="255" filename="Id05.geniff" from="1268791790"
name="mono_stream" name_bis="mono" salt="4F45CF85A0B6FEF8DC712505A52F1649" to="2113229234"/>
    </dataset>
  </study>
</file>
```

With this file we have all the information needed to place the GENIFF content on a server, and serve it with ease. For example, we know that the information stored in the dataset "test", and the stream "NC_007605" is stored from the position 116493 to 123982, that the governing XACML policy can be found between the positions 55942 and 69885, and we also have access to the salt required to decrypt the stream.

3.6 Installing the GENIFF server

```
#Please download the jar file used in the next line
java -jar Jetty_Geniff-1.0-SNAPSHOT-jar-with-dependencies.jar
```

The GENIFF content is now being served over HTTPs on the port 9998 (and over HTTP on the port 9999). The server certificate is a self-signed one; therefore, it needs to be added to the exceptions. Additionally, the client also uses a certificate to prove his identity and more importantly role. We provide a certificate to test this:

```
curl --cert researcher1_2.pem:holahola -r 0-100 https://localhost:9998/request_data/test/14/index -o output
```

In the previous command we use a certificate proving our fictitious researcher role, and we request the bytes 0 to 100, for the index of the stream 14 (according to our configuration this corresponds to the data aligned to the reference sequence of the chromosome 14) of the study test. Depending on the XACML privacy rule provided during the creation of the GENIFF file, the requests will be granted or denied.

The server supports HTTP HEAD and HTTP GET requests. The first one is intended to recover the sizes of the elements, the second one to obtain the actual information. A word of caution: the server requires the range in byte to be set in the HTTP GET, in order to obtain the entire file simply use the range 0-.

In order to simplify the testing, it is possible to avoid the certificates: for this only use the HTTP connection over 9999. In this case, the server will assume that the role of the client is researcher.

Here after some examples of possible URLs:

http://localhost:9999/request_data/test/14
  Accesses the data stored in stream 14 of dataset test
http://localhost:9999/request_data/test/14/index
  Accesses the index stored in stream 14 of dataset test
http://localhost:9999/request_data/test/14/header
  Accesses the header stored in stream 14 of dataset test
http://localhost:9999/request_data/wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam
  Accesses the data stored in the dataset wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam which is monostream
http://localhost:9999/request_data/wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam/xacml
  Accesses the xacml data stored in the dataset wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam which is mono-stream
http://localhost:9999/request_data/wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam/index
  Accesses the index data stored in the dataset wgEncodeCshlShortRnaSeqGm12878CellShortAln.bam which is mono-stream