TranSMART-XNAT-importer-plugin Manual

Erwin Vast (e.vast@erasmusmc.nl), Stefan Klein
Biomedical Imaging Group Rotterdam
Erasmus MC, University Medical Center
Rotterdam, the Netherlands

1. Introduction

Medical imaging (MRI, CT and Ultrasound) is more and more included in multi-center clinical studies. Quantitative imaging biomarkers, such as brain matter volumes, can be derived from the images to aid in the diagnoses, to serve as surrogate endpoints for clinical trials, or for epidemiological analysis. Often, imaging biomarkers are analyzed with other clinical data such as disease status, genetics or age to provide more accurate diagnoses (see Figure 1) or to verify hypotheses. Therefore, what is needed is a user-friendly data infrastructure to support centralized correlative analysis between medical image-derived and other clinical data.

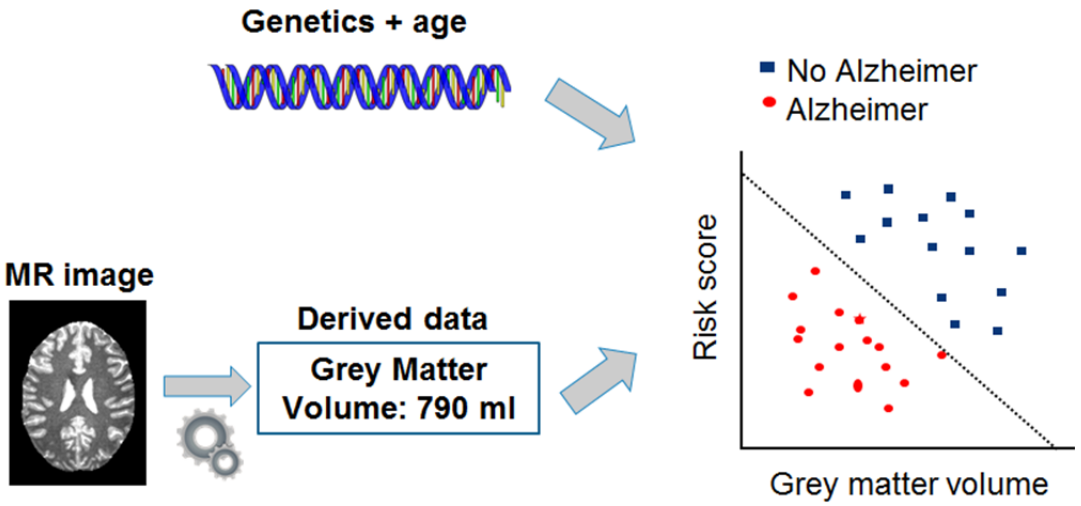

Figure 1: Medical image-derived data combined with genetic and demographic data to improve diagnosis.

TranSMART (http://transmartfoundation.org) is a data integration and browsing platform for central statistical analysis in translational research.

XNAT (www.xnat.org) is a framework to store, share and manage medical imaging data. This imaging archive can be connected to a computing cluster for further image analyses. After analyses of the images, the analysis results (e.g. grey matter volumes) can be stored back in XNAT.

To correlate the image-derived results with other medical data, such as demographic or genetic data, it would be useful to import the XNAT results in TranSMART. To this end, we developed the TranSMART-XNAT-Importer plugin (hereafter called plugin). To configure which XNAT image-derived data is imported in TranSMART, an administrator can create a coupling configuration in the online administration panel and start the import process. The plugin uses the default ETL Importer for clinical data to import the XNAT results in TranSMART. This process is also depicted in Figure 2.

To simplify the amount of work for an administrator, no access to the machine is required to import XNAT data. The plugin is setup in such a way that after installation, the configuration and import process can all be managed from the TranSMART web interface. In particular, we created a new panel in the administration interface to manage the import. This part of the document explains the plugin in more detail. Note that the plugin and this documentation are most useful for TranSMART administrators as full administrative privileges for TranSMART are required.

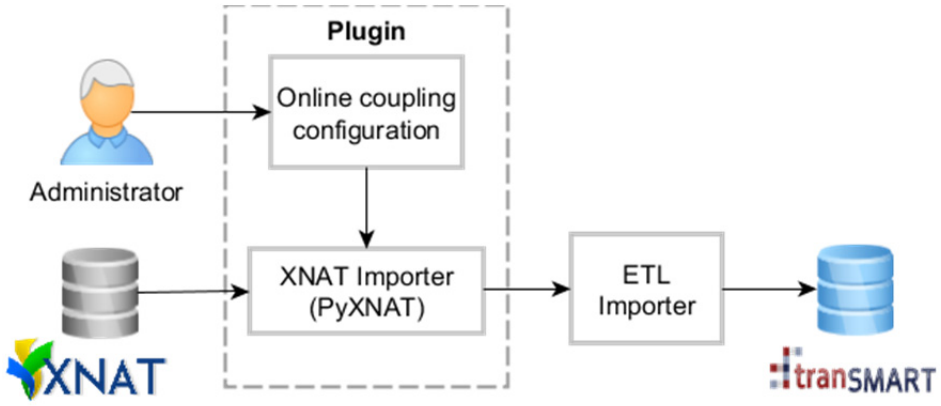

Figure 2: Architectural overview of importing XNAT data in TranSMART. An administrator configures the import coupling.

1.2 Installation

The TranSMART-XNAT-Importer can be installed by following the below steps. All steps below should be run by the user that also built TranSMART, because this user has the local Maven repository and cache. This guide assumes that a TranSMART installation with all pre-requisites already exist on the server. If not, please follow the instructions at the TranSMART wiki: https://wiki.transmartfoundation.org/display/TSMTGPL/Installing+tranSMART+from+Source. Furthermore, the TranSMART version should at least be of version 24ab7e4 or the changes in that commit should be manually updated, if a version update is not preferred.

1. Install Python (2.7.*) if not already installed. See: https://www.python.org/downloads/.
2. Install PyXNAT:
    a. Install httplib2: easy_install httplib2 or apt-get install python-httplib2
    b. Install lxml: easy_install lxml or apt-get install python-lxml
    c. Download latest version from https://github.com/pyxnat/pyxnat (the normal download version may not be working with the latest XNAT). Extract if needed.
    d. Install PyXNAT with (from within the download directory): python setup.py install.
3. Clone the Git repository to a preferred directory with the following commands: 'git clone https://github.com/evast/transmart-xnat-importer-plugin'
4. Go to the created directory: 'cd transmart-xnat-importer-plugin'
5. Add the database tables by going to 'cd grails-app/ddl/postgres/' and run 'psql -U postgres -d transmart -f install-plugin.sql'. The syntax (mostly regarding -U postgres) depends on the used authentication technique of psql. Please see Postgress documentation for more information. When completed, go back to the plugin folder root with 'cd ../../../'.

6. Build the plugin source code with: ‘grails compile’ and create the module in the local Maven repository with ‘grails maven-install’.
7. Add the plugin and configuration:
    a. [editor] [transmartApp]/grails-app/conf/Config.groovy
    b. At the end, add*:
       org.transmart.data.location = ["file:${baseDir}"]
       org.transmart.xnatImporterEnabled = true
    c. Save the file
8. Include the plugin in the transmartApp:
    a. [editor] [transmartApp]/grails-app/conf/BuildConfig.groovy
    b. Within grails.project.dependency.resolution { …} / plugins { … } scope / if (!dm) { …} scope, add:
       runtime ':transmart-xnat-importer:0.1'
    c. Save the file
9. Rebuild and restart transmartApp in [transmartApp] with grails compile && grails run-app.

The new admin panel should now be present in the administration panel.

* = We assume that transmart-data is in the same directory as transmartApp. If this is not the case, change ${baseDir} to the transmart-data directory.

2. User guide

An administrator can create an importing configuration and import the data. This part of the document explains both processes in more detail, with an introduction of the XNAT data structure.

2.1 Datastructure

We assume the reader is already familiar with XNAT and how to import image-derived data. To learn more about these two techniques, please look at the mailing list:

https://groups.google.com/forum/#!searchin/xnat_discussion/custom$20variables

There are two techniques to store image-derived results in XNAT:

- As custom variable(s).
- As variable(s) in a custom datatype.

The plugin supports both techniques. In XNAT both subject as session data can contain custom variables and custom datatypes. A total of four locations are thus possible to store image-derived data, which we abbreviate in the plugin as:

- Subject: variable in the subject datatype.
- SubjectVariable: custom variable in the session.
- Session: variable in the session datatype.
- SessionVariable: custom variable in the session.

These four locations are called *datatype* in the plugin. To locate the exact variable in XNAT for datatype variables, we use the REST interface structure as *XNAT (REST) URL*. To be precise, the importer has as input a list of *XPath*'s which link to a specific variable in XNAT. For example, the patientname can be retrieved by the link: xnat:imageSessionData/dcmpatientname. The XNAT REST webpage https://wiki.xnat.org/display/XNAT/XNAT+REST+XML+Path+Shortcuts lists all possible REST addresses. This page can be used to lookup the address for an XNAT variable when configuring the import coupling. For custom variables, only the exact name of the custom variable is required as the XNAT (REST) URL. Note that capital and special characters are not supported for custom variables. In TranSMART, the target name of the variable is given by *name*.

In the coupling configuration we only give the paths of the variables, but do not specify for which patient and imaging session the data is requested. This is not needed, as the plugin automatically retrieves all subjects from the given XNAT project. It then iterates over these subjects to find all sessions. It is not possible to retrieve data for a specific patient and imaging session. The reason for this choice is that it is expected that the plugin is used to retrieve large datasets (e.g. epidemiological studies, clinical trials) where it is not feasible to enter each patient and session number manually.

2.2 Creating an import configuration in TranSMART

To import a new XNAT clinical dataset, go to the TranSMART administration panel and select Create Configuration below the header Import XNAT Clinical Data. Enter the project details, also see Figure 3, and click Create.

Figure 3: Creating an import configuration.

After creation, an empty list will be shown. This page is three-fold:

1. List of variables: show a list of all variables that are imported in TranSMART.
2. A form to add a variable.
3. Help with more information about the import process.

In this page, the user can add each XNAT variable by entering the details in the Add variable form, and clicking Create. Then the variable will be added to the list. The user can continue this process to add all

variables. Figure 4 shows an example configuration list with variables. The SUBJ_ID variable is mandatory for all import configuration. A variable can also be deleted by clicking the delete button. The complete configuration can also be download by clicking Download as XML. Currently there is not yet an import option to import this configuration again, so this XML is only for viewing and debugging purposes.

The variable list is automatically saved when a variable is added. After completing the coupling configuration, the user can go to the Configuration List in the left menu. This panel shows a list of all configurations; the user can also change configurations if required. After clicking on Show, there are four options:

1. Edit: User can edit the import configuration shown above the buttons.
2. Edit coupling: User can edit the coupling configuration (XNAT variables).
3. Import data: Imports the data, explained in next paragraph.
4. Delete: deletes this configuration, but not the already imported data.

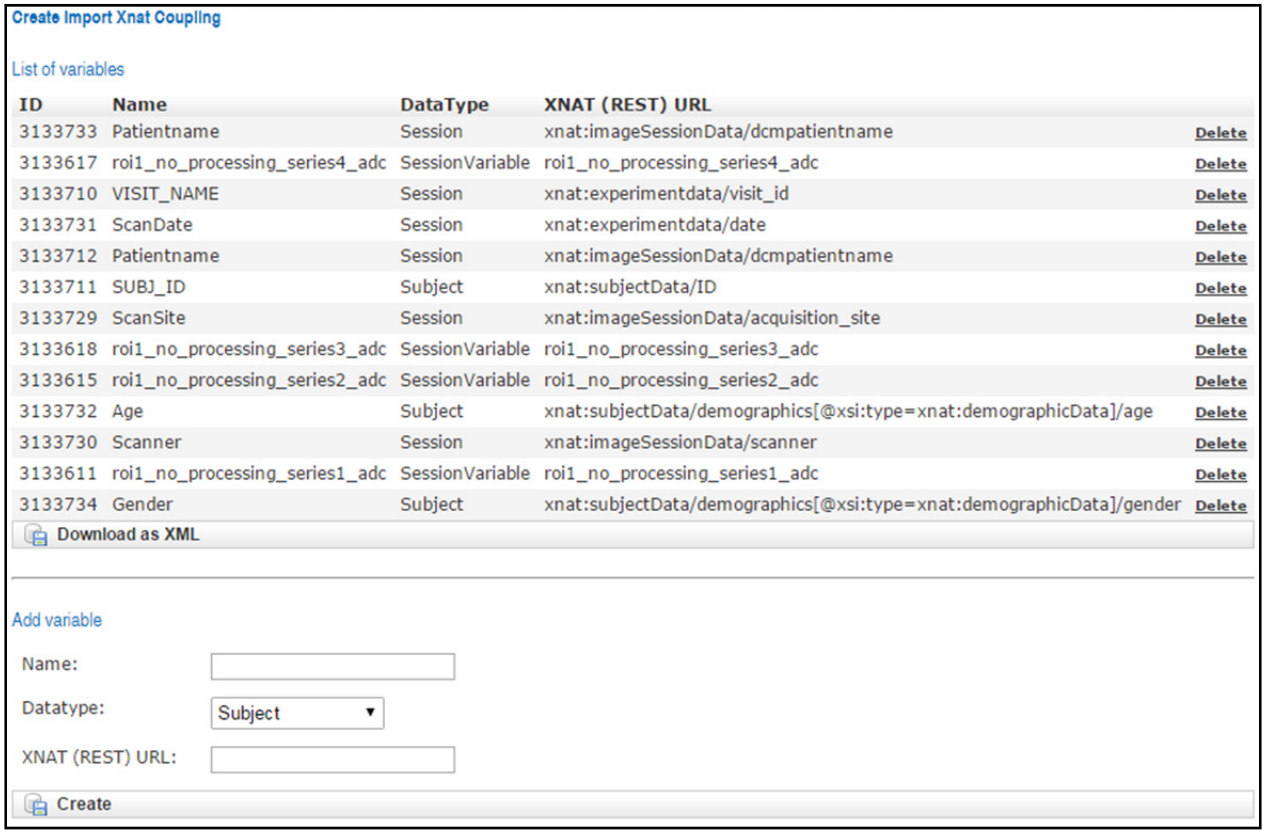

Figure 4: Example import configuration.

2.3 Importing XNAT clinical data in TranSMART

In the menu, click on Configuration List, select the appropriate configuration, click show and click on Import Data. Here the password will be asked for the XNAT user. Note this is the user that is filled in the configuration described in paragraph 2.2. For security reasons, the password is not stored in TranSMART

and will therefore be asked when opening a connection with XNAT to import the data. After entering the password, click Import data. Then data will then be imported, also see Figure 5.

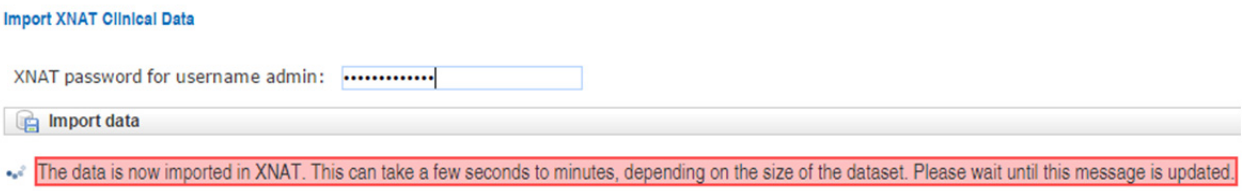

Figure 5: Message after starting the import.

As the message shows, the import can take a few seconds to minutes or even hours for large datasets. Do not close the page during this import process. The import is completed when the red bar is replaced with a blue bar showing success, see Figure 6.

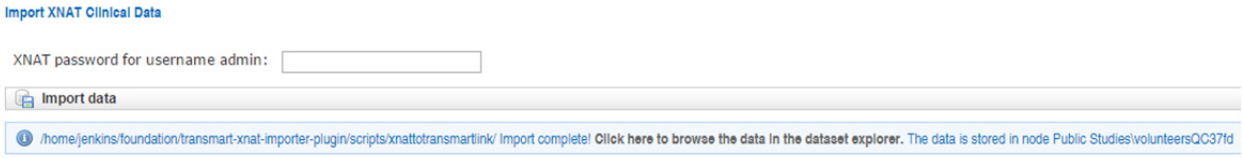

Figure 6: Message after completing the import.

When clicking on the link, it will open the TranSMART browser in a new page. Scroll down to the created project node (also shown in the import message at the end as shown in Figure 6). Here the XNAT clinical data is shown. See Figure 7 for an example.

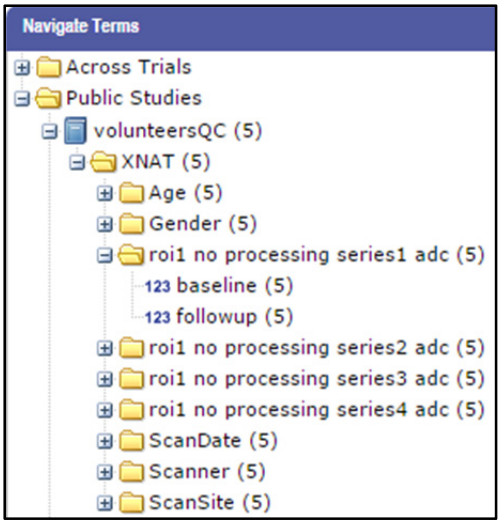

Figure 7: XNAT data shown in TranSMART explorer.

3. Technical documentation

For developers, this part of the document explains the plugin design and scripts in more detail.

3.1 Architecture

The plugin architecture consists of four components, also depicted in Figure 4:

- XNAT: The imaging database.
- XNAT Importer: Python application that uses a PyXNAT (https://pythonhosted.org/pyxnat/), a Python wrapper for the XNAT REST calls. This simplifies the communication with the XNAT server significantly. The Importer downloads the image-derived data and writes them to the TranSMART import files. These files are the data file (.tmd extension) and the mapping file (.tmm extension).
- Online configuration: An administrator can create an online coupling configuration via the TranSMART administration panel. This configuration describes which XNAT data is stored where in TranSMART. Multiple configurations can be made and stored in the TranSMART (Postgress) database. The import process can also be started via the configuration panel.
- ETL Importer: The TranSMART ETL Importer (https://wiki.transmartfoundation.org/display/TSMTGPL/Data+ETL) for clinical data is used to import the XNAT data from the data and mapping files, previously created in step 2, into TranSMART. No changes were required for the ETL importer. When the import process is completed, the data is available in the TranSMART data explorer.

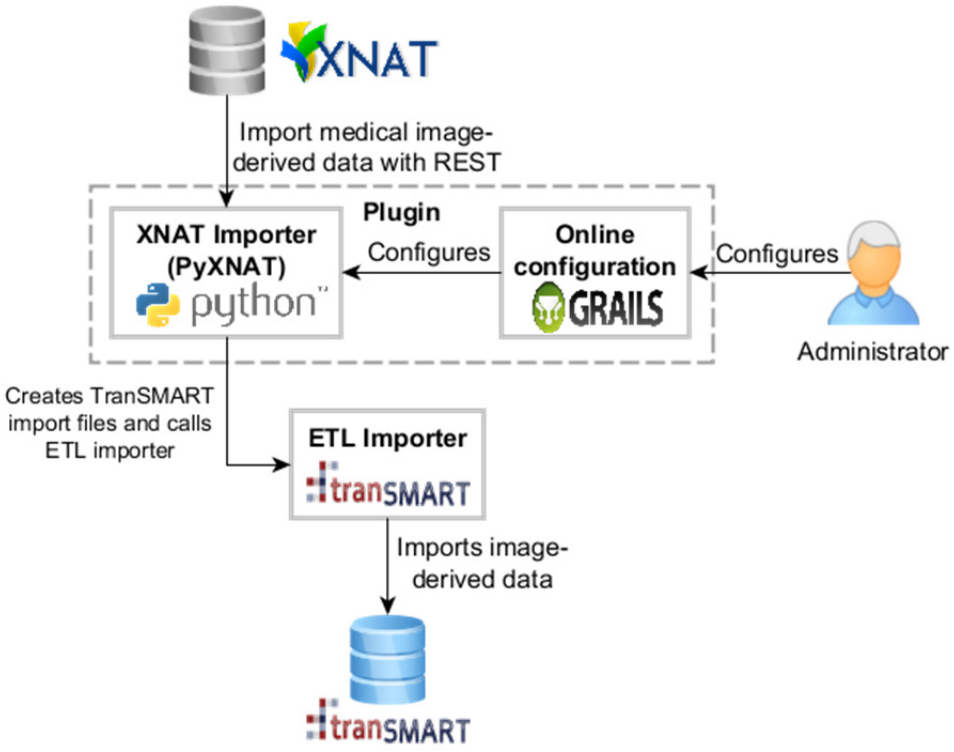

Figure 3: Architecture to import image-derived data to TranSMART

3.2 Import process

The following steps are performed:

1. The user is asked via the website to enter his XNAT password.
2. The import configuration is loaded from the database.
3. The configuration is written to an XML file stored in transmart-xnat-importer-plugin/scripts.
4. The Python script transmart-xnat-importer-plugin/scripts /download.py is called with the required arguments (explained below).
5. Waiting for the script to complete.
6. Report success or error message to the user via the website.

3.3 Pyxnat script

The Python script downloads the XNAT image-derived data with the PyXNAT library. The script file can be found at {importer-directory}/scripts/xnattotransmartlink/download.py and has 7 inputs:

1. The XNAT url or ip address
2. The XNAT username
3. The XNAT password
4. The XNAT project identifier (note that this may not be same as the projectname. Please check in XNAT within the project the projectidentifier).
5. The TranSMART destination node

6. The Kettle directory
7. The data target directory (location of the target *.tmd and *.tmm files)

The Python script is automatically called from the importer that is started via the web interface. The Python script reads the import configuration from an XML file. This is both of historic reasons and for simplicity, as no database access is then required for the Python script. PyXNAT (https://pythonhosted.org/pyxnat/) is the python library for XNAT, which is used by the Python script. This library is a wrapper for the XNAT REST interface, which simplifying the communication with XNAT.

The Python script has the following steps:

1. Connect to the XNAT server with the given address, username and password.
2. Load all subjects of the given project.
3. Load all visitnames of the given project.
4. For all subjects, read all variables from the XML and add to a map.
5. Create mapping list based on the XML from the attributes.
6. Create the data file.
7. Create the mapping file.
8. Run the Kettle job.
9. Return the result (e.g. success or error messages) to the website application.

The XML is written to {importer-directory}/scripts/xnattotransmartlink/[projectname].xml where projectname is the name of the XNAT project. This file is not deleted and can be reviewed for error or debugging purposes.

3.4 Database design

The XNAT import configuration is stored in two PostgreSQL database tables, also shown in Figure 5. The table ImportXnatConfiguration stores the general information regarding the XNAT server: a server name, description, webaddres, account username, XNAT project, TranSMART node and the internal version and id number. Note that for security reasons, the XNAT password belonging to the username is not stored in the database. The administrator has to give his password via the webinterface and is not stored in the database for later usage. The ImportXnatVariable stores all references to XNAT variables, with the id, TranSMART name, XNAT datatype and XNAT (REST) URL. A foreign key configuration_id referes to the ImportXnatConfigurationId. There is a one-to-many relationship between the configuration and variable. Each variable belongs to one configuration and one configuration can contain multiple variables. One variable cannot belong to multiple configurations, however, for simplicity.

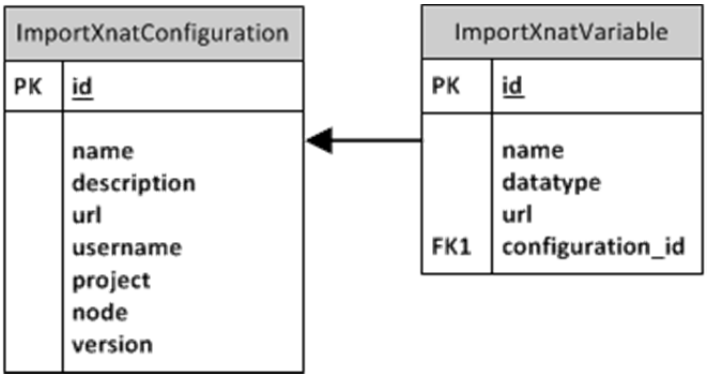

Figure 4: Database diagram

The SQL scripts to create the database stored in: transmart-xnat-importer-plugin/grails-app/ddl/postgres. The grails domain scripts are located in: transmart-xnat-importer-plugin/grails-app/domain/org/tranSMART/searchapp.

3.5 Configuration XML

It is possible to download the configuration to an XML file with a standardized structure. In the configuration coupling panel, click at the bottom of the table on *Download XML*. An example XML is shown in Listing 1. All variables are stored in the subset project/variables. Each variable has three attributes:

1. The TranSMART ontology name
2. The XNAT datatype
3. The XNAT REST URL

See Section 2.3 for the description of each of these variables. This configuration XML was the way to store the configuration before the configuration was stored in the Postgress database. Currently it is not possible to import an XML configuration. The XML is still used to send the configuration from the web application to the Python script, so that the Python script does not have to access the database, which is more difficult to implement.

```
<project name="test">
  <variables>
    <variable name="SUBJ_ID" dataType="subject" url="xnat:subjectData/ID" />
    <variable name="VISIT_NAME" dataType="session" url="xnat:experimentdata/visit_id" />
    <variable name="Age" dataType="subject"
url="xnat:subjectData/demographics[@xsi:type=xnat:demographicData]/age" />
    <variable name="Gender" dataType="subject"
url="xnat:subjectData/demographics[@xsi:type=xnat:demographicData]/gender" />
    <variable name="Scanner" dataType="session" url="xnat:imageSessionData/scanner" />
    <variable name="PatientName" dataType="session" url="xnat:imageSessionData/dcmpatientname"
/>
    <variable name="ScanDate" dataType="session" url="xnat:experimentData/date" />
    <variable name="ScanTime" dataType="session" url="xnat:experimentData/time" />
    <variable name="ScanSite" dataType="session" url="xnat:imageSessionData/acquisition_site" />
```

```
    <variable name="BrainVolume" dataType="sessionvariable" url="brainvolume" />
    <variable name="BrainWeight" dataType="sessionvariable" url="brainweight" />
    <variable name="BrainDimensions" dataType="sessionvariable" url="braindimensions" />
    <variable name="Braintype" dataType="subjectvariable" url="braintype" />
    <variable name="Braincolour" dataType="subjectvariable" url="braincolour" />
  </variables>
</project>
```

Listing 1

3.6 Calling ETL script

The Pyxnat script calls the ETL Importer for clinical data automatically via an intermediate bash file, located at: transmart-xnat-importer-plugin/scripts and is shown below.

```
#!/bin/bash
export KETTLE_HOME="/home/jenkins/transmart-data/env/tranSMART-ETL/Postgres/GPL-
1.0/Kettle/Kettle-ETL/"

./kitchen.sh \
-norep=Y \
-file="/home/jenkins/transmart-data/env/tranSMART-ETL/Kettle-GPL/Kettle-
ETL/create_clinical_data.kjb" \
-log=load_clinical_data.log \
-param:LOAD_TYPE=I \
-param:COLUMN_MAP_FILE=./xnat.tmm \
-param:DATA_LOCATION=./xnattotransmartlink \
-param:TOP_NODE="\\Public Studies\\$1\\" \
-param:STUDY_ID=$1 \
-param:SORT_DIR=/home/transmart/ETL \
-logging=Rowlevel \
-level=Rowlevel \
> command.out
```

This script sets the correct data directories to read the data and mapping files, several TranSMART arguments and calls the TranSMART kitchen.sh script. The last script calls the ETL Importer.