GNU Health Radiology

Wei Zhao

Jul 23, 2024

CONTENTS:

CHAPTER

ONE

INTRODUCTION

The aim of this project is to connect the PACS Orthanc server to GNU Health, allowing GNU Health to access patient radiology data. The new module has been developed to configure the Orthanc server and to manage the image data. Since GNU Health and image data have different structures, it's important to create a mapping structure that seamlessly connects GNU Health patient data with Orthanc image data for easy handling.

This document describes the integration of the Orthanc PACS with GNU Health. We'll examine the structure of Tryton, the Enterprise Resource Planning (ERP) system used by GNU Health, and its modules. The focus will be on GNU Health itself, its general structure and its main modules. In particular, we'll explain the functionality of the module 'health_radilogy', which provide the radiology capabilities of GNU Health. This includes a brief overview of the classes, methods and functions within these modules. Finally, the document provides instructions on how to install and run GNU Health.

1.1 Authors and acknowledgment

UCLouvain ICTEAM/INGI: Wei Zhao, Professor Sébastien Jodogne.

GNU Health Community: Thanks to Luis Falcon, Gerald Wiese, Axel Braun for their support and feedback during the deisgn.

1.2 License

CHAPTER

TWO

TRYTON

Tryton[1] is a free software platform designed to help businesses of all sizes manage their operations effectively. It offers a modular structure that allows companies to select and integrate specific modules tailored to their needs. These modules cover various business functions such as accounting, inventory management, sales, purchasing and human resources. Users can customise Tryton to suit their unique workflows, providing flexibility not found in predefined systems.

As open source, Tryton's source code is freely available for modification and redistribution, fostering collaboration and innovation within its community. It's designed for businesses of all sizes, from start-ups to large enterprises, and can handle different levels of complexity and transaction volumes. In addition, Tryton works on many different operating systems, as Python and GTK, a toolkit used for the Tryton graphical user interface (GUI), are both available on many different platforms. Security is a priority, with features such as user authentication, access control and data encryption ensuring the protection of sensitive information and compliance with data protection regulations.

Overall, Tryton provides a robust and customisable platform for businesses to efficiently manage their operations, drive growth and achieve their goals.

2.1 What are Modules in Tryton?

Modules in Tryton are units of functionality that encapsulate specific features or business logic. They serve as building blocks for constructing comprehensive business applications tailored to specific requirements. Each module focuses on a particular aspect of business operations, such as accounting, inventory management, or customer relationship management (CRM).

> The first core module is the `ir` module. It provides many of the basic functionalities needed for creating a module, including:
>
> - **User Interface (UI)**: Tryton allows different modules to add their user interface elements easily using simple Extensible Markup Language (XML) files. This includes basic views like email forms.
> - **Translations**: Tryton simplifies the process of translating the software into different languages. Modules can create translation files containing translated text for each language, making it accessible to a wider audience.
> - **Task Scheduling**: Tryton enables the automation of tasks by allowing modules to create scheduled tasks. Users can configure these tasks, such as changing the frequency of execution, to suit their needs.

Another important module called `res` that manages user-related tasks. This module contains the `User` Structured Query Language (SQL) model that stores and manages user data. It performs tasks such as creating or removing users, managing passwords, user preferences and more. It also allows other modules to create customised user groups with specific permissions.

[1] https://www.tryton.org/

2.2 Module Structure

Tryton modules follow a well-defined structure to ensure consistency and maintainability. At a high level, a typical Tryton module consists of the following components:

- **Metadata:** Each module contains metadata describing its purpose, dependencies and other relevant information. These files include important ones like `__init__.py`, which tells Python that the folder contains a module, and other files written in Python and XML languages. So Tryton modules are structured and set up in a way that makes them easy to work with. Tryton manages the modules using configuration files. A `setup.py` file, which is executed when the module is installed, and a main configuration file called `tryton.cfg`, which contains important details about the modules.
- **Model Classes:** Models in Tryton represent the data structures used in the system. These classes define the structure of database tables and the relationships between them. Each module can introduce one or more custom models to meet specific business requirements.
- **Views:** Views define how data is presented to users within the application's user interface. Tryton supports several types of views, including form views, tree views, and search views. Modules can include XML files that specify the layout and behaviour of these views.
- **Menus and Actions:** Menus and actions provide navigation paths and trigger specific operations within the application. Modules can define new menu items and associate them with corresponding actions to facilitate user interaction.
- **Business Logic:** Modules encapsulate business logic in the form of Python code. This code implements various functionalities such as data validation, computation, and workflow automation. Business logic is typically organized into methods within model classes or separate Python modules.
- **Security Rules:** Tryton includes a robust security model that allows fine-grained control over access to data and functionalities. Modules can define security rules to restrict certain operations based on user roles and permissions.
- **Translations:** To support internationalization, modules can include translation files for different languages. These files contain localized versions of text strings used within the module's user interface.

CHAPTER

THREE

GNU HEALTH

GNU Health[1] is a free and open source health care system developed by GNU Solidario. GNU Solidario is a group that wants to help people get better medical care. They also care about animals. They started GNU Health in 2008 to improve healthcare, especially in places where it's hard to get help. It's become a big project. In 2011 it won a Good Project award from the Free Software Foundation. Anyone can help improve GNU Health because it's open source. Many people are working on it, and it's still getting updates. The latest version is called GNU Health 4.2.3 and was released on 24 September 2023.

GNU Health mainly concentrates on these key areas:

- **Individual management** Keeping track of someone's basic information, family history, home life and other details. The GNU Health project focuses on social medicine, paying close attention to things like education level and occupation.
- **Patient management** This is about gathering information about a person's health. It includes details about how they live, like what they eat or if they have any habits like smoking. It also includes information about their medical care, such as operations they've had or tests they've had.
- **Health center management** This article is mainly about ERP, which means it's about managing things like people, money, keeping track of what's in stock and other important things. It also pays a lot of attention to managing pharmacies and labs.
- **Information management** This means handling the information created by GNU Health. One thing it does is create reports. Reports collect and display data for studies on diseases and other things.

3.1 Modules

Because GNU Health is based on Tryton, it is divided into many parts, called modules. This allows healthcare organisations to tailor the installation to their specific needs and disable features they don't use. The main part of GNU Health is the core module called `health`. All other modules in GNU Health rely on this core module, so it is automatically installed when you install any other module. Figure 3.1 shows this modular structure and highlights some of the modules in GNU Health.

In total, GNU Health has over 50 different parts. There are parts specific to GNU Health, as well as some parts from Tryton. These include parts for inventory, product, financial and analytical accounting. You can also add Occhiolino, a new system for organising information in health and biomedical laboratories. Occhiolino is a separate project within GNU Health.

- **Socioeconomics** This part helps to keep track of details about different things that affect how people live and work together. Some of these things include where people live, how much education they have, where they work, the buildings and roads around them, and their family history.

[1] https://www.gnuhealth.org/about-us.html

- **Lifestyle** As people's health is strongly influenced by the way they live, both physical and mental health are very important. This part of the survey collects information on things like what people eat, how much they exercise, how well they sleep and what they drink or smoke.
- **Gynecology** The Gynecology section focuses on keeping the female reproductive system healthy.
- **Function and Disability** The module handles information about patients' disabilities.
- **Obstetrics** All information about pregnancy, birth and postnatal care will be kept on file.
- **Genetics** The aim of this module is to make patients healthier by using genetic information to change their treatment. It explains the information about genes that can affect a patient's health.
- **Surgery** This tool helps to record surgical procedures. The operations are grouped according to the WHO ICD-10 Procedure Coding System (ICD-10-PCS).
- **Nursing** This tool can be used to record all the tasks that nurses carry out during ward rounds and ambulatory care.
- **Reporting** The Reporting module makes it easy to generate reports on a variety of measurements. This module includes the Demographics Summary, which generates a report showing the number of patients, age distribution and other details.

3.1.1 Core modules

The main part of the health module is important for all other parts of GNU Health. It contains the basic models and classes needed to start a health centre. Other parts only use these models when they need them. It's important to know how these models work and what information they store. Some important things this module provides are

- **Health Institutions** The first step in starting a new GNU Health System is to create a Health Institution. This represents the hospital or company. It's connected to Tryton's party model and contains many details about it. The main purpose of this model is to store information about the different buildings and facilities of the health institution. The most basic facilities are the beds.
- **Domiciliary Units** Domiciliary units are places where people live. They include houses, flats and other types of dwellings. These units can hold information about the condition of the building, the facilities available and the people who live there.
- **Individuals** GNU Health builds on Tryton's party models by extending them with additional attributes. These attributes help to distinguish between different roles, such as health professionals, patients, or both. They also include details about a person's demographics, such as name, gender, age, education level and marital status.
- **Families** GNU Health can also keep track of family relationships. This is really important for understanding genetics and following laws.
- **Book of Life** The The Book of Life is like a large folder of information or pages of life. These pages don't just contain medical records, but also other things that are important for a person's health. There are four different types of Book of Life pages: Biographical, Medical, Demographic and Social.
- **Health Professionals** As mentioned in the "Individuals" section, we made the party tool longer to show what job someone does. The main part of GNU Health also adds a new plan just for health workers. This plan still requires the party to have simple facts about someone and just point to them.
- **Medicament** This model can store details about a medicine. It has things like what it's used for, how strong it is, how much to take and information for pregnant people. It is also connected to Tryton's product model, which is used in the stock module.
- **Prescriptions** This model holds all information about prescriptions. It links patients and their medicines. It includes details such as the patient's name, who prescribed it, how much to take, how often, and when the prescription starts and ends.

- **Vital Records** GNU Health can keep records of when people are born and when they die. Birth certificates contain details such as the person's name, their parents' names, where and when they were born, and any special notes. Death certificates include information such as the person's name, when and where they died, and the cause of death. You can also choose the type of death, such as natural, homicide, suicide or unknown, and whether there was an autopsy. Both types of certificate must be signed by a health professional, who will also add their name and the date and time of the death. Once a certificate has been signed, it cannot be altered.
- **Immunization** Vaccines are part of the medication plan. A model helps you make a schedule for getting one or more vaccines. You can add details such as what the vaccine is for, how many doses are needed, and the best age to get it. Another model helps with the process of giving a vaccine to a person and is linked to the vaccine stock. You can also see if someone has had all their vaccines with the vaccination status report.
- **Radiology** This module designed to upload and update patient's images and to use special viewers to look at the images.

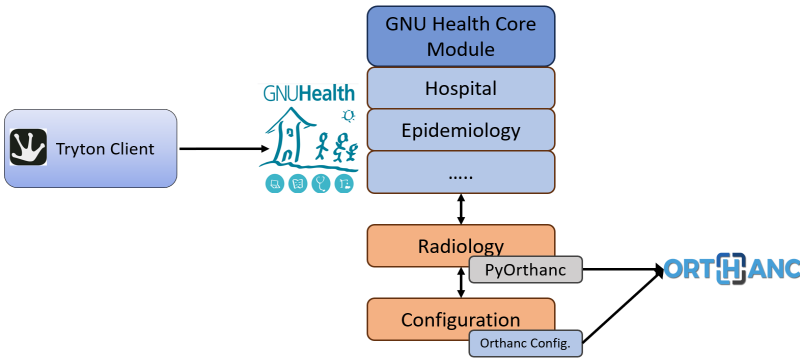
3.1.2 Orthanc Integration Module

Orthanc is a free and open source medical image management system. Its main component is the Orthanc server, which is designed to store DICOM images. Orthanc also provides several DICOM viewers for users to access the stored patient image studies. The integrated Orthanc ecosystem extends the GNU Health System with advanced radiology capabilities directly accessible from the patient's electronic health record. These applications demonstrate the flexibility of the Orthanc server enabled by its rich REST API. We chose `PyOrthanc`[1], a Python client designed for the Orthanc REST API. This client comprehensively covers all methods of the REST API. It also provides many utility functions for interacting with an Orthanc instance.

The new module, `Health_radiology` has been created and smoothly integrated within the GNU Health server. This integration is based on PyOrthanc, a Python client that is used to drive Orthanc from GNU Health through its REST API.

- **Radiology** This is the main module. It is designed for uploading and updating patient images and for viewing the images using special Orthanc DICOM viewers. See the *Orthanc Server Configuration* section for a detailed description of its features. The Orthanc configuration function is used to configure the connection between Orthanc's DICOM servers and GNU Health.

The picture below displays how the integration is designed:



[1] https://pypi.org/project/pyorthanc/

3.2 Plugin

For this project, we've added a new widget plugin called 'DicomBinary' to GNUHealth. This plugin allows to upload multiple DICOM images, as well as files in .gz and .zip formats. It's located in the plugins folder of GNUHealth. The 'DicomBinary' plugin is based on Tryton's binary widget, but with additional functionality for selecting and uploading multiple images in DICOM format.

class DicomBinaryMixin(*Widget*)

This class is a mixin class that extends the functionality of a widget with features related to handling DICOM files and file operations.

Here's a brief description of part of the methods:

- `toolbar`: Returns a toolbar with buttons for saving, selecting, and clearing files.
- `filename_field`: Returns the value of the *filename* field from the *group* dictionary in the *record* object.
- `filters`: Gets a list of file filters to apply when selecting files.
- `_set_uris`: Sets the URIs of the files and updates the content of the field in the wizard.
- `get_data`: Retrieves data from the fiels associated with the record.

class DicomBinary(*DicomBinaryMixin*, *Widget*)

This class inherits from *DicomBinaryMixin* and *Widget* of Tryton. Here's a brief description of each method:

- `__init__(self, view, attrs)`: Initializes the DicomBinary class with given view and attris.
- `__readonly_set(self, value)`: Sets the sensitivity of buttons and text widget based on the given value.
- `sig_key_press(self, widget, event)`: Handles key press events and performs corresponding actions based on the event's key value and widget's editability.
- `sig_icon_press(self, widget, icon_pos)`: This function handles the press event for the signal icon. It takes in the widget, icon position, and event as parameters.
- `display(self)`: Displays the DicomBinary object on the screen.
- `set_value(self)`: Sets the value of the filename field in the record based on the text entered in the `wid_text` field.

3.3 GNU Health Client

The GNU Health HMIS client is a Python application that derives from the Tryton GTK client, with specific features of GNU Health hand healthcare sector. You can find out more about this client and download it from[2].

Tryton SAO serves as a user-friendly interface for accessing and interacting with the Tryton ERP system through a web browser. It provides a streamlined experience for managing various business processes, including inventory, accounting, and customer relationship management. The integration between Tryton SAO and GNU Health is straightforward, and users can seamlessly access GNU Health functionalities through the Tryton SAO web interface, facilitating a smoother workflow and improving the user experience.

[2] https://en.wikibooks.org/wiki/GNU_Health/Installation#Installation_of_the_GNU_Health_Client

CHAPTER

FOUR

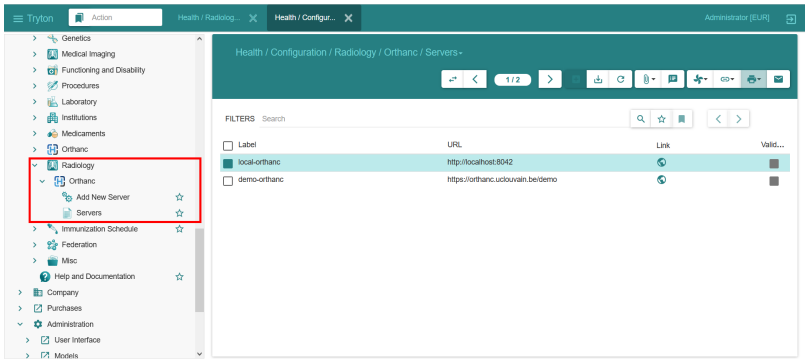
HEALTH RADIOLOGY MODULE

The Health Radiology Module provides functions for management Image data. This module's documentation includes the following sections:

4.1 Orthanc Server Configuration

The primary function is to establish the connection between Orthanc's DICOM servers and GNU Health. The label of a server is a name that the user can choose to represent the server directly. The domain field contains the full web address (URL) to the Orthanc server. This helps GNU Health to find the correct server to connect to. The user must also provide a username and password to log in to the Orthanc server.

The image below displays a snapshot of the Orthanc module:



4.1.1 Configuration

class health_orthanc_configuration.**ServerConfig**(*ModelSQL*, *ModelView*)

This class, *ServerConfig*, is used to connect to an Orthanc DICOM server and to check if a connection to the corresponding domain can be established.

Parameters

- **ModelSQL** (class: `trytond.model.ModelSQL`) – Inherit from the Tryton ModelSQL class for SQL database operations.
- **ModelView** (class: `trytond.model.ModelView`) – Inherit from the Tryton ModelView class for user interface operations.

Here's a brief description of each method:

- **__setup__(cls): Set up the class for database access by initializing properties and contstrains, such as ensuring the uniqueness**
  of the label and domain fields.
- quick_check(domain, user, password): Checks if the server details are correct by attempting to connect to the Orthanc DICOM server with the provided domain.
- on_change_with_validated(self): Updates the validated field based on the current server details by calling the quick_check method with the domain, user, and password attributes.

4.1.2 Wizard

class health_orthanc_configuration.wizard.**add_orthanc_init**(*ModelView*)

This class definition *add_orthanc_init* is a model view for initializing an Orthanc connection.

Parameters
ModelView (class: trytond.model.ModelView) – Inherit from the Tryton ModelView class for user interface operations.

- label: Represents the label of the Orthanc server. Must be unique.
- domain: Represents the full URL of the Orthanc server.
- user: Represents the username for the Orthanc REST server.
- password: Represents the password for the Orthanc REST server.

class health_orthanc_configuration.wizard.**ConnectNewOrthancServer**(*Wizard*)

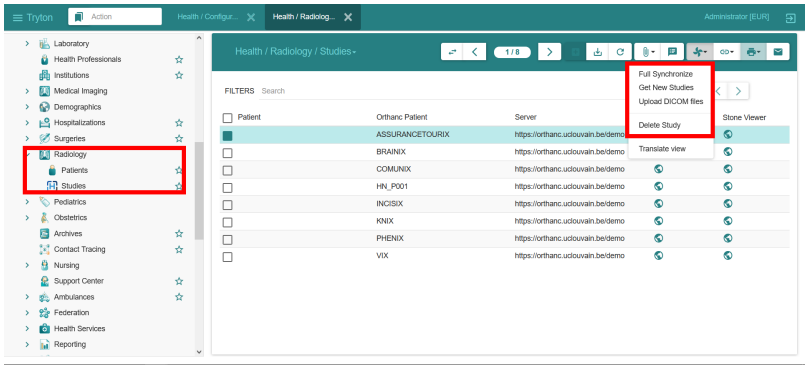
This class, *ConnectNewOrthancServer* defines a wizard for connecting to an Orthanc server.

Parameters
Wizard – A finite state machine.

- start: Displays the initial state of the wizard, asking for the label, URL, username, and password of the Orthanc server.
- connect: Handles the transition when the 'Begin' button is pressed. It attempts to connect to the Orthanc server using the provided credentials.
- status: Displays the status of the connection attempt. It includes a 'Close' button.
- transition_connect(): Connects to the Orthanc servers, handles different exceptions, logs the success or failure of the connection attempt, and return 'status' after completion.
- default_status(fields): Generates a default status dictionary based on the provided fields.

4.2 Radiology

This section presents the core aspect of the Orthanc integration. It is designed for uploading and updating patient images and for viewing the images using special Orthanc DICOM viewers. The image below displays a snapshot of the radiolgy module:



4.2.1 Data Models for image data

In this module we have developed three data models to organise image study data according to the DICOM format. These models represent studies, series within studies (such as CT, MR or PET scans) and the individual instances within each series. They represent the relationships between these elements. For example, a single patient may undergo multiple studies, each study may consist of multiple series, and each series may contain multiple instances.

Patient Orthanc Study

class health_radiology.**PatientOrthancStudy**(*ModelSQL*, *ModelView*)

This class, *PatientOrthancStudy*, defines a model for radiology study.

Parameters

- **ModelSQL** (class: `trytond.model.ModelSQL`) – Inherit from the Tryton ModelSQL class for SQL database operations.
- **ModelView** (class: `trytond.model.ModelView`) – Inherit from the Tryton ModelView class for user interface operations.

Here's a brief description of each method:

- `__setup__()`: Sets up the class with additional buttons for deleting a study and selecting a viewer.
- `get_gnu_patient()`: Retrieves the GNU patient with the given name.
- `ohif_viewer_link()`: Generates a URL for the OHIF viewer and study.
- `ohif_viewer_link()`: Generates a URL for the OHIF viewer and study.
- `delete_study()`: Deletes a study record from the Orthanc server.
- `update_studies()`: Updates the studies in the GNU Health database by fetching studies from the Orthanc servers and creating new studies if needed.

Study Series

class health_radiology.**radiology_study_series**(*ModelSQL*, *ModelView*)

This class definition is for the `Study Series` in the `gnuhealth.radiology` module.

Parameters

- **ModelSQL** (class: `trytond.model.ModelSQL`) – Inherit from the Tryton ModelSQL class for SQL database operations.
- **ModelView** (class: `trytond.model.ModelView`) – Inherit from the Tryton ModelView class for user interface operations.

Here's a brief overview of what each class method does:

- `__setup__()`: Initializes the class and sets up the buttons.
- `get_study_server()`: Retrieves the server for the given study.
- `get_study_patient()`: Returns the name of the patient associated with the given study.
- `get_study_patient()`: Returns the name of the patient associated with the given study.
- `delete_series()`: Deletes study series records and handles exceptions.

Series Instances

class health_radiology.**SeriesInstances**(*ModelSQL*, *ModelView*)

The provided Python code defines a class *SeriesInstances* which is a SQL and view model for a database table. This class represents instances of radiology studies, typically in the context of radiology.

Parameters

- **ModelSQL** (class: `trytond.model.ModelSQL`) – Inherit from the Tryton ModelSQL class for SQL database operations.
- **ModelView** (class: `trytond.model.ModelView`) – Inherit from the Tryton ModelView class for user interface operations.

Here's a succinct explanation of its methods:

- `__setup__()`: Initializes the class and sets up the buttons.
- `get_study_server()`: Retrieves the server for the given study.
- `get_image()`: A method to retrieve an image from an Orthanc server

4.2.2 Wizard

The Tryton wizard is a specific kind of finite state machine utilized within this module. It facilitates tasks such as updating studies, uploading new image data to the server, and deleting existing image data.

Update Image Studies

class health_radiology.wizard.**update_studies**(*Wizard*)

This class definition is a custom Tryton wizard for update patient image studies stored in Orthanc server.

Parameters
Wizard (class: trytond.wizard.Wizard) – A finite state machine.

Here's what each class method does:

- transition_update(): Updates radiology studies and returns 'end'.
- end(): Signals the end of the process and returns the string 'reload'.

Upload Image Data

class health_radiology.wizard.**UploadImageData**(*Wizard*)

This class definition is a custom Tryton wizard for uploading image data.

Parameters
Wizard (class: trytond.wizard.Wizard) – A finite state machine.

Here's what each class method does:

- upload_image_data(): Uploads the image data to the server and handles exceptions.
- transition_upload(): Transitions the upload process by uploading the image data and then updating the radiology studies using update_studies()
- end() Returns the string 'reload'.

Full Synchronize Studies

class health_radiology.wizard.**FullSynchronize**(*Wizard*)

This class definition is a custom Tryton wizard for full synchronizing studies. It updates the studies in the GNU Health database by fetching studies from the Orthanc servers and creating new studies if needed.

Parameters
Wizard (class: trytond.wizard.Wizard) – A finite state machine.

Here's what the class method does:

- FullSynchronize(): Full synchronizes radiology studies.

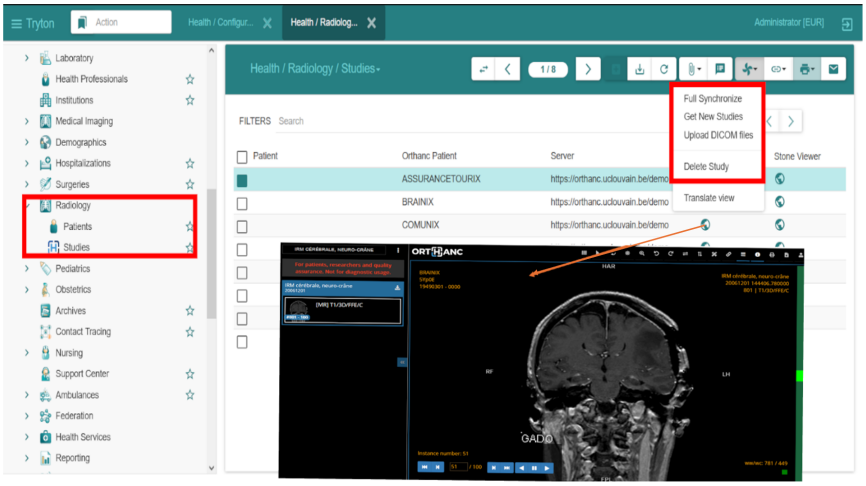
Get new studies

This class defines a Tryton wizard that is responsible for updating studies from an Orthanc server by processing changes to studies, series and instances. It retrieves configuration information, fetches changes from the Orthanc server and updates the studies in the GNU Health database. If there is an error in the process, it throws an exception.

Here's what the class method does:

- GetNewStudies(): Retrieves new studies from the Orthanc server and updates the studies in the GNU Health database.

The following shows the outcome of the integration between GNU Health and Orthanc:



CHAPTER

FIVE

INTEGRATION OF THE DICOM SERVER ORTHANC INTO THE HOSPITAL INFORMATION SYSTEM GNU HEALTH

5.1 Description

The document explanes how to integrate the DICOM server Orthanc into the hospital information system GNU Health. Here is a video showing the new module look like for the end user in the tryton SAO web client:[4]

5.2 Installation

1. You need a working GNU Health 4.4.0 server and an Orthanc server. Follow the instructions at[1] to install and setup the GNU Health server on your test system. In the following, we assume that you have followed the instructions on the website and that your GNU Health server is located in the directory `/home/gnuhealth/gnuhealth`.
2. We also assume that you have created the demo database `ghdemo44` as described at hddttps://en.wikibooks.org/wiki/GNU_Health/The_Demo_database.
3. Install 'PyOrthanc':

```
pip install pyorthanc
```

4. Clone this repository and `cd` into its directory.
5. Copy the new modules `health_radiology` into the GNU Health server:

```
cp -r health_radiology /home/gnuhealth/gnuhealth/tryton/server/modules/

ln -s /home/gnuhealth/gnuhealth/tryton/server/modules/health_radiology /gnuhealth/
↪tryton/server/trytond-6.0.43/trytond/modules
```

Note: If your GNU Health server uses a different version of tryton, you have to adapt the directory names accordingly.

1. Activate the modules for the demo database:

```
/home/gnuhealth/gnuhealth/tryton/server/trytond-6.0.43/bin/trytond-admin -d␣
↪ghdemo44 -u health_radiology --activate-dependencies
```

[4] https://www.youtube.com/watch?v=wL8MbM8iu8A
[1] https://en.wikibooks.org/wiki/GNU_Health/Installation

Note: If you are using a different database, you have to replace `ghdemo44` by its name.

1. Install the local Orthanc server and plugins:

```
sudo apt install orthanc

sudo apt install orthanc-dicomweb

sudo apt install orthanc-webviewer

sudo apt install orthanc-wsi
```

Download the OHIF viewer `libOrthancOHIF.so` in https://orthanc.uclouvain.be/downloads/linux-standard-base/orthanc-ohif/1.2/index.html and copy it in the plugins directory `/usr/share/orthanc/plugins/`.

Download the stone web viewer `libStoneWebViewer.so` in https://orthanc.uclouvain.be/downloads/linux-standard-base/stone-web-viewer/2.5/index.html and copy it in the plugins directory `/usr/share/orthanc/plugins/`.

8. We have created a new widget that allows to select and upload multiple DICOM files in the desktop client and web client. The widget has to be added to the clients.

   - For the desktop client: Let's assume that your desktop client is installed in `/home/gnuhealth/health-hmis-client/`. Copy the widget into the plugin directory of the client:

```
cp -r /home/health_radiology/dicombinary /home/gnuhealth/health-hmis-
↪client/gnuhealth/plugins/
```

   - For the SAO web client: Let's assume you have installed the SAO client in `/home/gnuhealth/sao` (see[3]). If there is not already a custom.js file in the web client's directory, you can just copy the widget into the client:

```
cp custom.js /home/gnuhealth/sao
```

   Otherwise, you have to manually add the content of our custom.js file to the existing file in the SAO client.

9. If you are using nginx as proxy:

   In file `/etc/nginx/sites-enabled/GH_HIS_HTTP.conf` add the following line in the server block.

```
client_max_body_size 500M;

proxy_send_timeout 300;
```

Restart nginx (debian, ubuntu,...):

```
  systemctl restart nginx


.. note:: If you are using ansible to install gnuhealth as described in https:/
↪/docs.gnuhealth.org/ansible/examples/gnuhealth_server_and_client.html. You␣
↪have to modify the ansible files to make the above changes permanent.
```

[3] https://foss.heptapod.net/tryton/tryton/-/tree/branch/default/sao

10. Restart your GNU Health server.
11. Connect to the demo database with the client and open the module `Configuration/Radiology/Orthanc`. Select `Add Orthanc Server` to configure the connection to the Orthanc server.
12. Watch the video[Page 15, 4] to see how to use the `Health Radiology` module to fetch DICOM studies from the Orthanc server and open medical images.

CHAPTER

SIX

INTEGRATION OF THE DICOM SERVER ORTHANC INTO THE HIS GNU HEALTH IN ONE SYSTEM

6.1 Description

The document explains how to integrate the DICOM server Orthanc into the GNU Health hospital information system, combining both the GNU Health server and client into a single system[1].

6.2 Installation

1. To begin, follow the steps to install both the GNU Health server and clientPage 19, 1.

   Run GNU Health:

   ```
   gnuheath-client
   ```

2. Create the virtual environment:

   ```
   sudo su gnuhealth -s /bin/bash

   cd

   activate
   ```

3. Install 'PyOrthanc':

   ```
   pip install pyorthanc
   ```

4. Clone the last version of HIS GNU Health:

   ```
   git clone -b wip-orthanc-integration https://codeberg.org/gnuhealth/his.git
   ```

Note: It is the test branch `wip-orthanc-integration`.

5. Integrate the new modules:

   ```
   ln -s /opt/gnuhealth/his/tryton/health_radiology /opt/gnuhealth/venv/lib/python3.11/
   ↪site-packages/trytond/modules/
   ```

[1] https://docs.gnuhealth.org/ansible/examples/gnuhealth_server_and_client.html

6. Update modules dependencies:

```
trytond-admin -d health -c etc/trytond.conf -u health_radiology --activate-
↪dependencies
```

Note: If you are using a different database, you have to replace `ghdemo44` by its name.

7. Install the local Orthanc server and plugins:

```
sudo apt install orthanc

sudo apt install orthanc-dicomweb

sudo apt install orthanc-webviewer

sudo apt install orthanc-wsi
```

Download the OHIF viewer `libOrthancOHIF.so` in https://orthanc.uclouvain.be/downloads/linux-standard-base/orthanc-ohif/1.2/index.html and copy it in the plugins directory `/usr/share/orthanc/plugins/`.

Download the stone web viewer `libStoneWebViewer.so` in https://orthanc.uclouvain.be/downloads/linux-standard-base/stone-web-viewer/2.5/index.html and copy it in the plugins directory `/usr/share/orthanc/plugins/`.

8. We have created a new widget that allows to select and upload multiple DICOM files in the desktop client and web client. The widget has to be added to the clients.

   - For the desktop client: Let's assume that your desktop client is installed in `/home/gnuhealth/health-hmis-client/`. Copy the widget into the plugin directory of the client:

```
cp -r /home/health_radiology/dicombinary /home/gnuhealth/health-hmis-client/
↪gnuhealth/plugins/
```

   - For the SAO web client: Let's assume you have installed the SAO client in `/home/gnuhealth/sao` (see[2]). If there is not already a custom.js file in the web client's directory, you can just copy the widget into the client:

```
cp custom.js /home/gnuhealth/sao
```

9. **If you are using nginx as proxy:**
   In file `/etc/nginx/sites-enabled/GH_HIS_HTTP.conf` add the following line in the server block:

```
client_max_body_size 500M;

proxy_send_timeout 300;
```

Restart nginx (debian, ubuntu,...):

```
systemctl restart nginx
```

[2] https://foss.heptapod.net/tryton/tryton/-/tree/branch/default/sao

Note: If you are using ansible to install gnuhealth as described in https://docs.gnuhealth.org/ansible/examples/gnuhealth_server_and_client.html. You have to modify the ansible files to make the above changes permanent.

10. Restart your GNU Health server.
11. Connect to the demo database with the client and open the module `Configuration/Radiology/Orthanc`. Select `Add Orthanc Server` to configure the connection to the Orthanc server.
12. Watch the video[3] to see how to use the `Radiology` module to fetch DICOM studies from the Orthanc server and open medical images.

[3] https://www.youtube.com/watch?v=wL8MbM8iu8A

INDEX

D

H