Shanoir NG – Import

Software Design Description

Table des matières

[1 Microservice context 1](#_Toc486500170)

[1.1 Microservice presentation 1](#_Toc486500171)

[1.2 Main functionalities 2](#_Toc486500172)

[1.3 Application users 2](#_Toc486500173)

[2 Functional architecture 2](#_Toc486500174)

[2.1 Imports 2](#_Toc486500175)

[2.1.1 Import from DICOM CD/DVD 2](#_Toc486500176)

[2.1.2 Import from PACS 2](#_Toc486500177)

[2.1.3 Import from ShanoirUploader 2](#_Toc486500178)

[2.1.4 Import processed dataset (NifTi) 3](#_Toc486500179)

[2.1.5 Import from files 3](#_Toc486500180)

[2.2 Steps 3](#_Toc486500181)

[2.2.1 Choose modality 3](#_Toc486500182)

[2.2.2 Upload files 3](#_Toc486500183)

[2.2.3 Select series 3](#_Toc486500184)

[2.2.4 Select clinical context 3](#_Toc486500185)

[2.2.5 Describe dataset 3](#_Toc486500186)

[2.2.6 Select/create data processing 3](#_Toc486500187)

[2.2.7 Anonymization 3](#_Toc486500188)

[2.2.8 Nifti conversion 4](#_Toc486500189)

[2.2.9 Select settings for research study 4](#_Toc486500190)

[3 Software/technical Architecture 4](#_Toc486500191)

[3.1 Anonymization 4](#_Toc486500192)

# Microservice context

## Microservice presentation

This microservice is a part of the Shanoir-NG application and is used to import data.

## Main functionalities

Shanoir NG main functionalities are:

* Import from DICOM CD/DVD
* Import from PACS
* Import from ShanoirUploader
* Import processed dataset (NifTi)
* Import from files

## Application users

Target population is:

* Administrators
* Experts
* Users

# Functional architecture

## Imports

### Import from DICOM CD/DVD

Steps of this import are:

* Choose modality
* Upload dataset (DICOM zip)
* Select series
* Select clinical context
* Anonymization
* Nifti conversion
* Select settings for research study

### Import from PACS

Steps of this import are:

* Choose modality
* Define DICOM query and import data
* Select clinical context
* Anonymization
* Nifti conversion
* Select settings for research study

### Import from ShanoirUploader

Steps of this import are:

* Choose modality
* Upload files
* Anonymization
* Select series
* Nifti conversion

### Import processed dataset (NifTi)

Steps of this import are:

* Upload files
* Describe dataset
* Select/create data processing
* Nifti conversion

### Import from files

Import from files allows to upload:

* Processed dataset (NifTi)
* Extra data
* Spectroscopy data

Steps to import processed dataset are:

* Upload files
* Describe dataset
* Select/create data processing
* Nifti conversion

Steps to import extra data are:

* Upload files
* Select clinical context

Steps to import spectroscopy data are:

* Upload files
* Select clinical context

## Steps

### Choose modality

### Upload files

### Select series

A tree is displayed here to let users select series. Here is the structure of the tree:

* Patient
* Study
* Serie
* Serie
* Serie
* …

The tree is an response from the upload MS and is a json object in this format:

{

“patient”: {

“id”: string,

“sex”: string,

“name”: string,

“birthDate”: date,

“study”: {

“name”: string

},

“serie”: [{

“id”: string,

“protocol”: string,

“description”: string,

“seriesDate”: date,

“seriesNumber”: number,

“numberOfImages”: number,

“modality”: string,

“numberOfNonImageObjects”: number,

“images”: [{

“imageId”: number,

“imageUrl”: string

}]

}]

}

### Select clinical context

### Describe dataset

### Select/create data processing

### Anonymization

### Nifti conversion

### Select settings for research study

# Software/technical Architecture

## Anonymization

### Anonymized fields :

See anonymization.xlsx document

### Prerequisite

Zip files should be unzipped.

Check if zip files are unzipped during anonymization step or before.

### Steps

* Anonymization
* Send email
* Create folders

### Technical Specification of Shanoir NG Anonymization

To anonymize a DICOM image, the anonymization service starts with reading the set of DICOM tags to anonymize. The DICOM tags to anonymize are listed in an excel document named anonymization.xlsx. Thus, the anonymization service parses the excel file, reads the list of tags and the operation to do to anonymize each tag based on the anonymization profile that should be applied (figure 2). Then it reads the DICOM tags of the image and anonymizes each tag mentioned in the excel file. For so, it follows several steps:

* Treats private tags: assume that a dicom tag is represented by (gggg, eeee) form. All private tags recognized when the “gggg” part is odd should be anonymized (see figure 1).
* Treats public tags: public tags specified in the excel file have to be anonymized. We distinguish DigitalSignaturesSequence tag and DataSetTrailingPadding tag that can not be converted to “int” using “Integer.decode”, so we treat them differently (see figure 1).
* Treats “xx” tags: some tags belonging to the same interval have to be anonymized the same way. For instance, curve data recognized when the tag is beginning with “50” should be removed. Thus all tags (50xx,xxxx) should be removed after anonymization.

In this step, we treat three particular use cases which are: (50xx,xxxx), (60xx,4000) and (60xx,3000) tags (see figure 1).

For each tag, there is an appropriate way to anonymize it: some of them should be removed, others should be replaced with a zero length value …etc . According to the anonymization profile, a different value will replace the original one as described by figures 3 and 4. Moreover, to replace the tag’s original value, the anonymization service have to respect its VR as described by figure 3.

The whole anonymization process is described by figure 1.

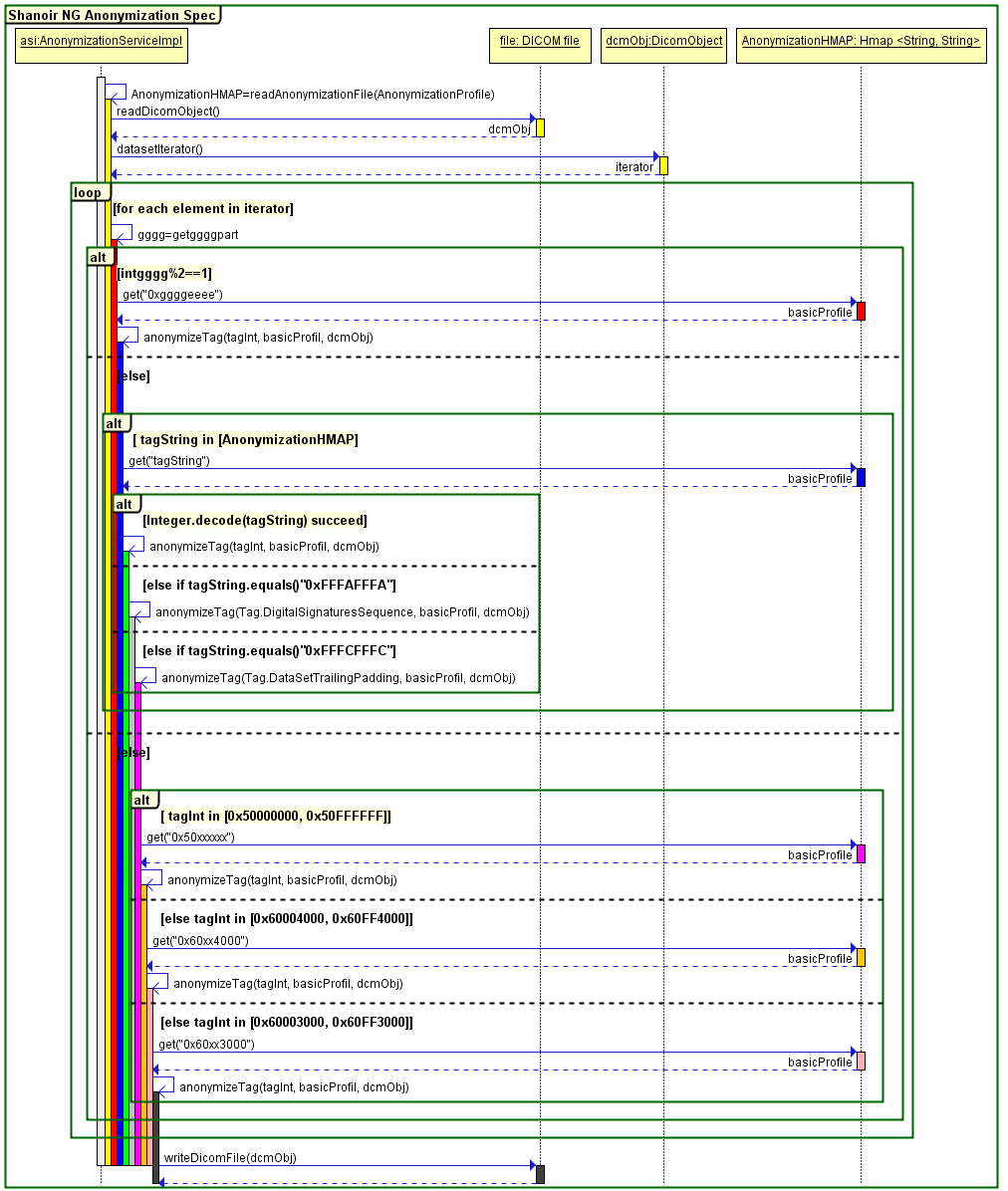


Figure 1. DICOM image anonymization process

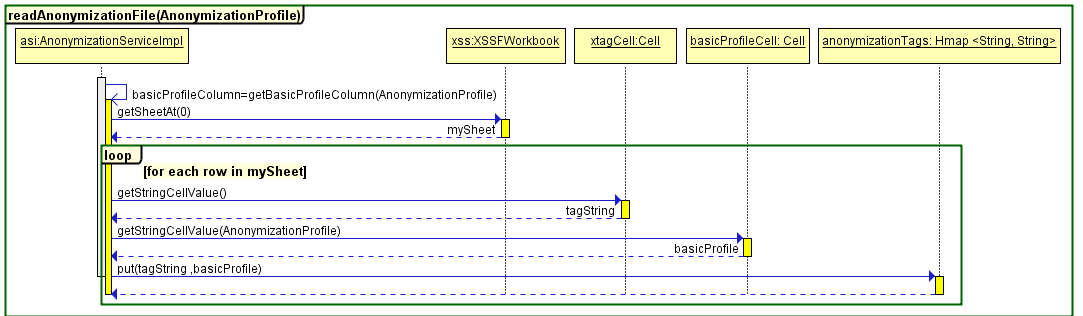


Figure 2. Excel anonymization tags file reading

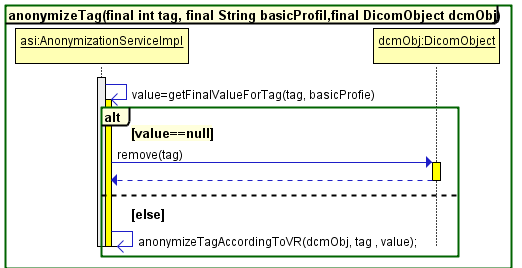


Figure 3. DICOM tag anonymization process

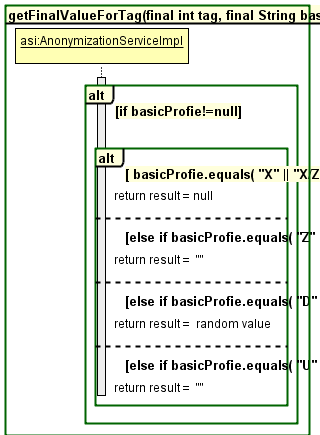


Figure 4. Anonymized tag value based on profile

#### UIDs anonymization

##### UIDs rules

To anonymize UIDs, they should be replaced by non-zero length UIDs that are internally consistent within a set of Instances.

There are several rules to be followed when making UIDs:

* Total length must be <= 64 characters, including the stops
* Must contain **only** digits 0-9 and full stops
* Each numeric "component" (between stops) must be a valid and unambiguous integer number, and so must **not** have a leading zero (unless the whole component is zero)
* Must be guaranteed to be unique - this means:
  + It must be derived from a proper official root under **your** sole control.
  + It must **not** be created by appending digits (however special you consider the combination!) to **someone else's** UID.
  + In particular, series UIDs for secondary capture images, KIN objects etc. must **not** be created as derivatives of the Study UID (unless you own that root!)
* Related to the above, there is **no** expectation or requirement that the Study UID, Series UID and Instance UID for images should be derived from the same root (though in practice, Series UID and Instance UID normally are, as both must be generated internally by the equipment which generates the images)
* Date and Time are useful for generating UIDs, but only if:
  + Each machine has a unique root (normally your company UID root + a machine specific suffix such as a serial number
  + If it is possible for UIDs to be generated at > 1 per second, then a sequential counter should also be used
  + if on a multi-threaded machine, then the thread ID or a properly interlocked counter are needed to prevent 2 applications or 2 threads in the same application from generating identical UIDs simultaneously.
  + Do not use time on its own - it is too easy to end up with a leading zero 0 - e.g. 20060724.093017 use instead 20060724093017

##### New UIDs generation

The anonymized UIDs are generated using the java source code of PixelMed Toolkit.

PixelMed toolkit is a stand-alone DICOM toolkit that implements code for reading and creating DICOM data, DICOM network and file support, a database of DICOM objects, support for display of directories, images, reports and spectra, and DICOM object validation.

The toolkit is a completely new implementation, which does not depend on any other DICOM tools, commercial or free. It does make use of other freely available pure Java tools for compression and XML and database support.

The toolkit is open source and made freely available for both non-commercial and commercial use, under [BSD License](http://www.opensource.org/licenses/bsd-license.php) terms.

##### DCM4CHEE constraints to correctly store DICOM data

###### DCM4CHEE constraints to accept to store data

SOPInstanceUID is a unique identifier for the DICOM dataset. **MediaStorageSOPInstanceUID should have the same value as in the SOPInstanceUID**. The only difference is that MediaStorageSOPInstanceUID is stored in the meta-information header of the DICOM file so that it becomes easy to read and fast to access while the SOPInstanceUID is stored in the dataset.

###### DCM4CHEE constraints to correctly visualize data

**Patient Identification**

If the issuer of patient id is null, dcm4chee attempts to identify the patient by name+id+dob.

dob: date of birth

The Patient Matching code has problem in StoreScp if the patient name in the database is incomplete and is "truncated", e.g. the patient has no middle name and the patient name in the database is 'SURNAME^GIVEN\_NAME' rather than 'SURNAME^GIVEN\_NAME^^^'.

**Study/Serie** **Identification**

Study Instance UID Attribute is a unique identifier for study, so all images acquired for the same study should have the same Study Instance UID.

Series Instance UID Attribute is a unique identifier for the Series that is part of the Study identified in Study Instance UID (0020,000D). So all images acquired for the same series should have the same Series Instance UID.

### Email

Send email

Define email content and when it is sent.

### New folders

TODO