

FHIR Interface to Controlled Substances Database Manual

Pants on FHIR

Ryan Cobb, Fredrick Galoso, David Mooney

Table of Contents

[Table of Contents](#)

[Introduction](#)

[Mirth](#)

[Install FHIR Extension into Mirth Connect](#)

[Debug Logging](#)

[FHIR Listener Channel](#)

[Configuration Map Properties](#)

[Import FHIR Listener Channel](#)

[Database](#)

[Installation](#)

[Data Population](#)

[Flask Application \(GUI\)](#)

[Installation](#)

[Configuration](#)

[Search](#)

[Example search request using cURL](#)

[Example search request using HTML form](#)

Introduction

This manual is supplied with the FHIR Interface to Controlled Substance Database developed for use within the Utah Department of Health's Case Management System. The following sections detail the components used to develop the interface and provide instructions for component setup, configuration, and use. While it is known that the database used for development differs from the database used by the production system, instructions for its installation and population are provided for completeness.

Mirth

Install FHIR Extension into Mirth Connect

Download the extension from Mirth's website.

<https://info.mirth.com/Fast-Healthcare-Interoperability-Resources-FHIR.html>

More information about this extension is on Mirth's public wiki at

<http://www.mirthcorp.com/community/wiki/display/mirth/FHIR+Technology+Preview+2>.

The following text is extracted from the public wiki User Guide at

<http://www.mirthcorp.com/community/wiki/display/mirth/User+Guide>.

The FHIR Listener is installed like any other Mirth Connect extension. This can be done through the Mirth Connect Administrator in the Extensions view, or by extracting the contents of the provided ZIP file into the extensions directory inside the Mirth Connect installation directory.

After restarting the server, make sure that you launch the Mirth Administrator by clicking the Launch Mirth Connect Administrator button and not using a previously downloaded JNLP file.

After installing, you should now have "FHIR Listener" as one of your source connector options.

Debug Logging

The FHIR Listener logs out information for every request that comes in. In order to view these logs you can add the following to your log4j.properties file:

```
# FHIR Listener debugging
```

```
log4j.logger.com.mirth.connect.connectors.fhir.server = DEBUG
```

FHIR Listener Channel

Configuration Map Properties

In MC Admin, go to Settings > Configuration Map and click Import Map. Navigate to the "fhir configuration map.properties" file under FHIR-Interface-to-Controlled-Substance-Database/Final Project/Final Application/Application/mirth and import those settings. Update the database username, password, driver jdbc URL and max retries to point at the controlled substances database.

Import FHIR Listener Channel

Go to Channels and click Import Channel. Navigate to FHIR Listener.xml under FHIR-Interface-to-Controlled-Substance-Database/Final Project/Final Application/Application/mirth and import the channel. It includes some code templates.

The default port is 9001. The default URL is dstu2 so the full address will be `http://your_server:9001/dstu2/`.

Both XML and JSON are supported.

Supported resources are Patient and MedicationOrder. Patients may be searched by family name, given name, birth date and patient id. MedicationOrders may be searched by patient id.

A JavaScript transformer is used to route requests to the appropriate destination. There are destinations for each of the supported resource type.

Each destination is a JavaScript writer which queries the database and uses the [HAPI FHIR API](#) to build response objects. Pay particular attention to the search-patient and search-medicationorder destinations. These are the two destinations used by the GUI.

Every attempt was made to keep the SQL interface as vanilla as possible to avoid integration issues with MSSQL. PostgreSQL was used in development. One area that may cause issues is date handling. We deal with two dates: patient birth dates and Rx order dates. The dates are parsed from strings using a very simple format. This code may need to be modified in production. The locations are:

1. Destination search-patient line 73
2. Destination search-patient line 175.
3. Destination search-medicationorder line 165.

See [SimpleDateFormat](#) for details on date parsing. [Mirth Connect Working With Dates](#) may also be helpful.

Deploy the channel and you should be up and running.

Test the interface by accessing the base URL in a browser, e.g. http://your_server:9001/dstu2/. Test patient search by accessing http://your_server:9001/dstu2/patient. Test medication order search by accessing http://your_server:9001/dstu2/medicationorder. Refer to the MC Admin Dashboard for errors. Mirth.log will also have some details if you enabled debug logging.

Database

The development environment for this project used [postgresql](https://www.postgresql.org/) to replicate the databases required for this project. PostgreSQL was used as an open-source alternative to the MSSQL system used in the production environment. PostgreSQL was also chosen due to its semantic similarity to MSSQL. Because the database management system used in the development environment differed from the one used in production, the following sections will detail the steps necessary to install and populate the databases using postgresQL.

Installation

If the installation of postgresQL is required, use the following link and find the appropriate installation guide for system-specific installation instructions.

https://wiki.postgresql.org/wiki/Detailed_installation_guides

Data Population

Within the virtual environment, the following commands can be entered to access postgresQL:

```
~$ sudo su - postgres  
-bash-4.2$ psql
```

This places the user in the postgres database. Indicated by the following:

```
postgres=#
```

If this is the first time accessing postgres, a new database should be created. This can be done with the following command:

```
postgres=# create database <database_name>;
```

For the purposes of this manual, the <database_name> will be `controlled_substances`. Once this database has been created, it can be accessed via

```
postgres=# \c controlled_substances;
```

If this database was connected to successfully, the following message will appear:

```
You are now connected to database "controlled_substances" as user "postgres".
```

And the prompt will change to

```
controlled_substances=#
```

To maintain consistency with the production environment, the schema provided in FHIR-Interface-to-Controlled-Substance-Database/Final Project/Final Application/Research/CSD schema tables.txt was replicated.

```
controlled_substances=# create schema dbo;
```

To create a table within this schema the following command should be entered:

```
controlled_substances=# create table dbo.<table_name>(<column_name> <data_type><size>)
```

The following tables were created: csd_persons2_transactions_2, doc_pd_master, ndc_master, and trans_hist. The column names, datatypes, and other column constraints were replicated from the provided "CSD schema tables.txt" file.

Anonymous synthetic test data was provided in .CSV files for each of the tables listed above. These files were used to populate each table. This was done with the following command:

```
COPY dbo.<table_name> FROM <full_path/data_file> DELIMITER ',' CSV HEADER;
```

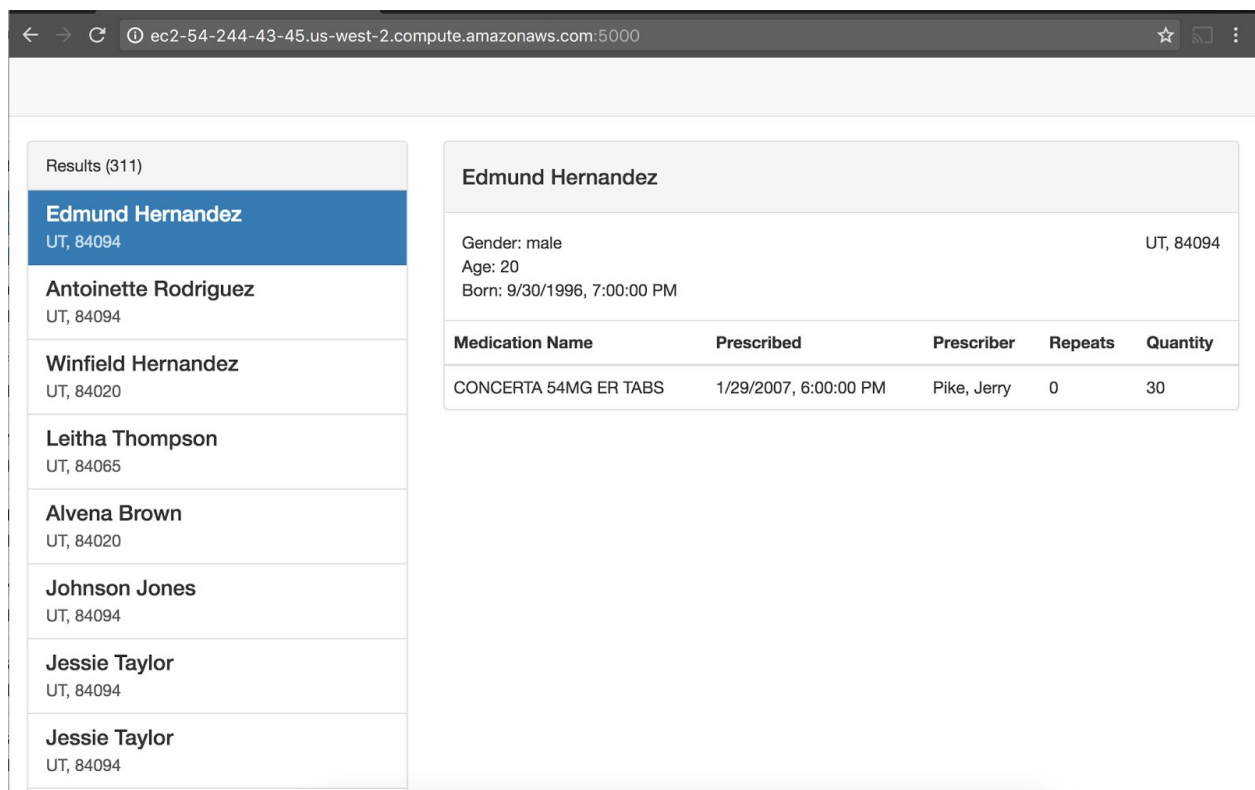
A snapshot of the fully populated was captured and is stored in FHIR-Interface-to-Controlled-Substance-Database/Final Project/Final Application/Application/database/csd_dump.txt. This file can be used to recreate/restore the test environment database using the following command:

```
psql controlled_substances < csd_dump.txt
```

Flask Application (GUI)

Results can be queried and displayed in the included web application. This application implements a REST endpoint for searching for a Patient and displays those results in a web page.

This web application consumes the FHIR resources defined in the [FHIR Listener Channel](#). Requests are done client-side using [AJAX](#), in a self contained web page created with [Flask](#), a server-side Python web framework. JSON responses from the FHIR server are rendered dynamically using [React](#), a JavaScript library for building user interfaces.



The screenshot shows a web browser window with the address bar displaying "ec2-54-244-43-45.us-west-2.compute.amazonaws.com:5000". The application interface is divided into two main sections. On the left, a sidebar titled "Results (311)" lists several patient entries. The first entry, "Edmund Hernandez", is highlighted with a blue background and includes the text "UT, 84094". Below it are entries for "Antoinette Rodriguez", "Winfield Hernandez", "Leitha Thompson", "Alvena Brown", "Johnson Jones", "Jessie Taylor", and another "Jessie Taylor". On the right, a detailed view for "Edmund Hernandez" is shown. It includes patient demographics: "Gender: male", "Age: 20", and "Born: 9/30/1996, 7:00:00 PM", with "UT, 84094" on the right. Below this is a table of medications.

Medication Name	Prescribed	Prescriber	Repeats	Quantity
CONCERTA 54MG ER TABS	1/29/2007, 6:00:00 PM	Pike, Jerry	0	30

Installation

The frontend application is implemented using Python, HTML5, CSS, and JavaScript. The application is self contained in FHIR-Interface-to-Controlled-Substance-Database/Final Project/Final Application/Application/app. The GUI has the following dependencies

- [Python 2.7.x](#) (required)
- [pip](#) (required)
- [Virtualenv](#) (optional dependency to isolate Python environment)

After installing these base dependencies, installing and running the web GUI can be done by running the following commands (Linux or Unix-like operating systems)

```
cd FHIR-Interface-to-Controlled-Substance-Database/Final\ Project/Final\
Application/Application
```

```
virtualenv env
```

```
source env/bin/activate # Run env\Scripts\activate in Windows
```

```
pip install -r app/requirements.txt
```

To run the web application, execute the following command. You should then be able to open a web browser to localhost:5000 or http://your_server:5000 to view it.

```
python -m app
```

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Configuration

The web application directly consumes the Mirth Connect FHIR server. In order to change the URL that is serving these requests, change the FHIR_URL variable in FHIR-Interface-to-Controlled-Substance-Database/Final Project/Final Application/Application/app/static/js/Results.js.

Search

The web application exposes a [REST](#) endpoint for searching. Patient searches can be performed using an [HTTP POST](#) to http://your_server:5000/ using zero or more form-data parameters. These form-data parameters are the following:

Parameter	Description	Format
identifier	Patient ID	String
family	Family Name	String
given	Given Name	String
birthdate	Birth Date	yyyy-mm-dd

This enables custom form based applications to be built that integrate directly with patient search. The web application includes an example integration in /search, but as long as the client

submits a POST request to / with the supported parameters, any application can use the search results web application.

Example search request using cURL

```
curl 'http://localhost:5000/' -d 'identifier=&family=Hernandez&given=&birthdate='
```

Example search request using HTML form

```
<form class="form-horizontal" action="/" method="post">
  <input type="text" name="identifier" value="" />
  <input type="text" name="family" value="Hernandez" />
  <input type="text" name="given" value="" />
  <input type="text" name="birthdate" value="" />

  <button type="submit">Search</button>
</form>
```

← → ↺

ec2-54-244-43-45.us-west-2.compute.amazonaws.com:5000/search

☆

Identifier

Family Name

Given Name

Birth Date

yyyy-mm-dd

Search

Search Criteria: Birth Date: 1996-08-27

Results (1)

Antoinette Rodriguez

UT, 84094

Antoinette Rodriguez

Gender: female

Age: 20

Born: 8/26/1996, 7:00:00 PM

UT, 84094

Medication Name	Prescribed	Prescriber	Repeats	Quantity
HYDROCODONE 5MG/APAP 500MG TABS	2/2/2007, 6:00:00 PM	Bender, Rain	1	25