Detailed Functional Specification of Study, Study Card, Subject & MR Examination implementation in Shanoir Uploader

Shanoir version ???

ShanoirUploader version ??

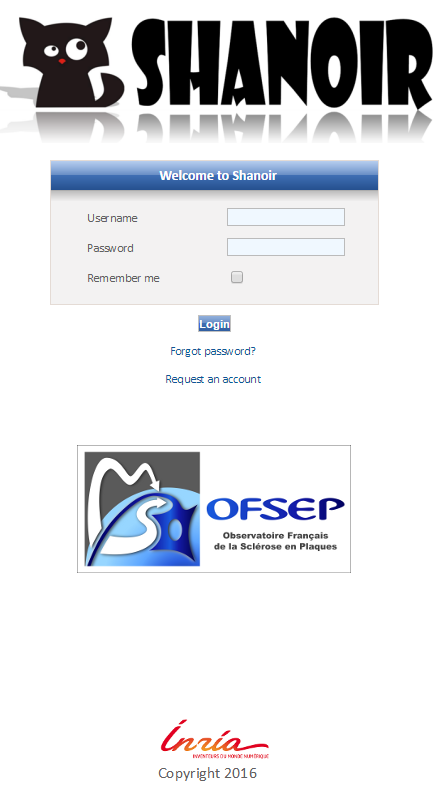


Table of contents

[Context 3](#_Toc484095611)

[Goal 3](#_Toc484095612)

[Version 3](#_Toc484095613)

[Conventions 3](#_Toc484095614)

[Abbreviations 3](#_Toc484095615)

[Shanoir Uploader description 4](#_Toc484095616)

[What is the ShanoirUploader? 4](#_Toc484095617)

[Workflows 5](#_Toc484095618)

[Technical solution 5](#_Toc484095619)

[Shanoir uploader implementation of the study, study Card, Subject and MR Examination selection 5](#_Toc484095620)

[MVC Pattern 5](#_Toc484095621)

[MVC Pattern Implementation in Shanoir uploader 6](#_Toc484095622)

[Services (when automatic import is enabled) 7](#_Toc484095623)

[Study/ Study cards lists 7](#_Toc484095624)

[Subject 7](#_Toc484095625)

[MR Examination 7](#_Toc484095626)

# Context

## Goal

This technical specification collects all information regarding the data import and export of the Shanoir platform, from a technical point of view. It helps to understand already existing functions better and to document them.

## Version

This document is version 1.0.

## Conventions

* Font types:
* Courier New is used for source code.
* *Italic letters* are used for file paths and file names.
* Synonyms:
* DICOM server == PACS (we try to use only DICOM server here)

## Abbreviations

**PACS** Picture Archiving and Communication System

**SU** ShanoirUploader

**ShS** Shanoir Server

# Shanoir Uploader description

## What is the ShanoirUploader?

ShanoirUploader is a Java desktop application that transfers data securely between a PACS and a Shanoir server instance (e.g., within a hospital). It uses a DICOM query/retrieve connection to search and download images from a local PACS. After the retrieval of the DICOM files, they are anonymized locally and uploaded to the Shanoir server.

The primary goals of that application are to enable mass data transfers between different remote server instances and therefore reduce the waiting time of the users, when importing data into Shanoir. Most of the time during import is spent during data transfers.

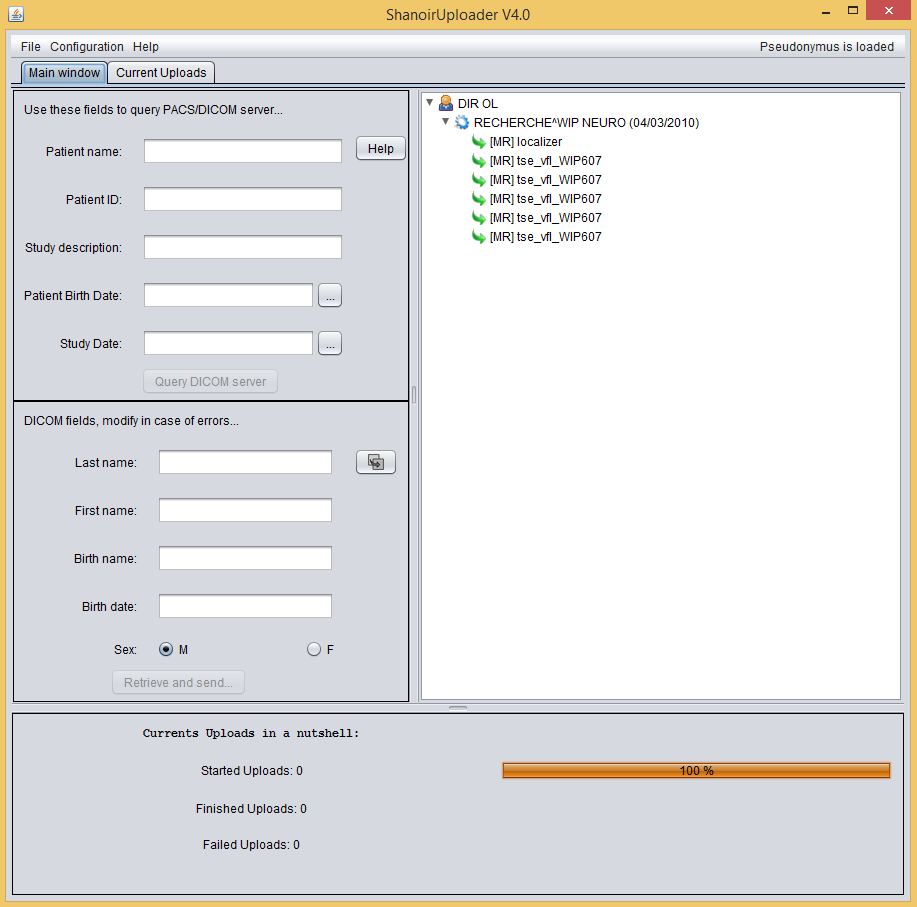


Figure 1: Screenshot of the ShanoirUploader application

## Workflow

This chapter describes the user workflows using the Shanoir Uploader.

Installation and configuration of Shanoir Uploader

Query DICOM server

Open ShanoirUploader

Auto import enable

Auto import disable

Select Study, Study card,

Subject and MR examination

Upload data to the Shanoir server

Upload data to the Shanoir server.  
*Note: import is done automatically*

Go to the Shanoir server

Complete the information on the study, examination and subject

Finish the import

**Figure 2: General use case of the import with SU and ShS**

# Technical solution

## Shanoir uploader implementation of the study, study Card, Subject and MR Examination selection

### MVC Pattern

This section describes the global picture of the MVC pattern implemented in Shanoir uploader.

In order to implement the selection of the study, study card, subject and MR examination, a MVC pattern has been set up. The following figure shows the communication between the different modules.

Update

View

Controller

Model

Event

Notify

Report Errors

**Figure 3: MVC Pattern**

### MVC Pattern Implementation in Shanoir uploader

This section describes the detailed implementation of MVC pattern.

Update

View: ImportDetailsDialog

Controller: ImportDetailsListener

Model : ExportData

Event

Notify

Report Errors

WSDL

Rest

Auth Token

Shanoir Old

Shanoir NG

keycloak

**Figure 4: MVC Pattern implementation in Shanoir uploader**

The view ImportDetailsDialog.java is located in the package org.shanoir.uploader.gui.

The design pattern observer is set between the view and the model. The view observes the model and when the model changes, the view is updated accordingly.

The model ExportData.java is located in the package org.shanoir.uploader.model.

The model contains the input and output data. When the model changes, it notifies the observers (i.e. the view).

The controller ImportDetailsListener.java is located in the package org.shanoir.uploader.action.

The controller catches the events from the view and update the model accordingly. In case of error, the view is directly notified. The controller handles the connection to the different services needed to retrieve the subject, the study and the study cards.

## Services (when automatic import is enabled)

In order work properly, the controller needs to retrieve the following information from different referential and with different protocols:

* Study/ Study cards lists,
* Subject (if it exists, else the subject will be created automatically during the export)
* MR Examination

### Study/ Study cards lists

In order to retrieve the study and study cards, the application queries the ShanoirNG server through a rest service {SPECIFY URL}. In order to access the rest service, the controller needs to obtain first an authentication token though Keycloak client. Once the token is granted, the controller query ShanoirNG rest service.

### Subject

In order to identify the subject linked to the selected subject in the Dicom server, the controller queries Shanoir old through a soap request {SPECIFY URL}.

Note: If the subject does not exists in Shanoir old, the user is asked to fill the subject fields accordingly. The subject is created, if and only if all the mandatory fields are filled by the user, once the user click on the export button. Another soap service is then used in order to create the subject in the Shanoir old referential. {SPECIFY URL}.

### MR Examination

{TO BE COMPLETED}