MOTECH Extensions - Developer Reference

# Architecture Overview

A pictorial diagram on overview of integration solution –

**Highlights on above architecture**

While travelling in-vehicle Health workers could have two mobile device (J2ME & Android) OR just one Android device, since MOTECH mobile app is supported both on J2ME and Android; while OpenXC App runs on Android device

* Both the devices will connect and interact with their respective backend (OpenXC Backend and MOTECH Backend) – mode of connection will be specific cellular network
* For India CommCare is the App that runs on Health worker mobile device (J2ME or Android) and provides relevant interface for patient registration and other details update. This App interacts with MOTECH Backend.
* Vehicle Data stream as received over OpenXC interface via OBD2 port VI, will be uploaded to OpenXC backed over provided interface.
* Developer could use both the web services interface (from OpenXC as well as from MOTECH) for leveraging vehicle data and Patient’s data – AND write their innovative Apps.

**For interfacing and fetching required data on patient and health worker from MOTECH a new module named *‘mHealthDataInterface’* is created which exposes interface in the form of web services.**

**The developer needs to use the whole set up of MOTECH platform in his/her development environment for using the module ‘*mHealthDataInterface’.***

For doing setup of MOTECH platform, please refer the document – ***‘mHealth – MOTECH-Setup.docx’***

# Custom Module (‘mHealthDataInterface’) on MOTECH

To expose custom APIs from MOTECH we have created a custom module named **mHealthDataInterface**. This module needs be added to the MOTECH, above the core platform along with other custom modules, to make APIs available.

**Purpose of mHealthDataInterface Module:**



Figure 1. MOTECH Core Platform Architecture (source: <http://docs.motechproject.org/en/latest/architecture/core_architecture.html>)

“To fetch detail of patients and health worker from MOTECH and expose required web service APIs providing these data to consumer (developer) for further usage.”

This data exposed through Spring-based web service APIs.

Additional application that is used with MOTECH core platform is OpenMRS for using patient and health worker data.

**IMP!! Developer must deploy a local running instance of OpenMRS for his/her development purpose of ‘mHealthDataInterface’ module. Since mHealthDataInterface’ interfaces with local instance of OpenMRS only.**

Module **mHealthDataInterface**

* Fetches data from OpenMRS running instance explained above, through the REST APIs exposed by OpenMRS.
* Fetches data from OpenMRS, in a direct interface with OpenMRS DB (this if especially for the data that is not available from web services.

# Adding jar of the new module into MOTECH Core Platform

To create a new module, following links can be referred:

* <https://docs.google.com/document/d/1uslp8H1uHDuw-rpZ9vLb0bX8zwdgqUAaMjvOcVHKXF4/edit>
* [http://docs.MOTECHproject.org/en/latest/get\_started/archetype.html](http://docs.motechproject.org/en/latest/get_started/archetype.html)

To add the new module into MOTECH, please refer the document **‘mHealth-MOTECH-Setup.docx’.**

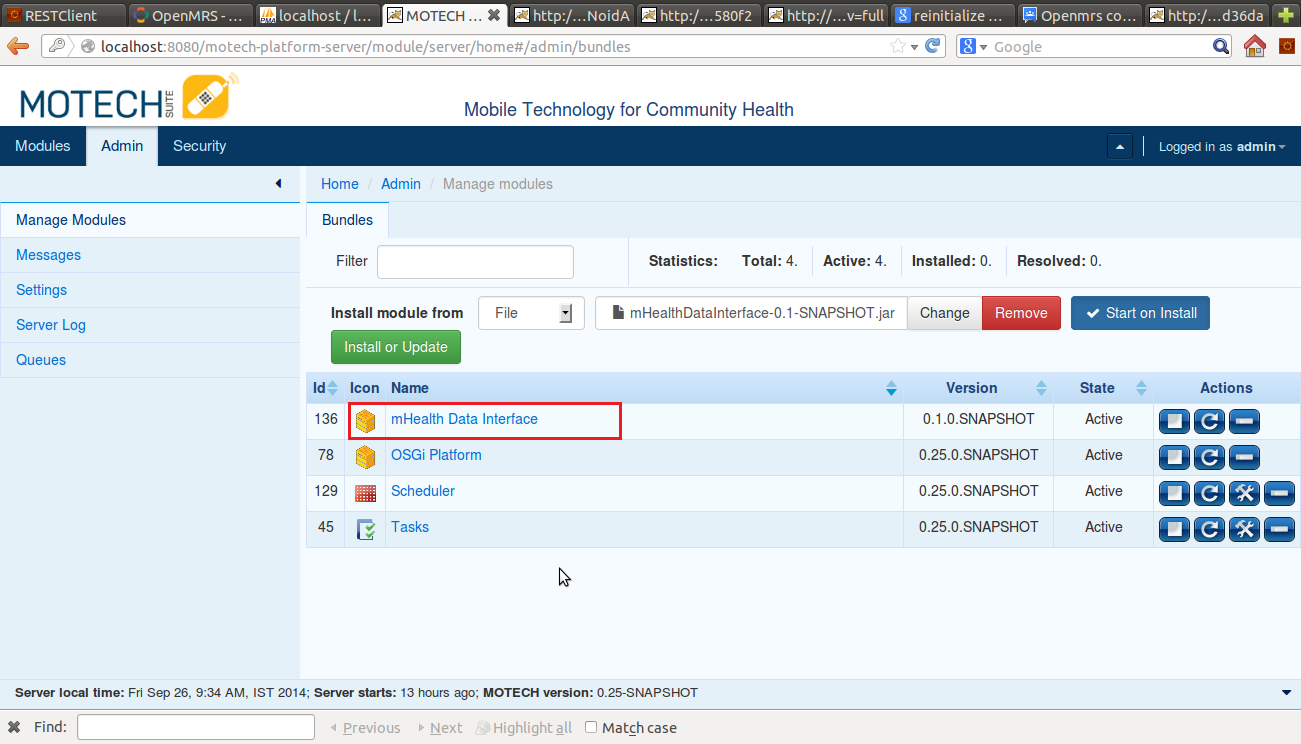
Below source artifacts are provided for developer usage and reference –

1. Jar file ‘**mHealthDataInterface-0.1-SNAPSHOT.jar** ’
2. ‘**mHealthDataInterface’** module source code

Once the MOTECH is setup in local environment, either the module can be imported to the development platform and compiled/executed to create a new jar same as provided. Or, the jar provided can be directly used in running local setup of MOTECH for immediate use.

Following steps can be followed to install the jar, so that the module gets added to the MOTECH backend –

1. Login to the MOTECH Portal and go to “*Admin >> Manage Modules*”.
2. In the field “*Install Module From*”, select from the dropdown “*File*” (highlighted in the screenshot below)
3. Click on “*Select file*” button. Browse and select the jar file created
4. Click on the “*Install or Update*” button.
5. The module will be then added into the MOTECH core and visible in the module list, as shown in the below screenshot –



# APIs Exposed From mHealthDataInterface Module on MOTECH

**Set of APIs are listed below along with ref url :**

1. ***verifyHealthWorker( )***

***ref url: /mHealthDataInterface/verifyHealthWorker/{healthWorkerId}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **verifyHealthWorker** | | | | |
| **Protocol** | HTTP - REST/SOAP API | | | |
| **Description** | Verify health worker’s credentials from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***healthWorkerID*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : valid health worker  : invalid health worker |
| ***errorMsg*** | : object not found |

1. ***getHealthWorkerDetail( )***

***ref url: /mHealthDataInterface*** ***/healthWorkerDetail/{healthWorkerId}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **getHealthWorkerDetail** | | | | |
| **Protocol** | HTTP - REST/SOAP API | | | |
| **Description** | Fetch health worker’s details from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***healthWorkerID*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : Provider (health worker) Object  Key values of Provider (health worker) Object: id, name, retired |
| ***errorMsg*** | : object not found |

1. ***getPatientDetail( )***

***ref url: /mHealthDataInterface*** ***/patientDetail/{patientId}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **getPatientDetail** | | | | |
| **Protocol** | HTTP - REST/SOAP API | | | |
| **Description** | Fetch patient’s details from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***patientID*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : Patient Object  Key values of Patient Object: id, name, gender, birthdate, address. |
| ***errorMsg*** | : object not found |

1. ***getPatientVillage()***

***ref url: /mHealthDataInterface/patientVillage/{patientId}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| getPatientVillage | | | | |
| **Protocol** | HTTP - REST/SOAP API | | | |
| **Description** | Fetch patient’s location (village) from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***patientID*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : Address Object  Key values of Patient object: uuiid, address, stateProvince,  country,postal code. |
| ***errorMsg*** | : object not found |

1. ***getVisitListByPatientId ()***

***ref url: /mHealthDataInterface*** ***/visitListByPatientId/{patientId}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| getVisitListByPatientId | | | | |
| **Protocol** | HTTP – REST/SOAP API | | | |
| **Description** | Fetch all visits of health workers for a patient from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***patientID*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : List of Encounter  Key values/Object in List: id, locationId, locationName, encounterType, Provider (health worker) Object  Key values of Provider (health worker) Object: id, name, address |
| ***errorMsg*** | : object not found |

1. ***getPatientsByVillage ()***

***ref url: /mHealthDataInterface*** ***/patientsByVillage/{village}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| getPatientsByVillage | | | | |
| **Protocol** | HTTP – REST/SOAP API | | | |
| **Description** | Fetch all patients belongs to a village from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***Village name*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : List of Patient  Key values/Object in List: id, name, address, state, country, postal code, phone no. |
| ***errorMsg*** | : object not found |

1. ***getPatientsByPostalCode ()***

***ref url: /mHealthDataInterface*** ***/patientsByPostalCode/{postalCode}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| getPatientsByVillage | | | | |
| **Protocol** | HTTP – REST/SOAP API | | | |
| **Description** | Fetch all patients living in a postal code area from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***Postal code*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : List of Patient  Key values/Object in List: id, name, address, state, country, postal code, phone no. |
| ***errorMsg*** | : object not found |

1. ***getPatientsDetailByName ()***

***ref url: /mHealthDataInterface*** ***/patientsDetailByName/{patientName}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| getPatientsByVillage | | | | |
| **Protocol** | HTTP – REST/SOAP API | | | |
| **Description** | Fetch all patients details on the basis of name. | | | |
| **Input Parameter** | Data Format : JSON data | | ***name*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : List of Patient having the name given.  Key values/Object in List: id, name, birth date, address, state, country, postal code, phone no. |
| ***errorMsg*** | : object not found |

1. ***getHealthWorkersDetailByName()***

***ref url: /mHealthDataInterface*** ***/healthWorkersDetailByName/{healthWorkerName}***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| getPatientsByVillage | | | | |
| **Protocol** | HTTP – REST/SOAP API | | | |
| **Description** | Fetch all health worker details on the basis of name from MOTECH using http REST API. | | | |
| **Input Parameter** | Data Format : JSON data | | ***name*** | : String |
| **Output** | Json String | **Response** | ***responseMsg*** | : List of Health workers having the name given.  Key values/Object in List: id, name, birth date, address, state, country, postal code, phone no. |
| ***errorMsg*** | : object not found |

# Testing APIs through REST Client

To test these APIs of ‘**mHealthDataInterface’** module in development environment, REST Client plugin can be used. Different browsers use different plugins for the REST Client. The download links of few are shared below:

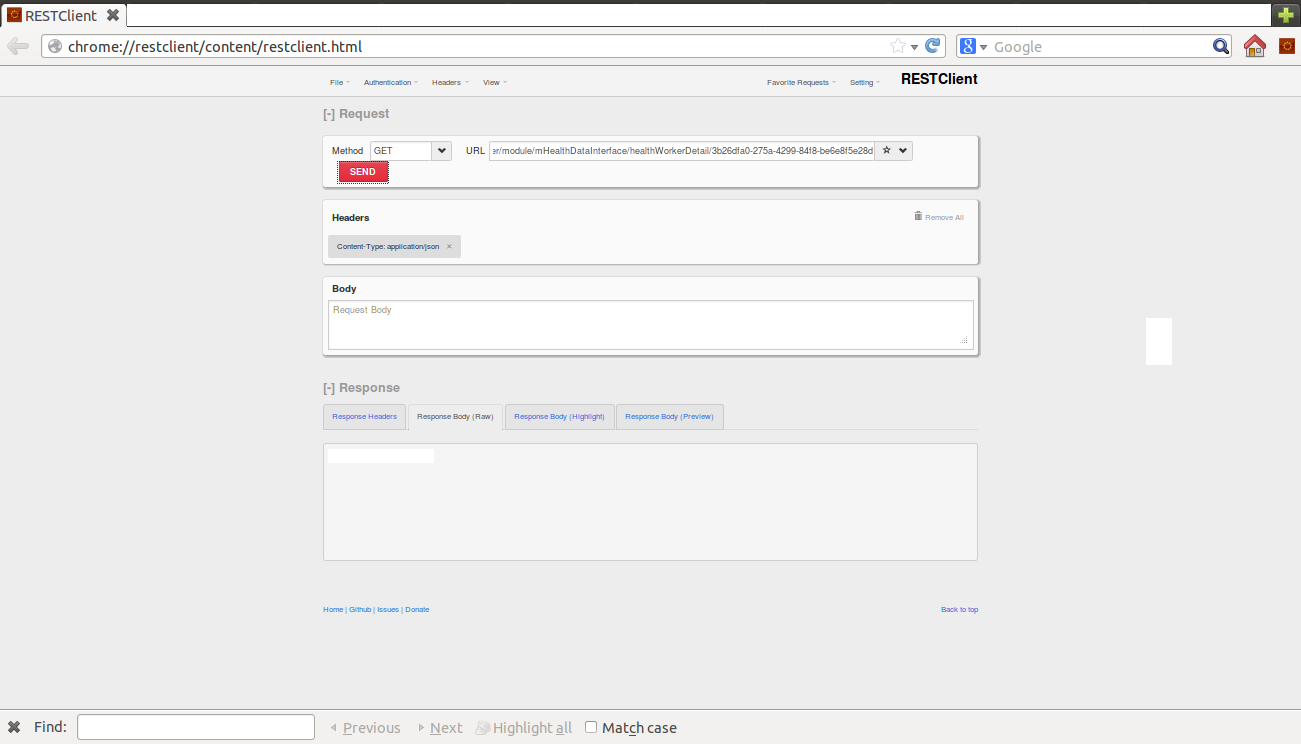
* Chrome :

<https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo>

* Mozilla Firefox:

<https://addons.mozilla.org/en-US/firefox/addon/restclient/>

Chrome REST Client screenshot is given below:



Following points needs to be noted to test APIs through REST Client:

1. In the **URL** textbox, enter the URL of REST API to be tested.
2. In the **METHOD** drop down, the request type (GET/POST/PUT/DELETE etc.) can be selected.
3. Selecting the **HEADERS** option on the top, context-type and response type can be changed. For exposing JSON data “application/json” can be selected.
4. Response can be checked in **RESPONSE** section. **Response headers** tab gives the status of the response and **Response Body** tabdisplays actual response.

# Appendix 1: OpenMRS

**OpenMRS** is a Java-based, web-based system which stores medical record of any person. It is an Electronic medical database, which includes data related to medical billing, clinics, hospitals etc. It also keeps the record of health workers (care providers like doctors, nurse, etc.). OpenMRS has been designed to have a tiered architecture. The real strength of OpenMRS is in its robust and flexible data model.  Its API layer allows a developer to only have to know Java objects and read/save to them.

<http://openmrs.org/>

OpenMRS exposes some REST based services through which data in OpenMRS database can be fetched and used.

To use the OpenMRS data and test, OpenMRS application needs to be installed/deployed along with the database in the same server where MOTECH Backend is setup. The data can be populated into the OpenMRS database, through the OpenMRS application UI or manually through database.

OpenMRS application can be downloaded with database from here – <http://openmrs.org/download/>

**OpenMRS Database**

The OpenMRS database dictates a robust & explicit representation of how care/medical information is stored. The structure of this data model defines the scalability and flexibility of a system.

The OpenMRS database gets created automatically, when the OpenMRS application is deployed.

Database has extensive set of tables, as per the requirement in development, required ones can be used. Developers can leverage other tables also as per the need.

The ER diagram of OpenMRS database can be found here:

<https://wiki.openmrs.org/download/attachments/589829/openmrs_data_model_1.9.0.png?version=3&modificationDate=1339463802000&api=v2>

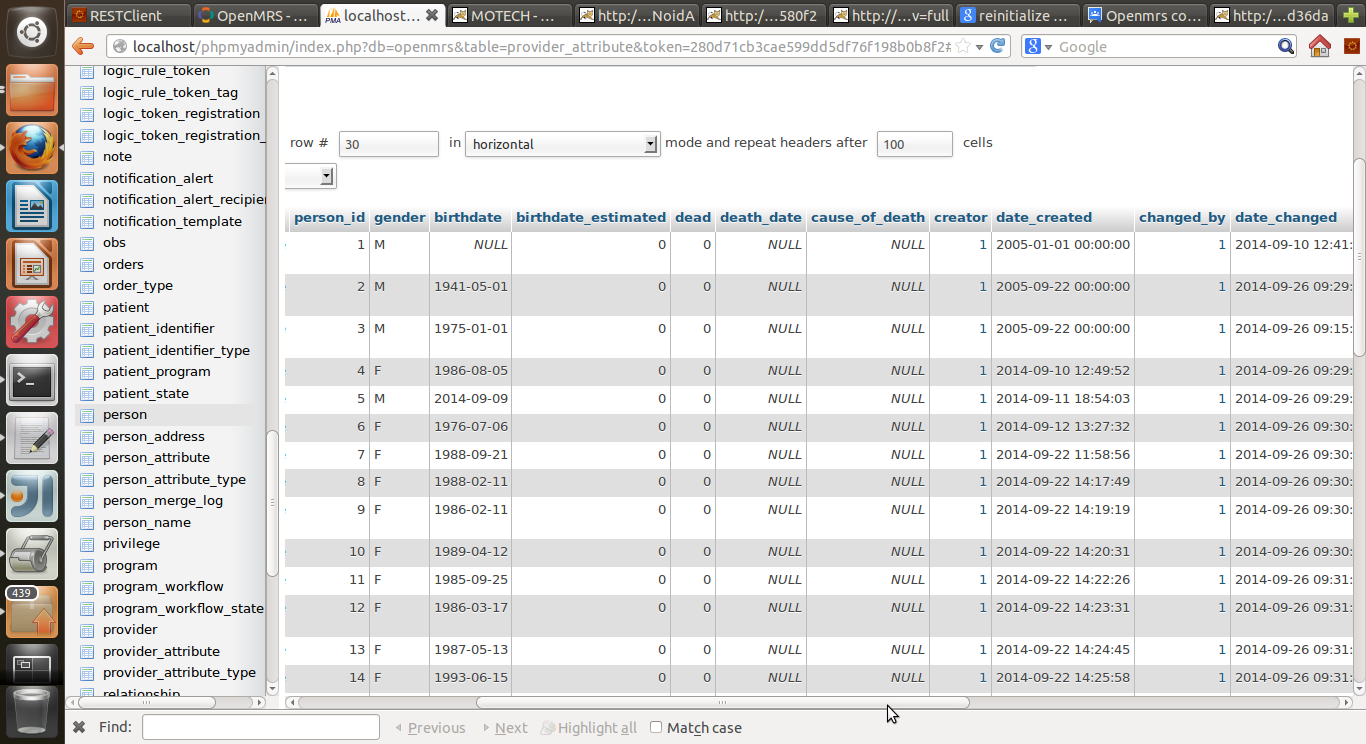
Detail on OpenMRS Data Model is given here:

<http://en.flossmanuals.net/openmrs-guide/openmrs-information-model/>

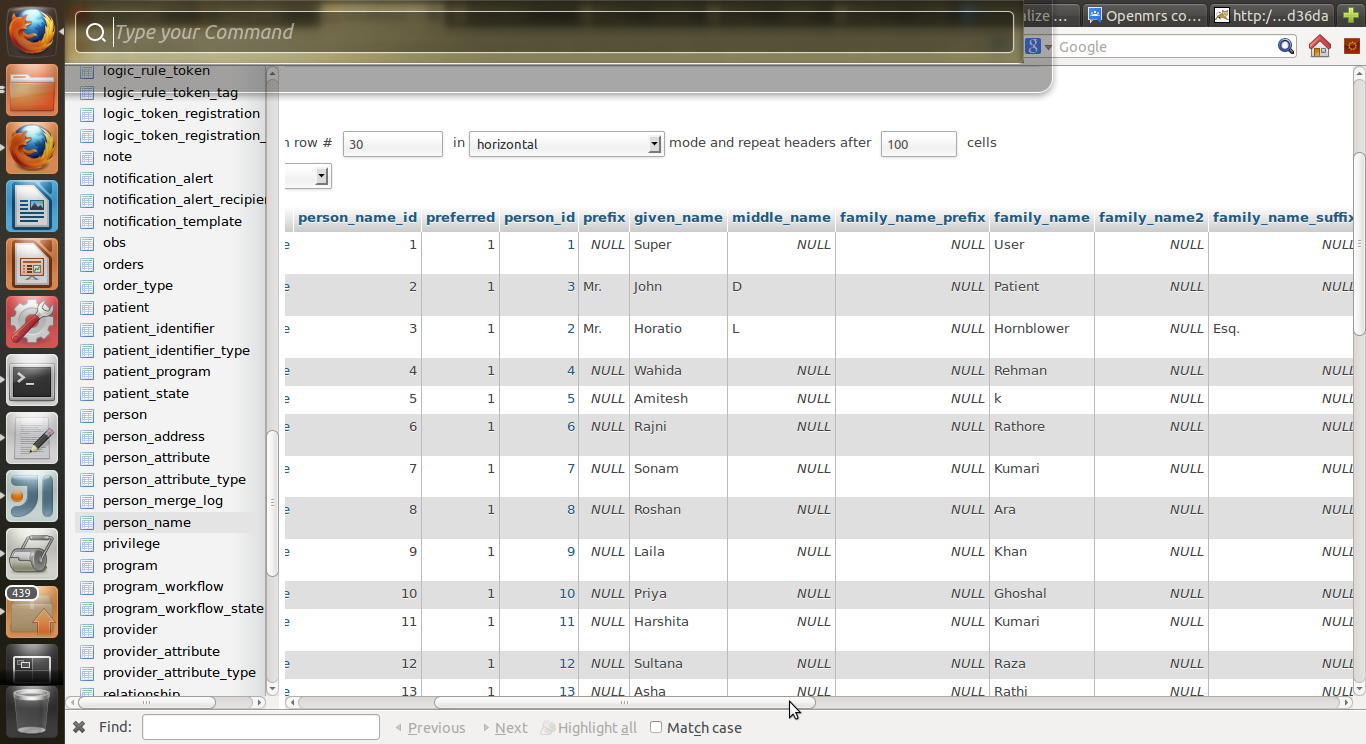
## OpenMRS Tables used in mHealthDataInterface module

Following are the list of OpenMRS tables that are used by ‘***mHealthDataInterface’*** module:

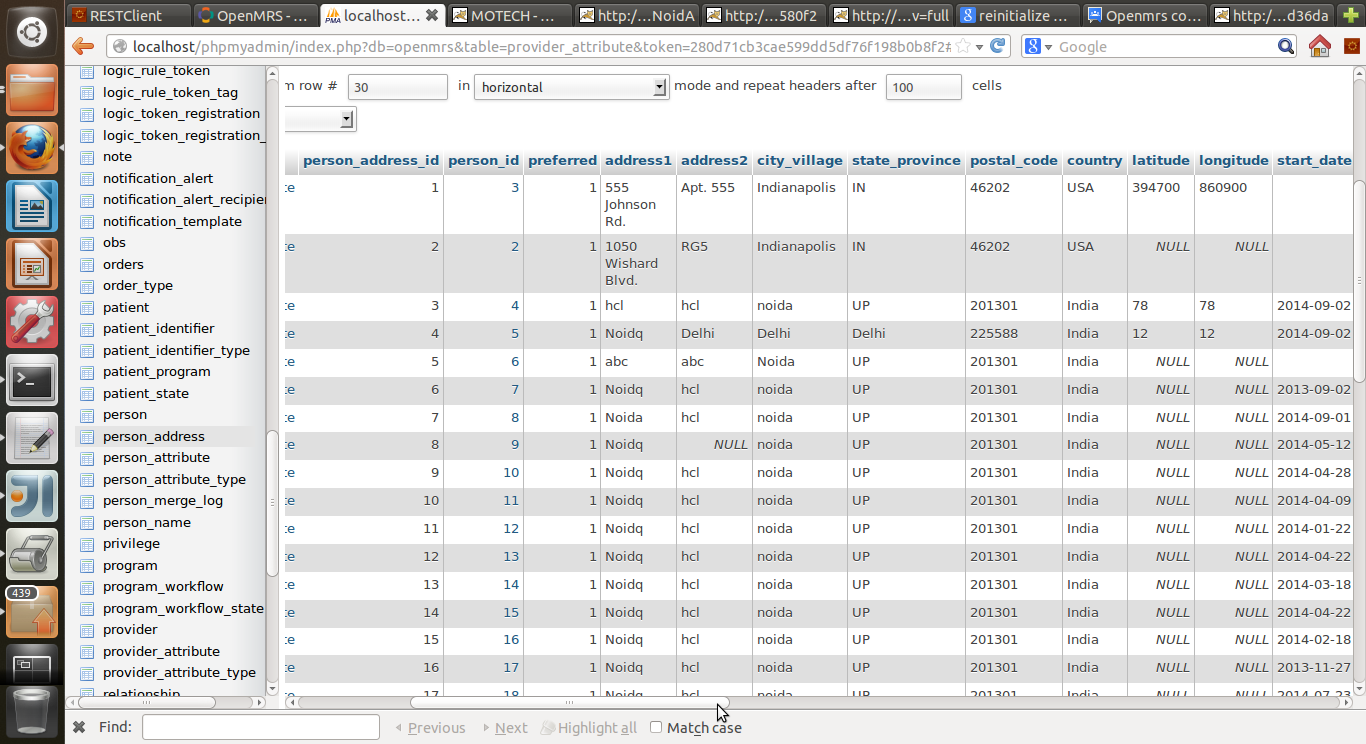
1. **person** – stores id of person (patient, provider (health worker), users)



1. **person\_name** – stores name of a person

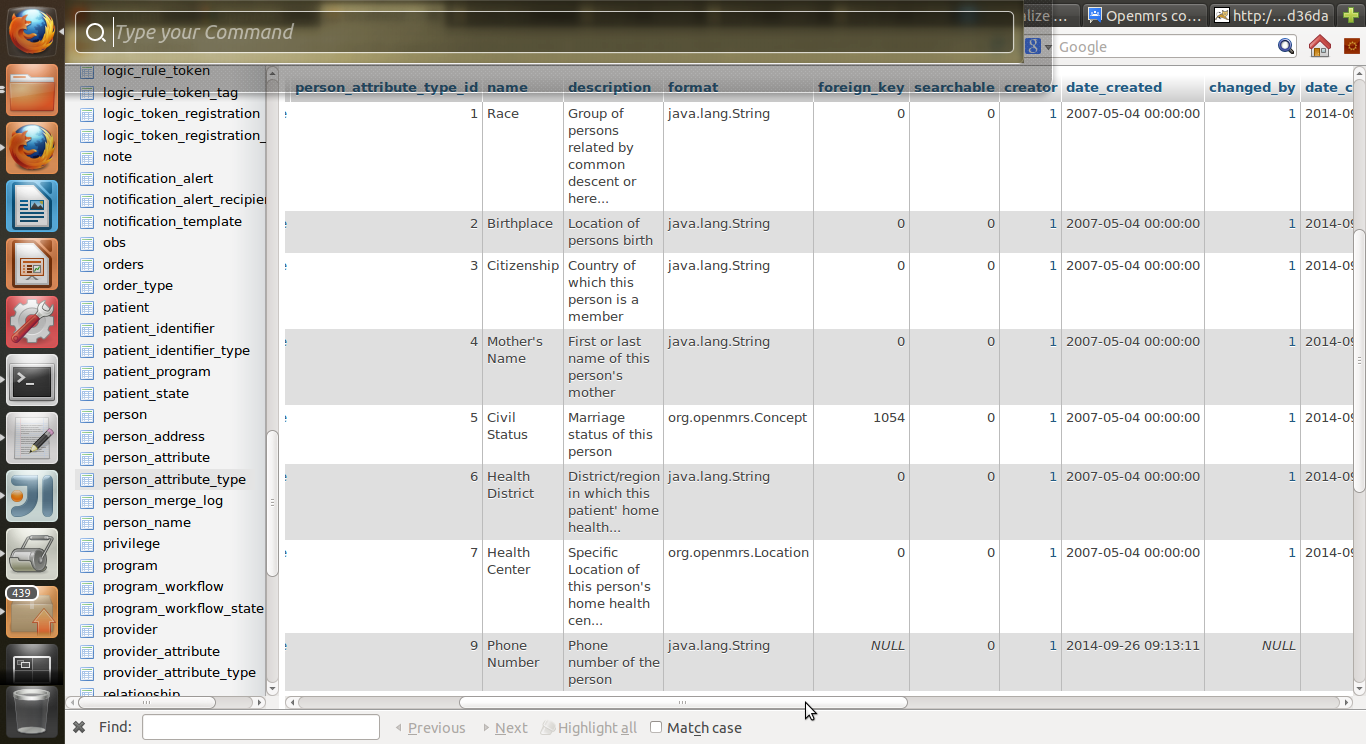


1. **person\_address** – stores address detail of all persons



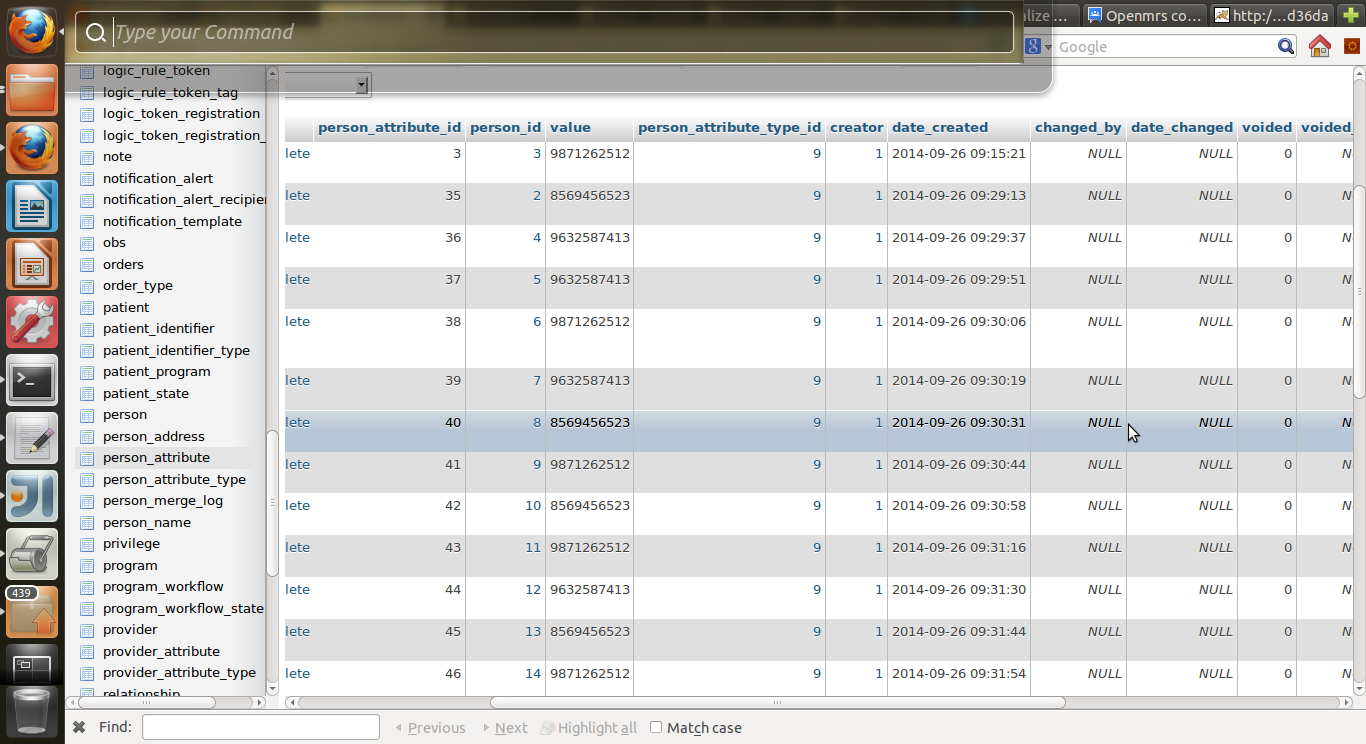
1. **person\_attribute\_type** – Here custom attributes for a person as per the requirement (like phone no, mother tongue etc.) can be added by adding new entry in person\_attribute\_type table.

**IMP!! For ‘*mHealthDataInterface’,* a custom attribute Phone Number is added (below screen shot)**

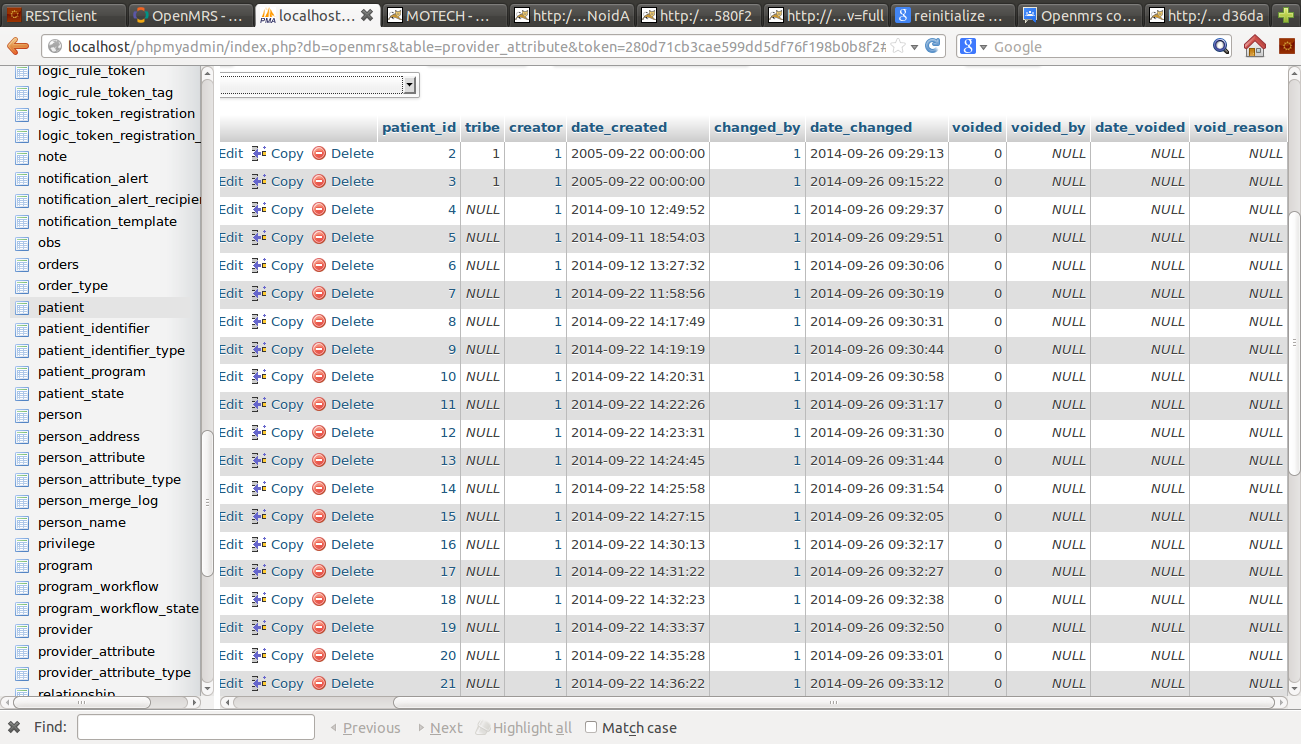
****

1. **person\_attribute** – stores the value of the attribute define against a person in the person\_attribute\_type table

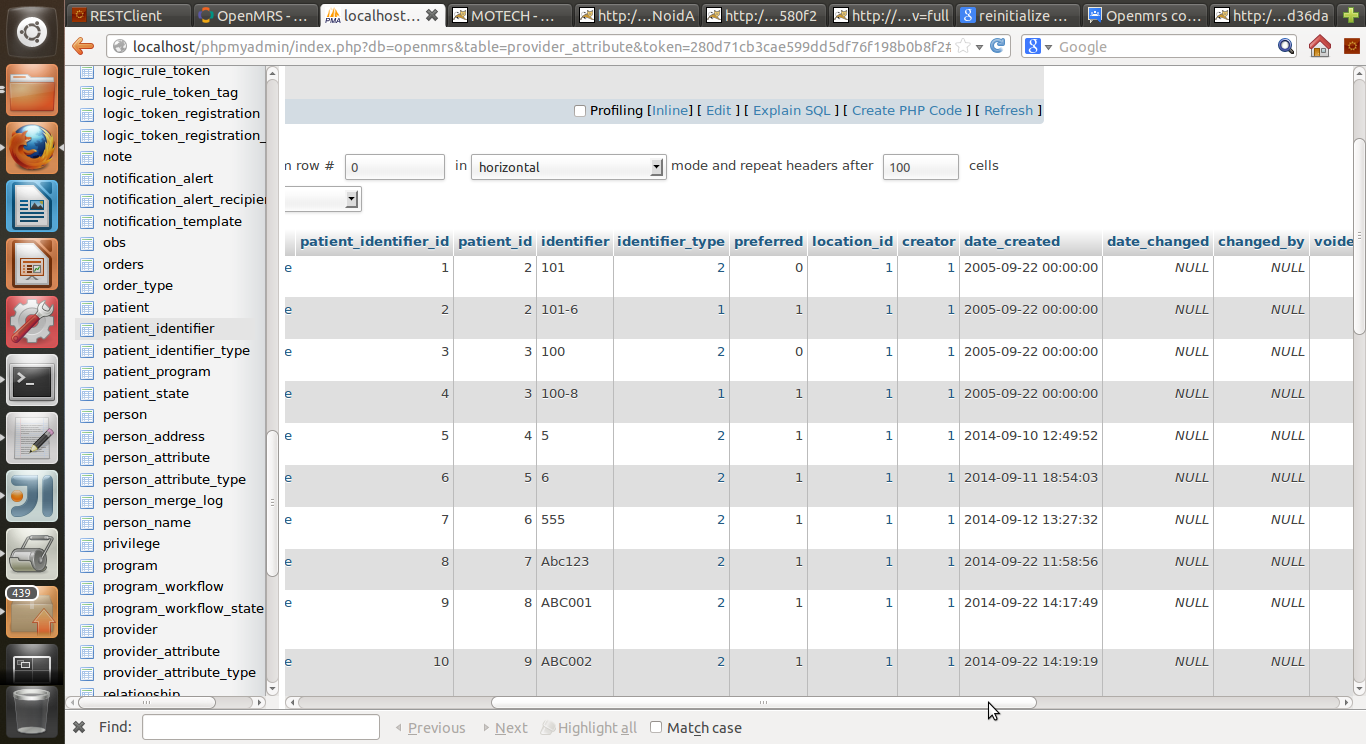
**IMP!! For ‘*mHealthDataInterface’,* new attribute type phone number is mapped to person\_id** (below screen shot)

****

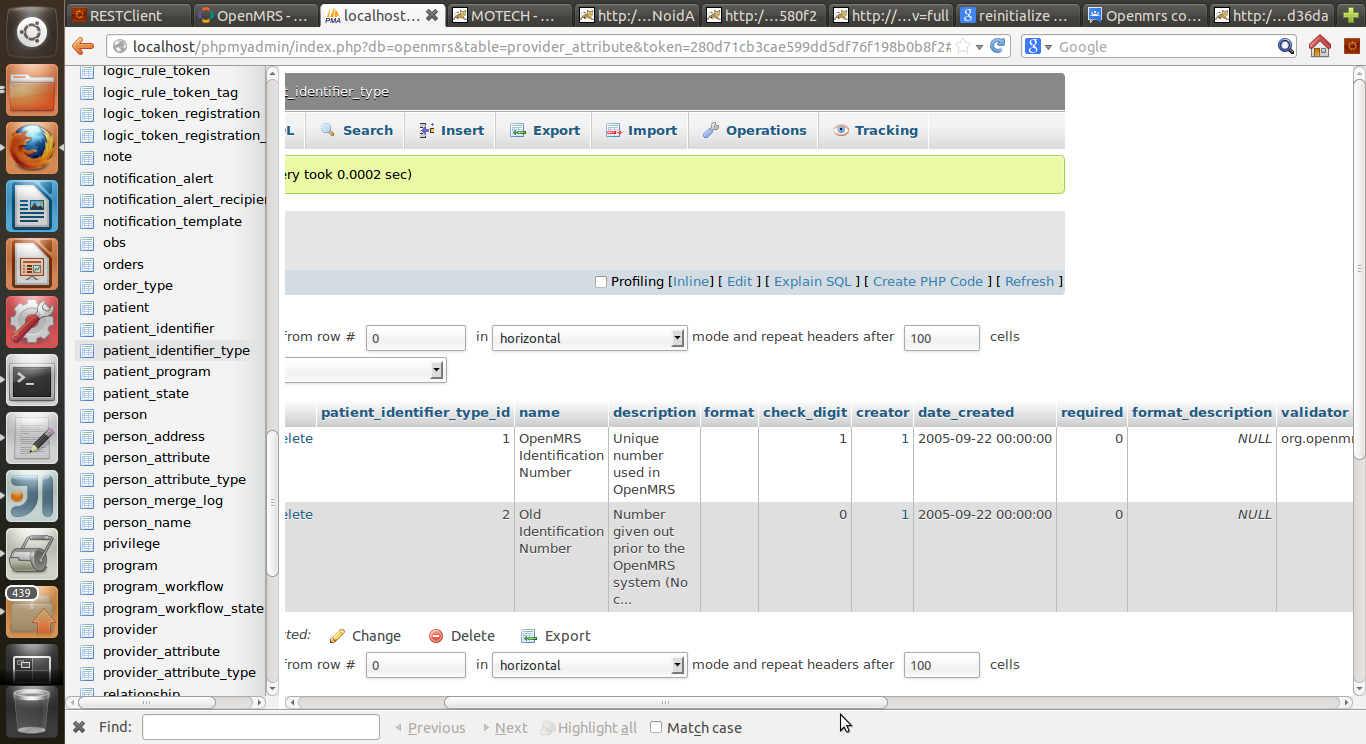
1. **patient** – stores patient detail



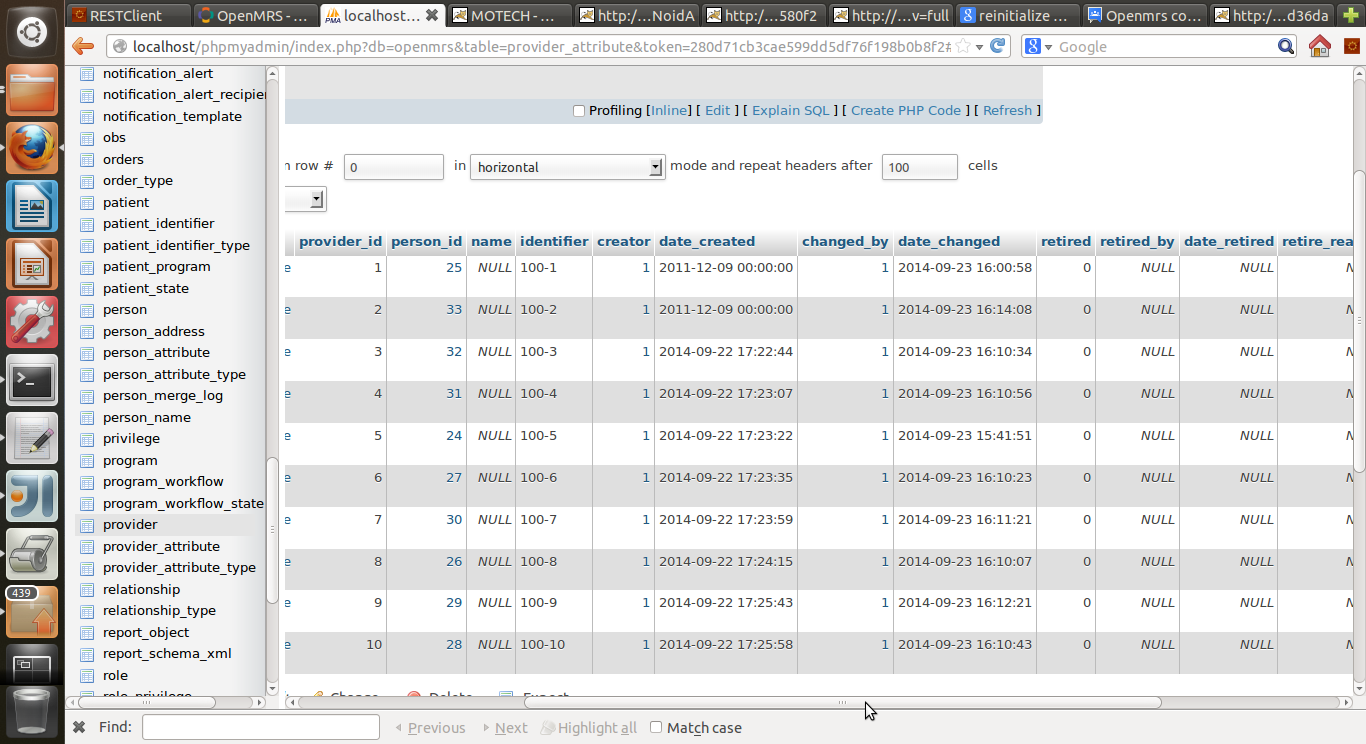
1. **patient\_identifier** –stores a unique identifier of all patients; it has a location linked with it.

****

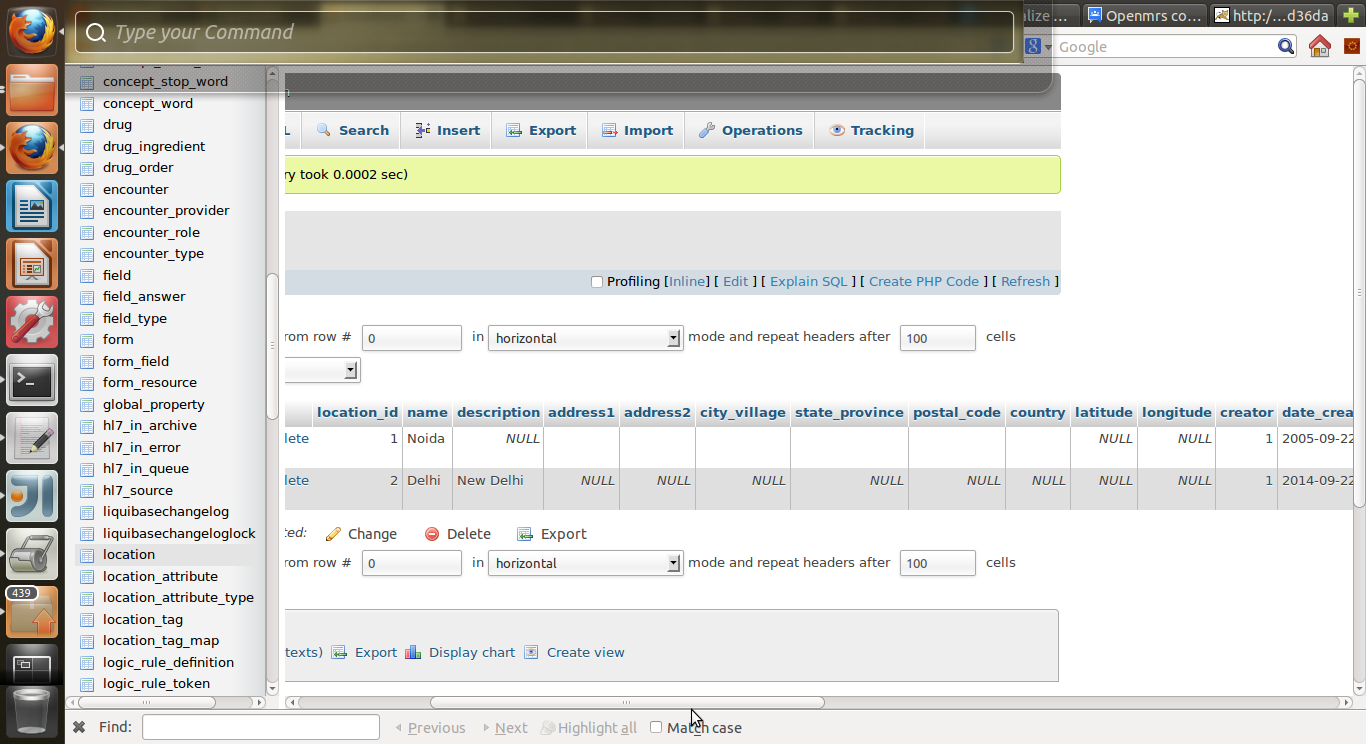
1. **patient\_identifier\_type** – stores all types of identifier

****

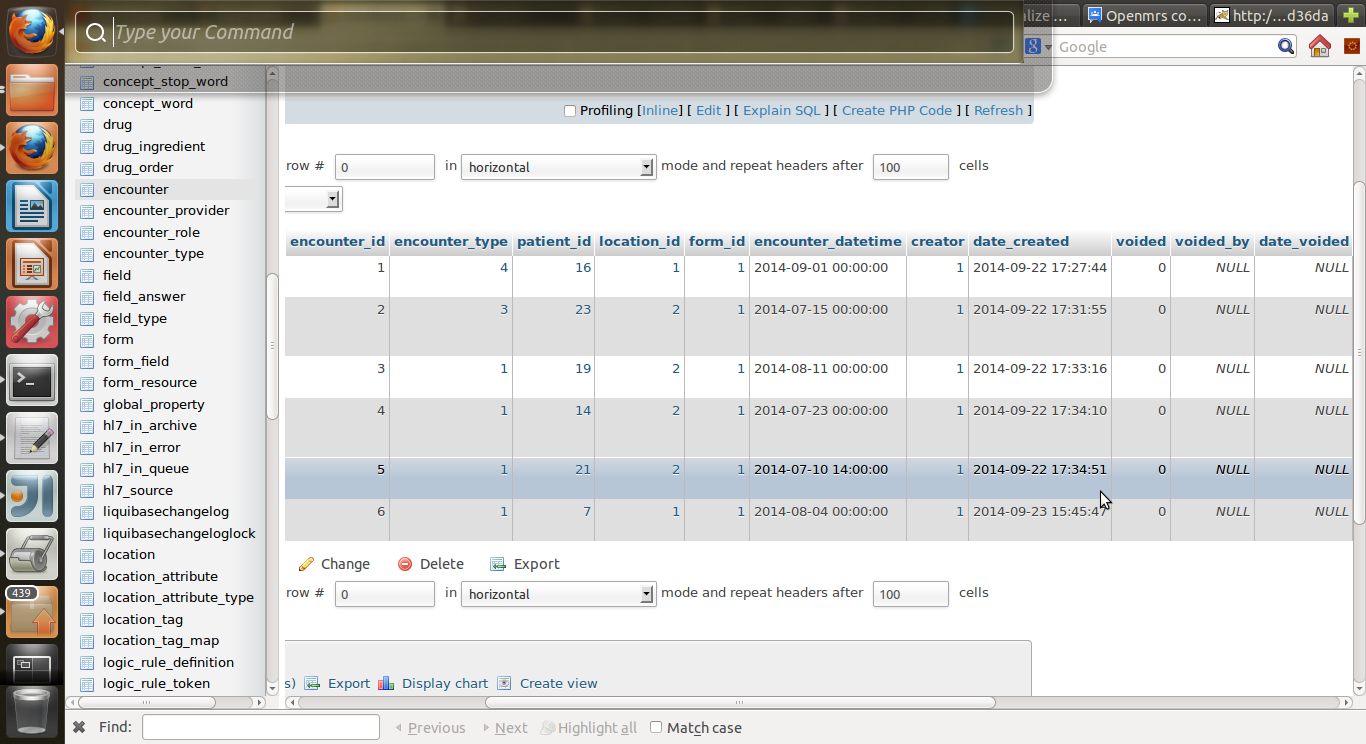
1. **provider** – stores the detail of provider (health worker)

****

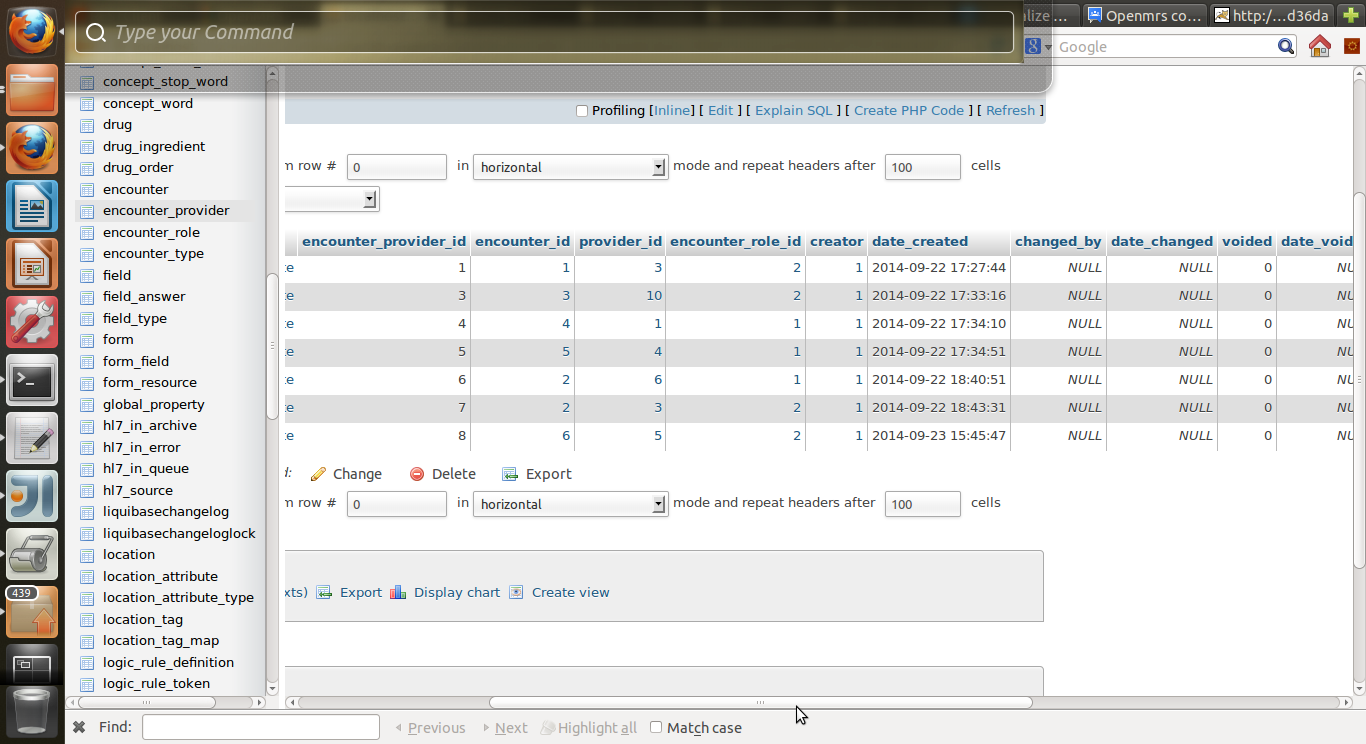
1. **location** - stores detail of locations

****

1. **encounter** – stores all detail of a visit by a health worker to a patient.

****

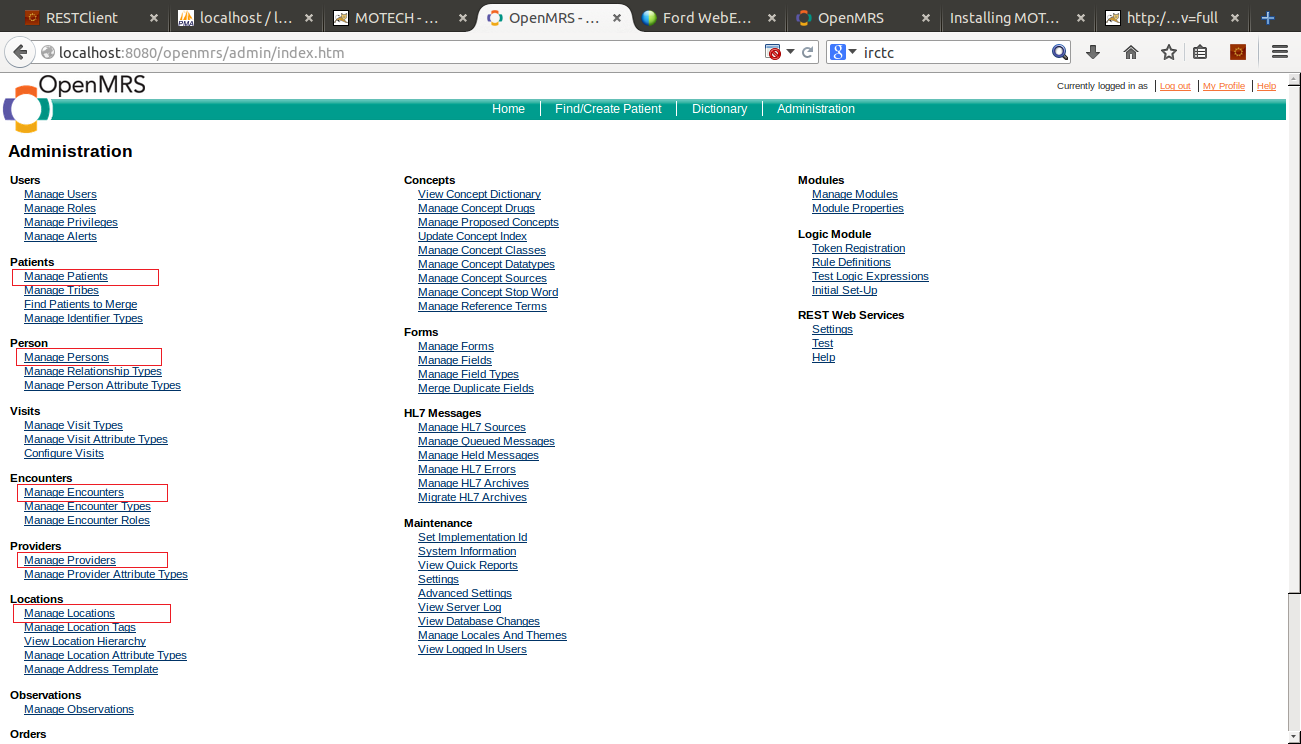
1. **encounter\_provider** – stores the id of the provider (health worker) of a particular encounter(visit)

****

## Populating Data in OpenMRS

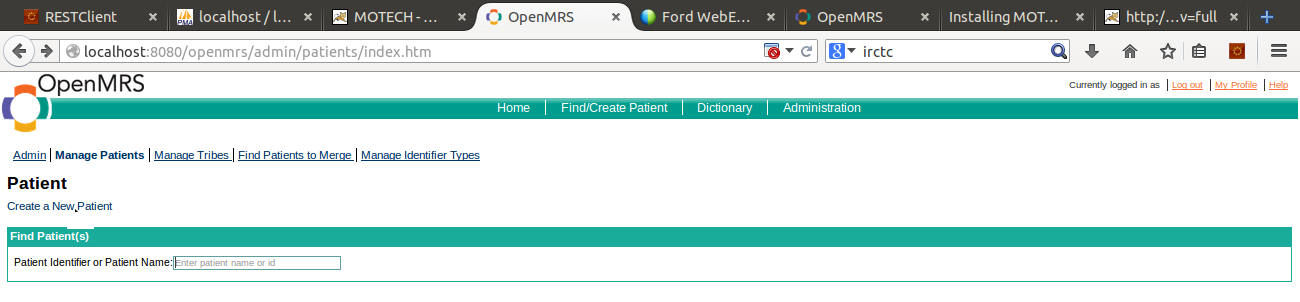
Developer can populate data in OpenMRS database through OpenMRS UI. As mentioned before, OpenMRS is deployed in localhost tomcat server. So to access OpenMRS, go to the URL <http://localhost:8080/openmrs/> . It will open a login page, where developer needs to login through admin credentials (by default: Username: admin / Password: Admin123). Developer while changing the credentials would need to change in code properties accordingly.

To populate data go to “*Administrator*” tab, in the top bar. It looks like as below screenshot:

 The links are given to add different entities, like patient, provider (health worker), person, location and encounter. The entities that are used for ‘**mHealthDataInterface’** , are highlighted in the screenshot. One can use these links to add/create new records and manage data.

For example, the steps to add a new patient are following.

1. First go to “*Administrator*” tab, as in the screenshot in previous page.
2. Click on “*Manage Patients*” under heading “*Patient*”
3. Below screen will appear with a link “*Create a New Patient*”



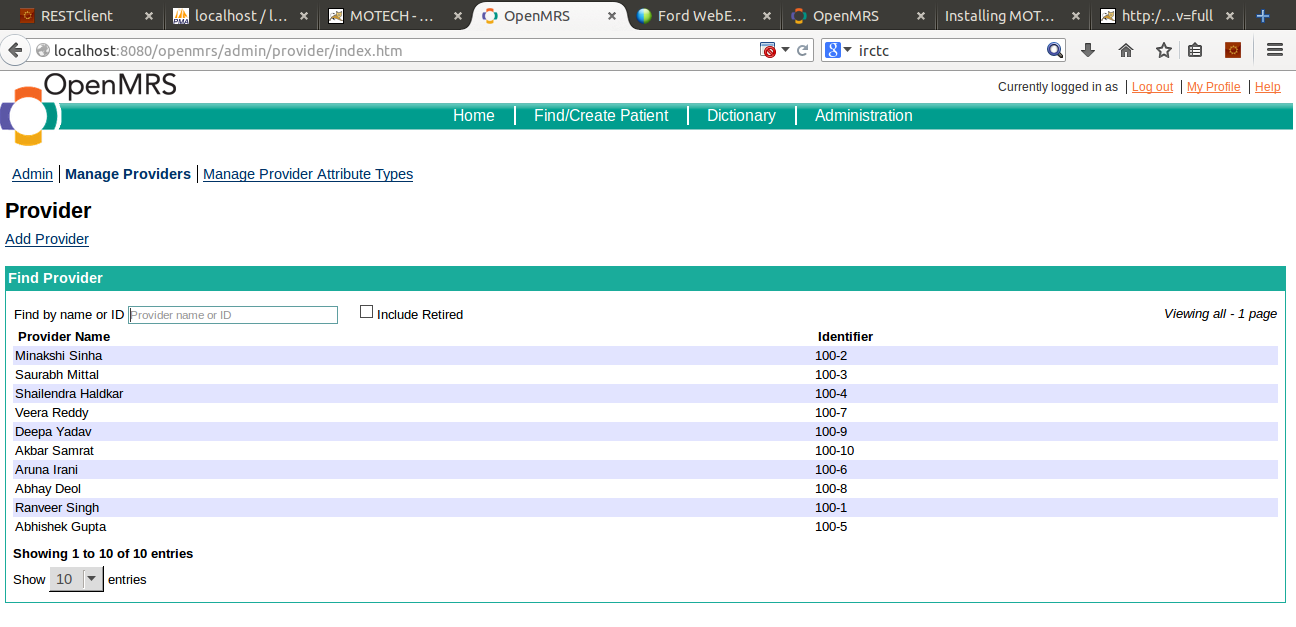
1. Click on the link.
2. Below screen will then appear.



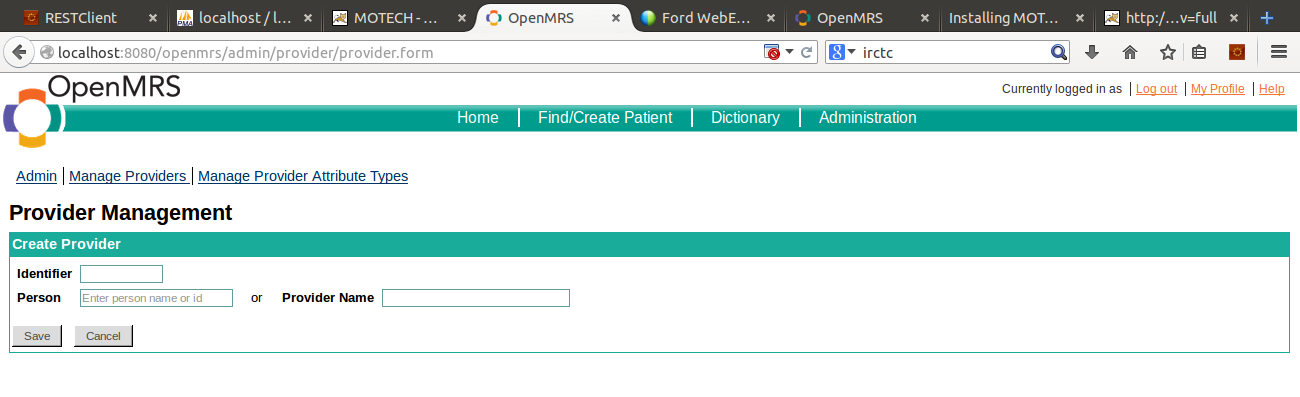
1. Enter all the detail of patient and click on “*Create Person*” button
2. Patient will then be created in the database.

The steps to add a provider (health worker) are as following:

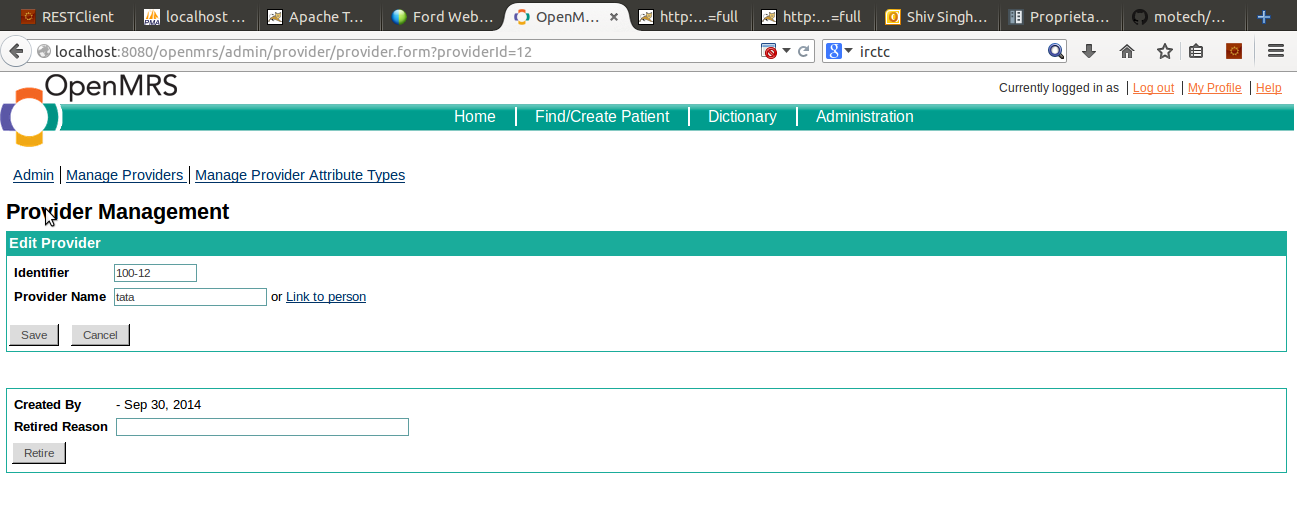
1. First go to “*Administrator*” tab, as in the screenshot in previous page.
2. Click on “*Manage Providers*” under heading “*Providers*”
3. Below screen will appear with a link “*Add Provider*”



1. Click on the link “*Add Provider*”.
2. Below screen will then appear.



1. Enter all detail required.
2. Developer can either give a new provider (health worker) name in “*Provider*” text box or provide an existing person name from the person table to link this provider (health worker) with the given person.
3. Click on the “S*ave*” button
4. A new provider (health worker) will be created.
5. To link a provider (health worker) with a person, follow the below steps
   1. Create a new person
   2. Go to “*Manage Providers*” from “*Administrator*” tab.
   3. The list of all providers (health workers) will appear.
   4. Click on the provider (health worker) that has to be linked with the new person
   5. Below page appears with a link “*Link to Person*”



* 1. Click on the link “*Link to person*”
  2. Enter the name of new person and click on “*Save*” button.
  3. This will map provider (health worker) with person. Person ID will be saved in provider table.

**Note:**  If developer wants to populate data directly into DB tables, then the ER diagram needs to be referred for all foreign key dependencies.