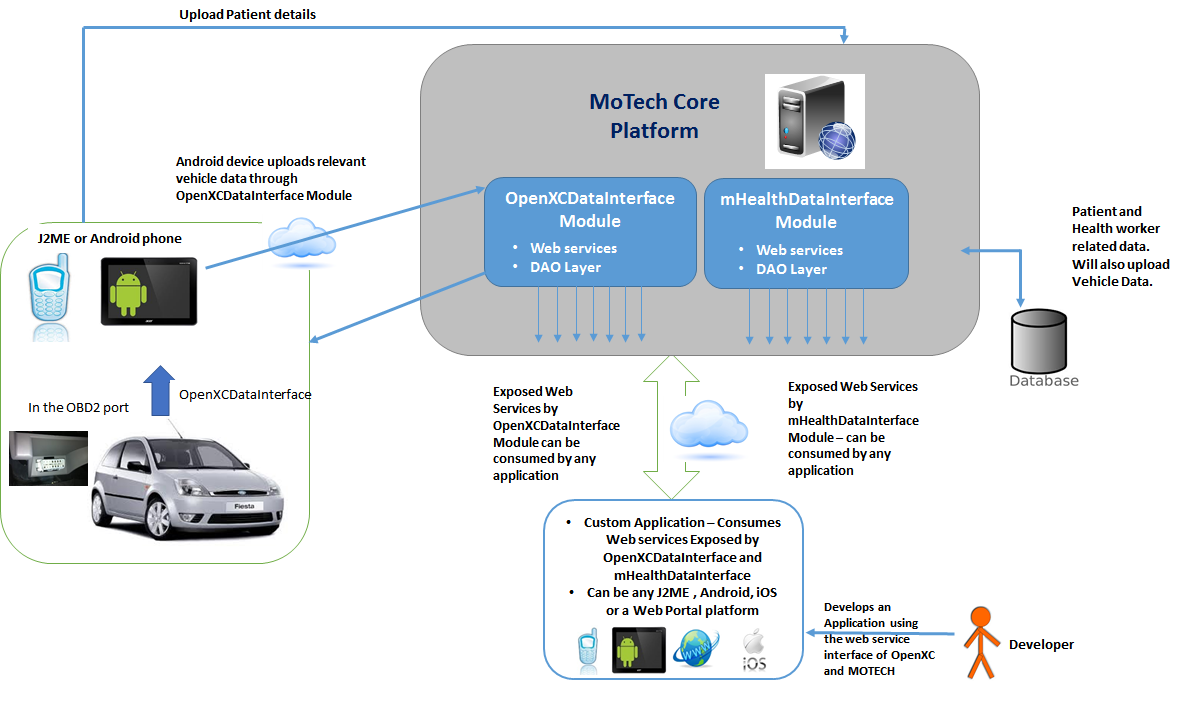
MOTECH-OpenXC Integration - Developer Reference

# Architecture Overview

A pictorial diagram on overview of integration solution –



**Highlights on above architecture**

Above diagram shows two separate new modules of MoTech as **mHealthDataInterface** and **OpenXCDataInterface**.

While travelling in a vehicle, the Health workers will carry a mobile device on which a custom application will remain installed. This custom application will use the APIs exposed from both of our new modules mhealthDataInterface andOpenXCDataInterface and will provide some valuable information to the Health Workers.

* The custom App that runs on any Health Worker’s Android mobile device, can again provide some new features like patient/health worker registration and many more.
* The custom App can use the mHealthDataInterface services for fetching patient details, health worker details, Patient visit details and much other useful information.
* OpenXCDataInterface module will use Vehicle Data stream as received from OpenXC device via OBD2 port VI present in the vehicles, and will upload the same to MoTech database through the provided APIs.
* Using OpenXCDataInterface services, the custom App can also fetch data like vehicle details, current location details, or the route details of a particular vehicle.
* One could use both the web services APIs (from OpenXCDataInterface as well as from mhealthDataInterface) for leveraging vehicle data, Patient’s data, and Health Workers’ data – AND write innovative APIs.

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

# Custom Module (‘OpenXCDataInterface’) on MOTECH

To expose custom APIs that will provide all vehicle data, a custom module named **OpenXCDataInterface** needs to be created. This module has to be added to the MOTECH, above the core platform along with other custom modules, to make APIs available.

**Purpose of OpenXCDataInterface Module:**

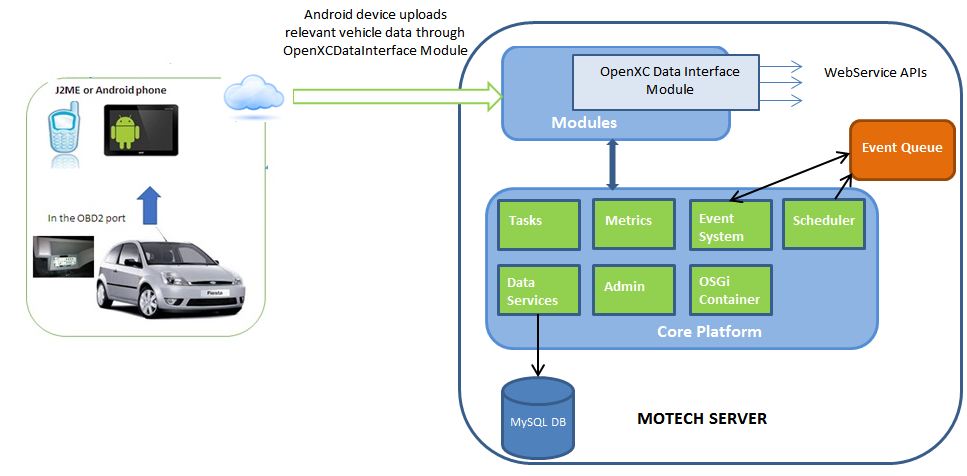
****

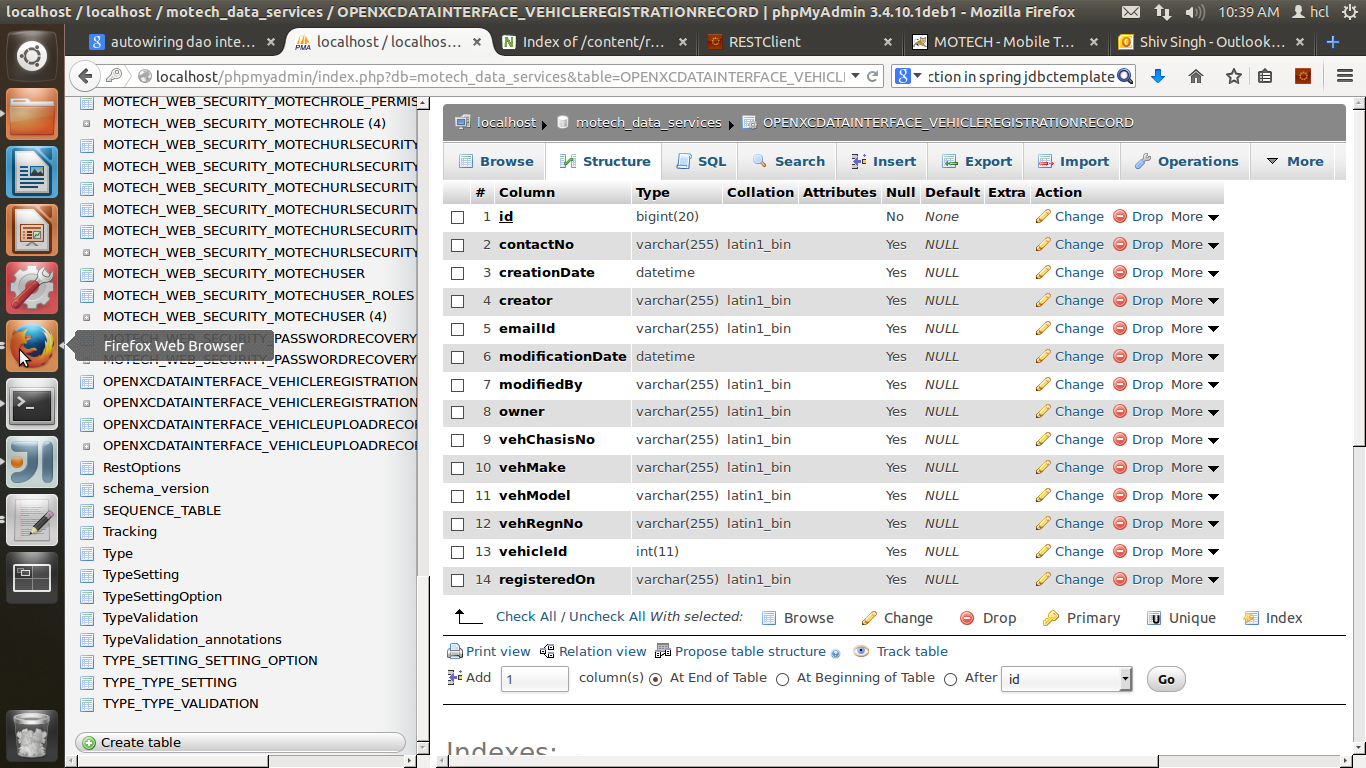
Figure 1. MOTECH Core Platform Architecture (source: <http://docs.motechproject.org/en/latest/architecture/core_architecture.html>)

“First to use this custom module, the vehicle data (like speed, current latitude-longitude) need to be fetched which is received from OpenXC device via OBD2 port VI present in the vehicles. Fetched data can be saved in the custom tables of MOTECH database and then exposed through the web service APIs that will provide these data to consumer (developer) for further usage.”

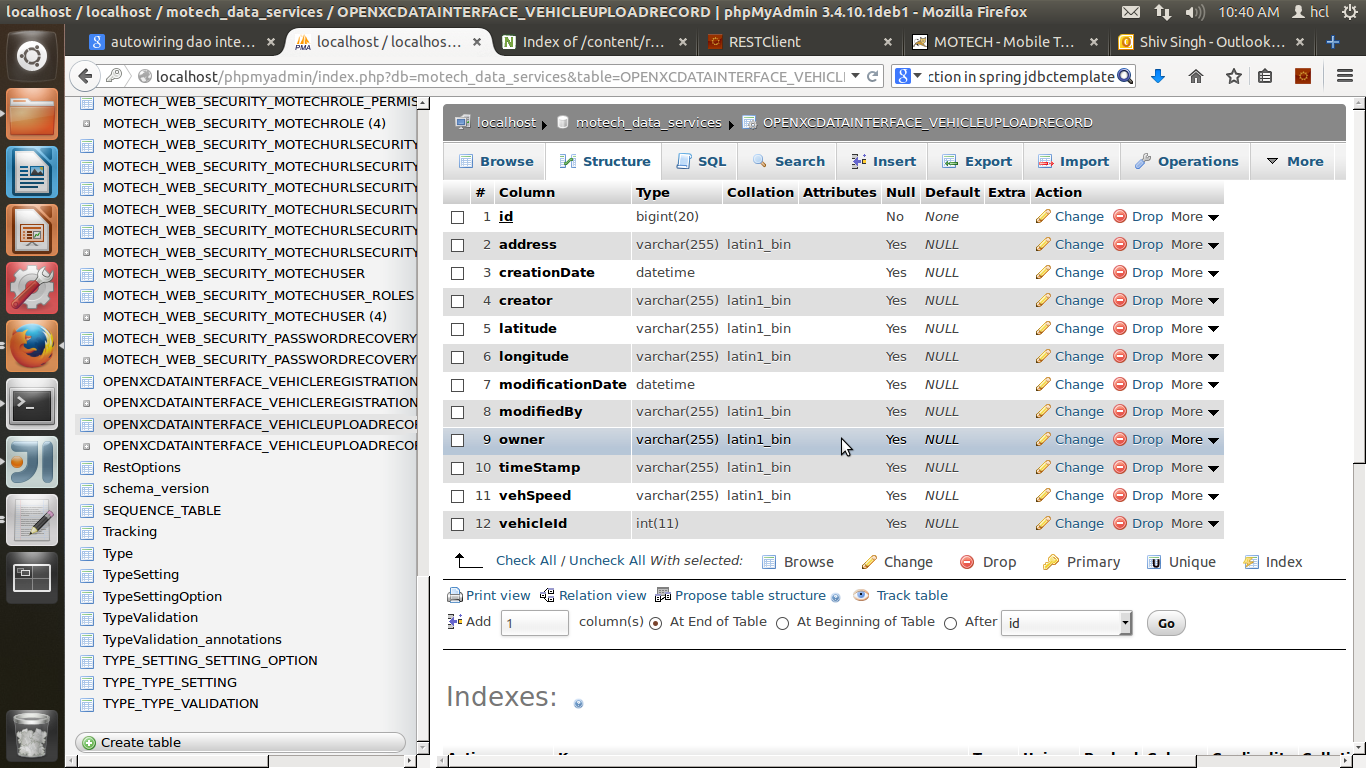
This data is exposed through Spring-based web service APIs.

To save the data, the module uses two tables:

1. OPENXCDATAINTERFACE\_VEHICLEREGISTRATIONRECORD



1. OPENXCDATAINTERFACE\_VEHICLEUPLOADRECORD



These tables get created into the MOTECH database **“motech\_data”**, whenever the module OpenXCDataInterface is installed into the MOTECH Core platform.

Module **OpenXCDataInterface**

* Fetches required vehicle data (like speed, current latitude-longitude) as received from OpenXC device via OBD2 port VI present in the vehicles.
* Saves/Uploads the data in the tables specific to the module, those get created in the MOTECH database while installing the module.
* Exposes the required web service APIs to provide these data for further usage.

# Adding jar of the new module into MOTECH Core Platform

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

* <http://docs.motechproject.org/en/latest/demos/hello_world.html>
* <http://docs.motechproject.org/en/latest/development/archetype.html>

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

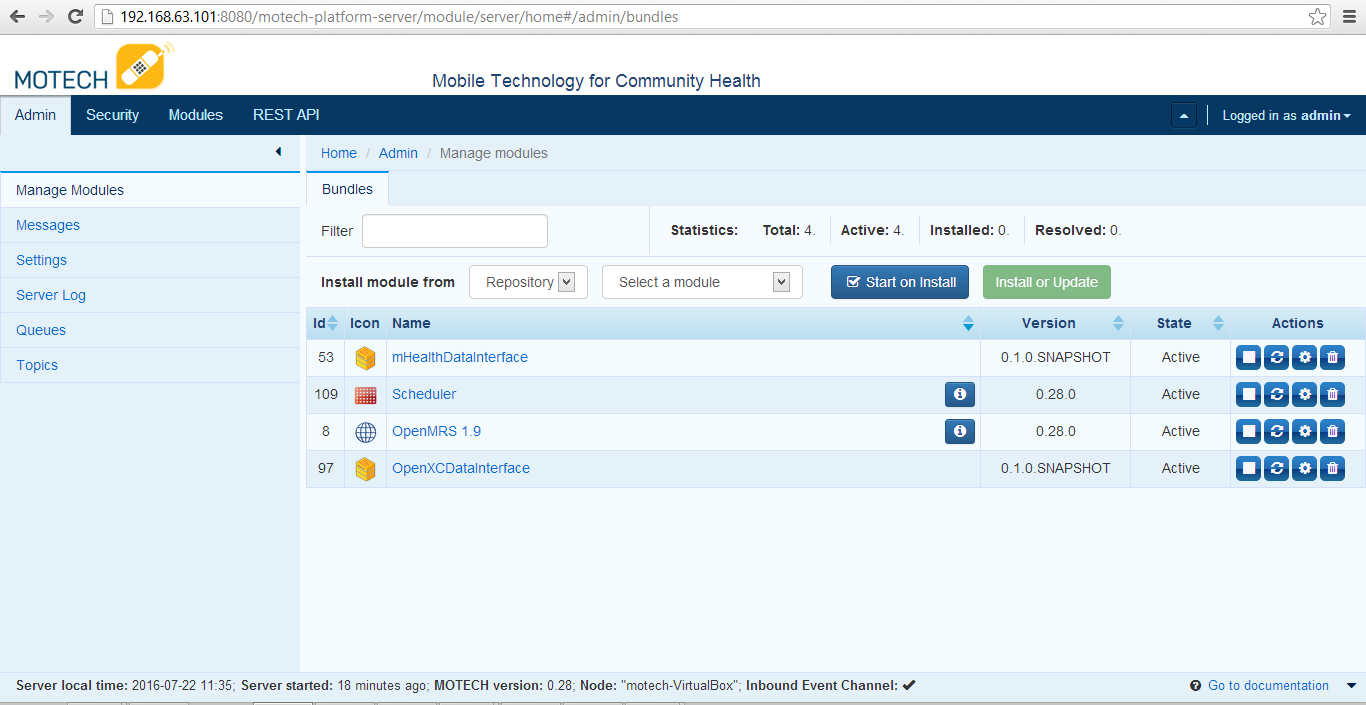
Below source artifacts are provided for developer usage and reference –

1. Jar file ‘**OpenXCDataInterface-0.1-SNAPSHOT.jar** ’
2. ‘**OpenXCDataInterface’** 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 –



# OpenXCDataInterface Module Configuration

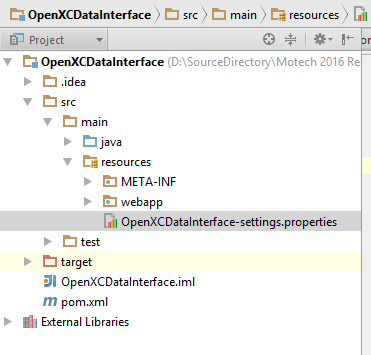
There are two ways that the OpenXCDataInterface module can be configured:

1. **Properties File configuration –**

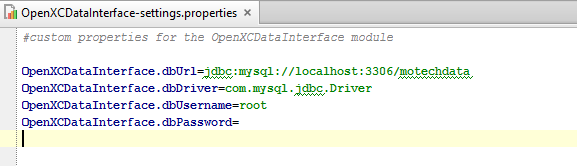
This configuration method provide pre-configured OpenXCDataInterface Module. Developer has to provide all the OPEN MRS related configuration in properties file named “***OpenXCDataInterface-settings.properties***”.

Location of the properties file is as follows –

**OpenXCDataInterface\src\main\resources\OpenXCDataInterface-settings.properties**



Properties File SNAPSHOT:

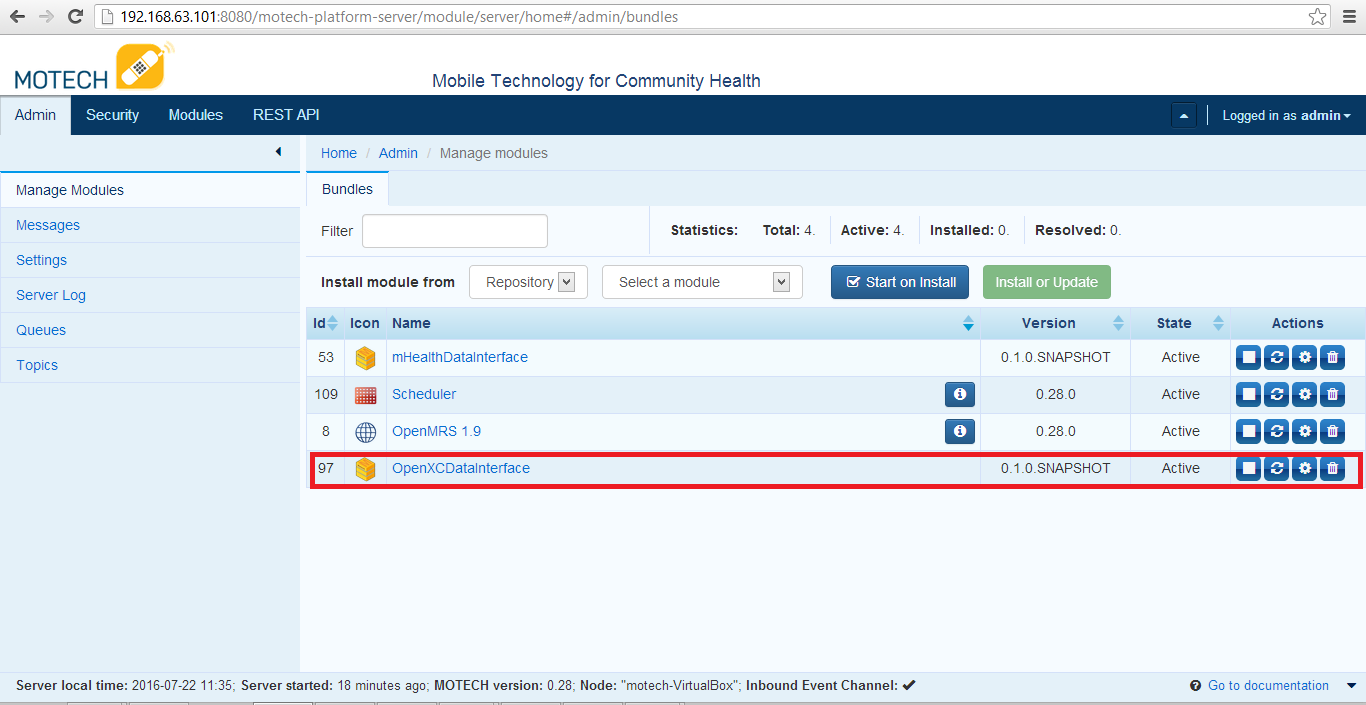


**Important:** Above settings are to be provided before building the OpenXCDataInterface module, so that it can be pre-configured with output jar file.

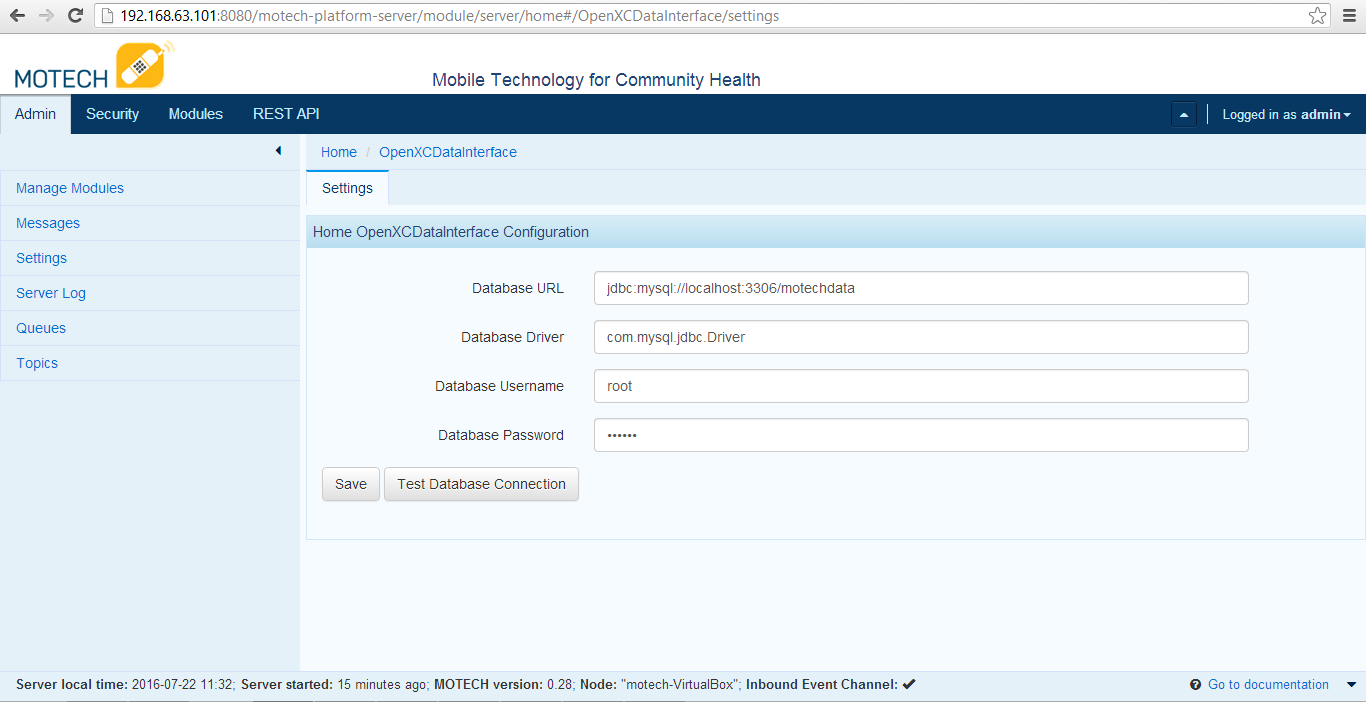
1. **OpenXCDataInterface Configuration using MOTECH module Management in Admin Panel –**

There are following steps to configure OpenXCDataInterface using this method.

* 1. Login to the MOTECH Portal and go to “*Admin >> Manage Modules*”.
  2. This screen has the list of all modules installed on MOTECH Platform Server like as following screenshot.



* 1. Click on Setting button Highlighted in above screen at right “Action” column of the Module list. This action takes user to Configuration screens shown on following screenshot.

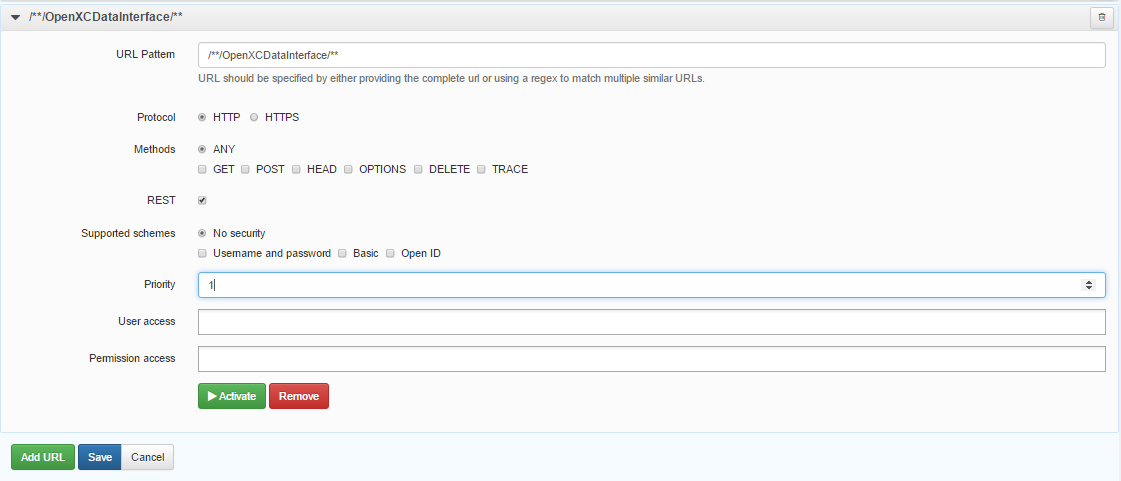


* 1. Configuration screen provides functionality to test the data connection properties with “Test Database Connection” button. After any modification in connection properties user can test the database connection properties before saving.
  2. Configuration screen provide functionality to save the modified settings with “Save” button. These settings are reflected in the system at run time so there is no need to restart the server.

# Motech Security – Manage dynamic URLs

This is required, if there is need of Rest Webservices is to be accessed with specific security rule for a particular REST URL. mHealthApp is using no security for REST Webservices APIs, So we need to configure / manage the dynamic URL of REST APIs with no security. There are following steps needed for that –

1. Login to MOTECH Platform Server.
2. Go to Security > Manage Dynamic URLs
3. Click on Add URL Button, it will open a form like as follows –



1. Enter the required Items as per above screen. URL should be specified by either providing the complete url or using a regex to match multiple similar URLs. In our case we are using regex for OpenXCDataInterface module’s REST APIs. URL Pattern for OpenXCDataInterface is –

/\*\*/OpenXCDataInterface/\*\*

1. Click on Activate then save the form to reflect the changes in system.

# APIs Exposed From OpenXCDataInterface Module on MOTECH

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

1. ***registermHealthVehicle ( )***

***ref url: http://localhost:8080/motech-platform-server/module/OpenXCDataInterface/ registermHealthVehicle***

|  |  |  |  |
| --- | --- | --- | --- |
| ***registermHealthVehicle*** | | | |
| **Protocol** | HTTP - REST API (POST) | | |
| **Description** | Register a vehicle using the web service API registermHealthVehicle (). | | |
| **Input Parameter** | Data Format : JSON data | ***Sample JSON ------***  *{"vehRegnNo":"AH17FT2387","vehChasisNo":"QW5T2Y8233","vehMake":"Honda","vehModel":"Jazz","contactNo":"9977657890","emailId":"mary@mail.com"}* | |
| vehRegnNo : String | |
| vehChasisNo : String | |
| vehMake : String | |
| vehModel : String | |
| contactNo : String | |
| emailId : String | |
| **Output** | Data in Json format. | ***Sample JSON ------***  *{ "vehicleId": "100004","errorMsg":"", "responseMsg": "Y"}*  responseMsg will only have values Y/N/E.  Y – Data successfully inserted  N – Exception occurred while inserting the data  E – Vehicle Registration Number already exists |  |
| ***vehicleId*** | : String |
| ***responseMsg*** | : String |
| ***errorMsg***(will be blank if no error) | : String |

1. ***getRegisteredmHealthVehicleList ( )***

***ref url: http://localhost:8080/motech-platform-server/module/OpenXCDataInterface/ getRegisteredmHealthVehicleList***

|  |  |  |  |
| --- | --- | --- | --- |
| ***getRegisteredmHealthVehicleList*** | | | |
| **Protocol** | HTTP - REST API (POST) | | |
| **Description** | Displaying the list of all the registered vehicles using the web service exposed API getRegisteredmHealthVehicleList (). | | |
| **Input Parameter** |  | ***No input data required*** | |
| **Output** | Json Array. | ***Displays all registered vehicle list which includes (vehicleId, vehRegnNo, vehChasisNo, vehMake, vehModel, contactNo, emailId, registeredOn)*** |  |

1. ***getmHealthVehicleId ( )***

***ref url: http://localhost:8080/motech-platform-server/module/OpenXCDataInterface/ getmHealthVehicleId/{ vehicleRegistrationNo }***

|  |  |  |  |
| --- | --- | --- | --- |
| ***getmHealthVehicleId*** | | | |
| **Protocol** | HTTP - REST API (POST) | | |
| **Description** | Get registered vehicle ID using the web service exposed API getmHealthVehicleId (). | | |
| **Input Parameter** | Passed as a parameter in URL | ***vehicleRegistrationNo*** | |
| **Output** | Data in Json format. | ***Returns Vehicle ID as Json*** |  |

1. ***getmHealthVehicleLocation ( )***

***ref url: http://localhost:8080/motech-platform-server/module/OpenXCDataInterface/ getmHealthVehicleLocation /{ vehicleId}***

|  |  |  |  |
| --- | --- | --- | --- |
| ***getmHealthVehicleLocation*** | | | |
| **Protocol** | HTTP - REST API (POST) | | |
| **Description** | Get current location of a vehicle using the web service exposed API getmHealthVehicleLocation(). | | |
| **Input Parameter** | Passed as a parameter in URL | ***vehicleId*** | |
| **Output** | Data in Json format. | ***Returns current location (latitude, longitude, address) of a vehicle as Json*** |  |

1. ***getAllmHealthVehiclesLocation ( )***

***ref url: http://localhost:8080/motech-platform-server/module/OpenXCDataInterface/ getAllmHealthVehiclesLocation***

|  |  |  |  |
| --- | --- | --- | --- |
| ***getAllmHealthVehiclesLocation*** | | | |
| **Protocol** | HTTP - REST API (POST) | | |
| **Description** | Get the list of all vehicle locations using the web service exposed API getAllmHealthVehiclesLocation(). | | |
| **Input Parameter** |  | ***No input data required*** | |
| **Output** | Data as Json Array. | ***Returns all vehicle locations (vehicleid, latitude, longitude, address) list as JsonArray*** |  |

1. ***getmHealthVehicleRoute ( )***

***ref url: http://localhost:8080/motech-platform-server/module/OpenXCDataInterface/ getmHealthVehicleRoute/{vehicleId}***

|  |  |  |  |
| --- | --- | --- | --- |
| ***getmHealthVehicleRoute*** | | | |
| **Protocol** | HTTP - REST API (POST) | | |
| **Description** | Vehicle traversed route is displayed in latitude and longitude using the web service exposed API getmHealthVehicleRoute (). | | |
| **Input Parameter** | Passed as a parameter in URL | ***vehicleId*** | |
| **Output** | Data as Json Array. | ***Returns Vehicle route (vehicleid, latitude, longitude, address) as Json Array*** |  |

1. ***uploadmHealthVehicleData ( )***

***ref url: http://localhost:8080/motech-platform-server/module/OpenXCDataInterface/ uploadmHealthVehicleData***

|  |  |  |  |
| --- | --- | --- | --- |
| ***uploadmHealthVehicleData*** | | | |
| **Protocol** | HTTP - REST API (POST) | | |
| **Description** | Upload location parameter and vehicle speed through uploadVehicleData () method of web service. | | |
| **Input Parameter** | Data Format : JSON data | ***Sample JSON ----***  *{"vehicleId":"100021","vehSpeed":"66.7","latitude":"28.580129333333335","longitude":"77.29830173333333","timeStamp":"1406195550"}* | |
| vehicleId : String | |
| vehSpeed : String | |
| latitude : String | |
| longitude : String | |
| timestamp : String | |
| **Output** | Data in Json format. | ***Sample JSON ----***  *{ "responseMsg": "Y","errorMsg":""}*  responseMsg will only have values Y/N/NE.  Y – Data successfully inserted  N – Exception occurred while inserting the data  E – Vehicle ID does not exists |  |
| ***responseMsg*** | : String |
| ***errorMsg***(will be blank if no error) | : String |

# Testing APIs through REST Client

To test these APIs of ‘**OpenXCDataInterface’** 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:

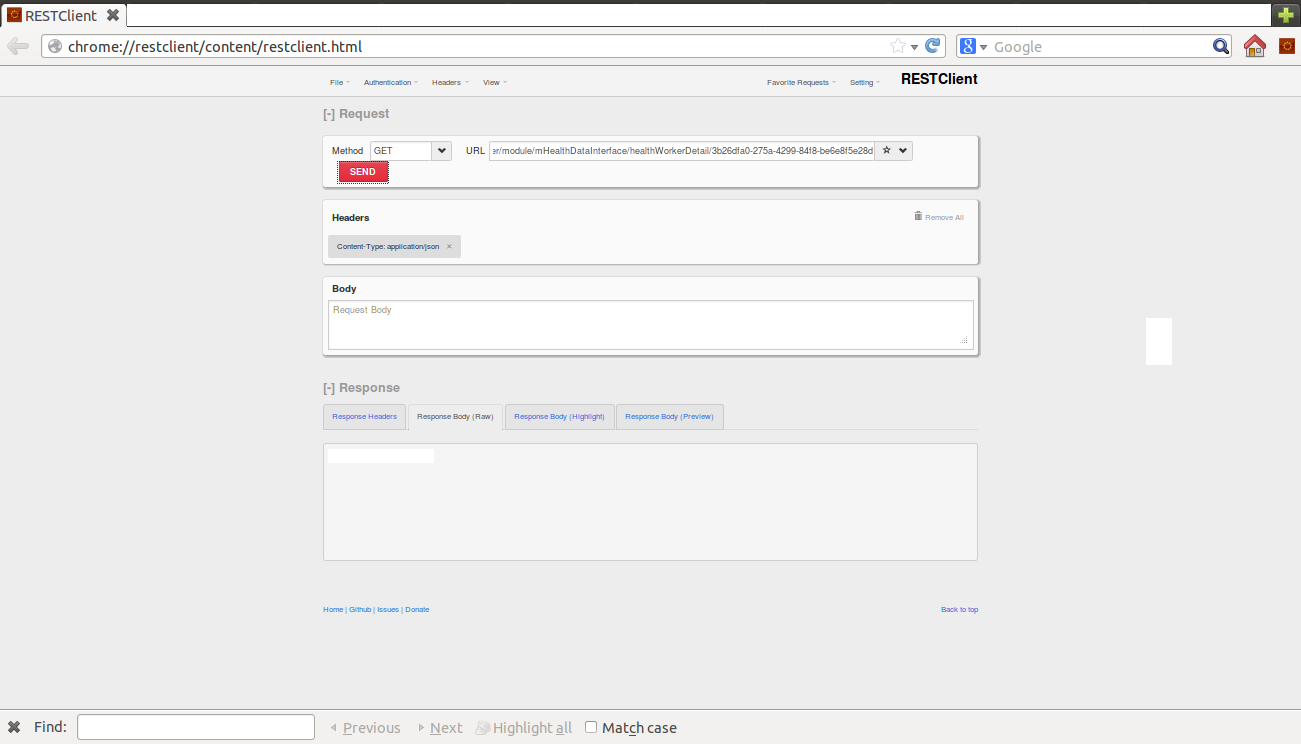
* Chrome :

<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.