Case Based Reporting Module

The HIV Case Based Reporting Module is a proof of concept and it should **NOT** be used in production. It is compatible with Reference Application 2.4 and above.

Background

This module was created as part of a CDC-funded project to demonstrate the power of driving public health decisions directly from data generated on the front lines of care, using HIV Case Based Reporting within OpenMRS integrated with an HIE as an initial use case. By design, the module allow trigger events to be defined as SQL queries. For HIV, these triggers include things like new HIV diagnosis, new HIV disease, evidence of treatment failure, lost to follow-up, etc. When trigger events are detected for a patient, the patient and associated event are added to a case report queue. A Surveillance Officer at the site reviews the queue and is able to easily submit case reports directly from the queue. Case reports are formatted into CDA documents so they can be fed directly into an Health Information Exchange (HIE).

Goal

The goal of this module is to serve as a demonstration of patient-level data feeding directly into an HIE to drive improved public health decision-making.

Champions

- Unknown User (terry)
- Burke Mamlin
- Wyclif Luyima
- Unknown User (jthomas)

Analysis

- Shared Health Record: Case Based Reporting- Use Cases and Requirements
- OpenHIE Save Patient Level Clinical Data Analysis
- Data Integration with OpenHIE Registries - REVIEW
- Security and Privacy - DRAFT

Requirements

- Reference Application 2.4 or later
- A destination for case reports (e.g., an HIE). In our demonstrations, we use OpenHIE.

Source Code

- Source code is available on GitHub at openmrs-module-casereport

User Guide

Also available here.

Also available here.

Integration With OpenHIE

These analyses are a work-in-progress and open to public comment.

Requirements

- Operating System: Ubuntu 14.04
- A running instance of OpenMRS with the case reporting installed, if not you can follow the OpenMRS installation guide and to install the module see this guide.

OpenHIE Architectural Overview

The OpenHIE website has a more detailed installation guide here, this is more like a summarized version that is more specific to help one set up an instance quickly.

In this guide we are going to see how to setup an OpenHIE instance comprised of a couple of systems interacting with each other, when everything has been setup we will be able to generate a case report in one system and send it to its intended destination (SHR) with the help of the all the components, below are the components that will be involved.

Interoperability Layer (IOL)

> Receives all communications from services within a health geography, and orchestrates message processing among the external systems and the OpenHIE component layer. OpenHIM is the implementation that we will be using for this, for more on the IOL see here

Shared Health Record (SHR)

> This is a repository containing the normalized version of content created within the community, after being validated against each of the previous registries. It is a collection of person-centric records for patients with information in the exchange. In this guide, we'll be using a combination of an OpenMRS instance and OpenXDS, OpenMRS being the patient data store and OpenXDS the document store, OpenXDS implements the IHE XDS profile, for more on the SHR see here

Client Registry (CR)

> An enterprise master patient index (EMPI) that manages the unique identity of the patients receiving health services at the various health facilities, it's the source of truth for patient identifiers. In this guide we are going use OpenEMPI, for more on the CR see here

External System

> Any external system used by clinicians and by community health workers to access and update a patient's person-centric shared health information and to record healthcare transactions, this is typically a point of care system (POC) running at any health center and there could be several in the HIE. In our setup we will have another OpenMRS instance which is different from that of the SHR we saw earlier, it is where the case reporting module is installed therefore it will be the source of the case report documents.

To get more details about OpenHIE visit this page

> ⚠ Note that in this guide we chose the particular software implementations for the different components in the HIE but you can install other implementations.
>
> **You also want to make sure that communication between the systems is over https**

Case Based Reporting Workflow Summary with OpenHIE Integration

A case report is generated for a patient and submitted from the OpenMRS based POC system to the interoperability layer, the document is submitted as a SOAP message based on the XDS profile with a CDA document embedded inside it. The IOL routes the request to the XDS.b mediator registered with it whose work is to enrich the document with an enterprise identifier of the patient, it does this by making a call to the client registry for the patient's enterprise ID and overwrites the local patient ID in the document with the enterprise one and then forwards the message to the SHR, the SHR stores the metadata associated to the document in the document registry (OpenXDS) and saves a copy of the original CDA document in the document repository, it also parses the embedded CDA document to extract any discrete data and saves in its data store.

Installation using docker

You can setup the entire stack of software using docker, for more details see.

Otherwise you can install the individual components, please continue reading.

Install the SHR

Run the commands below in the terminal, for more details see the setting up the openshr on ubuntu from a ppa 1404 trusty guide.

> ⚠ Only provide mysql root passwords and leave the default values including blank ones for the rest of the prompts, of course you need to accept the Oracle java license. When the installer is running it might prompt you on the command line to enter this mysql root password along the way.

```
sudo add-apt-repository ppa:webupd8team/java
sudo add-apt-repository ppa:openhie/release
sudo apt-get update
sudo apt-get install openshr
```

The SHR instance can take a while to start therefore you might have to wait for a couple of minutes, for a non production installation you can speed up the start up process by adding the text below to the value of the JAVA_OPTS environmental variable in the **/etc/init/openshr-rep.conf** file:

```
-Djava.security.egd=file:/dev/urandom
```

To access the OpenMRS instance from the browser you can go to this url **http://host_ip_address:8080/openmrs.** Assuming you kept the default values the same for the OpenSHR prompts, the OpenMRS username and password are **admin** and **OpenSHR#123** respectively, you'll definitely have to change the password for production installations.

There is a file that contains the codes from various coding systems for the metadata that is required by the SHR in order for it to receive documents from other systems in the HIE, it's located at ***/usr/share/openshr/openxds/conf/actors/XdsCodes.xml,*** there is some extra metadata codes that you need to add to this file before you can submit case reports to the HIE. You can obtain an already updated copy of this XdsCodes.xml to replace the original one and then restart the machine. The practiceSettingCode and healthFacilityTypeCode configured in the CBR must exist in the XdsCodes.xml.

You **MUST** restart the machine where this SHR is installed for the contents of the new XdsCodes.xml to be loaded.

You MUST set the default locale to English rather than

Install OpenHIM (core and console apps)

Run the commands below in the terminal, for more details see installing-the-openhim-core-and-console guide, this PDF contains the full OpenHIM documentation.

- Keep the default values for the prompts.
- For a demo installation you don't have to set up a TLS certificate to secure your server.

```
sudo add-apt-repository ppa:openhie/release
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
sudo echo 'deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.2 multiverse' | sudo tee /etc/apt
/sources.list.d/mongodb-org-3.2.list
sudo apt-get update
sudo apt-get install openhim-core-js openhim-console
```

Configuring openhim-core

Open the ***/etc/openhim/config.json file*** in an editor and set the following properties:

- You need to enable logging which is based on the ATNA profile, set the value of the ***auditing*** -> ***servers*** -> ***tcp*** -> ***enabled*** field to true.
- Because the tomcat instance installed as part of the SHR above is already listening on port 8080 unless you have installed them on different machines otherwise you'll need to reconfigure openhim-core to listen on another available port by editing the value of the ***api*** -> ***httpsPort*** field.

Restarting openhim-core

Run the command below:

```
sudo service openhim-core restart
```

Reconfiguring openhim-console

If you haven't yet, update openhim-console to use the new port you set above that openhim-core api is listening on. Also note that even when openhim-core is installed on the same machine as the console application, you need to set the host value to an IP address rather than localhost. To reconfigure the console application, below is the command to execute:

```
sudo dpkg-reconfigure openhim-console
```

Following the installing-the-openhim-admin-console guide, skip past the installation section since you already installed it in the previous step, find the section for how to access openhim-console and how to authenticate, you should be able to use the openhim-console in your web browser after this step at http:server_ip

Install The Client Registry

In this guide we use OpenEMPI as the client registry but there is other implementations like MEDIC-CR and OpenPIXPDQ.

- Download and install OpenEMPI using this guide.
- Start OpenEMPI by navigating to the ***/usr/share/openempi/bin*** directory and execute the ***sh startup.sh*** command, you might want to be patient with it because it takes a while to be ready for use.
- To access the application, go to http://server_ip:8080/openempi-admin in your browser where *server_ip* is the host name or IP address of the machine where you've installed OpenEMPI, the default username and password are both **admin**

- If you've installed OpenEMPI on the same machine as the SHR or OpenHIM, you might run into some port conflicts, therefore you might need to reconfigure ports for those affected, e.g it runs inside a different tomcat instance it ship with, meaning you would need to reconfigure its tomcat instance to listen on a different port if the SHR is installed on the same machine and listening on port 8080 too.
- You might want to check that the PIX/PDQ server has been started by running the ***netstat -plant | grep LISTEN*** command you should see a process listening on port 3600, if not then the PIX/PDQ server isn't started, see the docs here for how to start it through the OpenMEPI user interface. Alternatively, you could configure OpenEMPI to automatically start the PIX/PDQ server by setting the value of the ***admin-configuration/autostart-pixpdq*** tag to true in the ***/usr/share/openempi/openempi-entity-3.3.0c /conf/mpi-config.xml*** file.

Install The XDS mediator

The mediator can be installed on a difference machine from that where the OpenHIM is installed but it is recommended to install it on the same machine to minimize what needs to be configured for the two to communicate.

- Download the latest build mediator of the mediator from here.
- Switch to the terminal, navigate to the folder where you downloaded the tar file to and run the commands below.

```
mkdir -p /usr/share/openshr/mediators/xds-mediator
tar -xzvf openhim-mediator-xds-1.0.3.tar.gz -C /usr/share/openshr/mediators/xds-mediator --strip-components=1
cd /usr/share/openshr/mediators/xds-mediator
```

At this point the mediator files have been extracted to the ***/usr/share/openshr/mediators/xds-mediator*** directory and your terminal is rooted at this folder.

- Rename the ***mediator.properties*** file to ***xds-mediator.properties***
- There are other properties that you will need to update later in the ***xds-mediator.properties*** but for now update just set the ones below:

  mediator.host : The host name or IP address of the machine on which the mediator is hosted

  core.host : The host name or IP address of the machine on which OpenHIM is installed

  core.api.port : The api port on which OpenHIM is listening, should match the port configured in OpenHIM's ***/etc/openhim/config.json*** file for the api -> httpsPort field

  core.api.user : The OpenHIM username

  core.api.password : The OpenHIM password

  pix.manager.host : The host name or IP address of the machine on which OpenEMPI is installed

  pix.secure : Specifies if the connection to OpenHIM is secure i.e over HTTPs

  pix.manager.port : The port on which the PIX/PDQ server in OpenEMPI is listening, PIX/PDQ server by default listens on port ***3600***

  pix.manager.securePort : The secure port on which the PIX/PDQ server in OpenEMPI is listening if the connection to the PIX/PDQ server

  xds.repository.host : The host name or IP address of the machine on which the SHR is installed

  xds.repository.secure : Specifies if the connection to the xds repository (SHR) is secure i.e over HTTPs

xds.repository.port : The port on which the SHR is listening, since the the SHR is OpenMRS which is hosted in tomcat, this should be port tomcat is listening on

xds.repository.securePort : The secure port on which the SHR is listening, since the the SHR is OpenMRS which in tomcat, this should be secure port tomcat is listening on

xds.repository.path : Set this to ***/openmrs/ms/xdsrepository***

pnr.patients.autoRegister : true

pnr.providers.enrich : false

pnr.facilities.enrich : false

- You will need to add the OpenHIM SSL certificate to your java truststore by running the commands below, the script assumes you set the JAVA_HOME environmental variable otherwise you will need to replace it with the path to where the JRE is installed on your machine.

```
echo "Q" | openssl s_client -connect openhim_server_IP:openhim_api_port > openhim.crt
keytool -import -trustcacerts -alias openhim -file openhim.crt -keystore $JAVA_HOME/jre/lib/security/cacerts
```

- By default, logging is enabled in the mediator via tcp which implements the IHE ATNA profile because the value of the ***atna.useTcp*** property is already set to true in the ***xds-mediator.properties*** file. If for some reason it isn't turned on or incorrectly set up, you can re-enable it by doing the following: Set the value of the ***atna.useTcp*** property to true in the ***xds-mediator.properties*** file, open the ***/etc/openhim/config.json*** file on the machine where OpenHIM is installed and set the value of the ***auditing*** -> ***servers*** -> ***tcp*** -> ***enabled*** field to true, you will also have to configure the properties below in the ***xds-mediator.properties*** file.

  atna.host : The host name or IP address of the machine on which OpenHIM is installed
  atna.useTcp : Set this to true
  atna.tcpPort : The tcp port on which the ATNA logging service in OpenHIM is listening, should match the port configured in OpenHIM's **/etc/openhim/config.json** file for the ***auditing*** -> ***servers*** -> ***tcp*** -> ***port*** field, the default value is ***5052***
  atna.secure : Set this to false for a non-production instance otherwise true to communicate over a secure connection

- Start the mediator by navigating to the folder where you extracted the mediator files to i.e ***/usr/share/openshr/mediators/xds-mediator*** and run the command below:

```
java -jar mediator-xds-1.0.3-jar-with-dependencies.jar --conf xds-mediator.properties
```


When the mediator starts, it auto registers itself with OpenHIM and if this was successful you should be able to see it listed in OpenHIM, i.e. go to the OpenHIM console application and select ***Mediators*** from the left panel.

Configure the mediator in OpenHIM

See the documentation for more details on OpenHIM, channels and mediators.

Go to the OpenHIM console's home page i.e http://openhim_server_ip

Setup the XDS.b Mediator channel

- Select ***Mediators*** from the left panel, and then select ***OpenHIE XDS.b Mediator***
- On the far right, you should see a little green box with a white cross which you should click
- Select the ***Channels*** tab from the left panel, you should see a new channel named ***XDS.b Mediator*** that has just been added.
- Assuming you are still on the channels tab, click the edit icon (yellow pencil) on the right, select the ***Routes*** tab, you should see the ***XDS.b Mediator*** route, click the edit icon on the right, change the value of the ***host*** field from localhost to the IP address of the server where OpenHIM is installed. **Note** that localhost might not work even if you have OpenHIM and the mediator running on the same machine if they are running inside a virtual box.

The channel you've created above is private by default, you can make it public by selecting it from the list, i.e go to the **Request Matching** tab, mark it as public and save the changes. Typically in production you would need to keep it private, it requires an ***xds*** role as you can see under the section labeled ***Which clients should be able to access this channel?*** Therefore, you'll need to add a Client and assign it this role, see the adding clients guide.

If OpenHIM and the mediator are installed on separate machines or if they are installed in the same but inside a virtual box, you might need to change route settings by doing the following, select the ***Channels*** tab from the left panel, you should see a new channel named ***XDS.b Mediator*** that was created above, click the edit icon (yellow pencil) on the right, select the ***Routes*** tab, you should see the ***XDS.b Mediator*** route, click the edit icon on the right, change the ***host*** to value from localhost to the IP address of the machine instead.

Upgrade OpenMRS and the SHR modules

This should not be done for a production installation, you would need to reinstall a new version of OpenMRS form scratch with the require modules.

- In OpenMRS webapp, click on ***Administration*** in the green bar at the top of the page, under **Maintenance** select **Settings**, select the **Search** tab on the left panel and clear the value of the **Index Version** global property because it's necessary for the search index to get rebuilt.
- Delete the existing *openmrs.war* file and *openmrs* directory from the **/usr/share/openshr/tomcat/webapps** directory.
- Delete all the SHR modules i.e EXCEPT for the webservices.rest module from the **/usr/share/openshr/openmrs/modules** directory.
- Delete the *work* directory from the **/usr/share/openshr/tomcat** directory.
- Download and install the *openmrs.war* file version 1.11.5 or later i.e any version in the 1.11.x or 1.12.x release lines, don't try 2.0 or later versions because not all the SHR modules are compatible.
- Copy the downloaded *openmrs.war* file to **/usr/share/openshr/tomcat/webapps** directory.
- Copy the updated SHR modules below to the **/usr/share/openshr/openmrs/modules** directory.
    1. shr-atna-1.0.1-SNAPSHOT.omod
    2. shr-contenthandler-3.0.1-SNAPSHOT.omod
    3. shr-cdahandler-1.0.1-SNAPSHOT.omod
    4. xds-b-repository-1.1.1-SNAPSHOT.omod
    5. shr-odd-1.0.1-SNAPSHOT.omod
- Restart your machine.

It's important to restart your machine and not just tomcat after upgrading, in fact as a general rule you should always restart the machine instead of the individual components of the SHR if you installed it using the above ubuntu package because it creates some startup scripts that run the executables with the appropriate user account and required settings, otherwise you might run into some strange issues

Key configurations to make before you start using the setup

To successfully submit a case report for a given patient from the OpenMRS point of care system, their preferred identifier must be of an identifier type that is mapped to one in the health information exchange. The universal identifier of the mapped identifier domain in the client registry must be a valid OID, the SHR is the one that requires this. For testing purposes you can set it to a string of numbers separated by dots e.g *1.3.6.1.4.1.21367.2010.1.2.301* but in production you would need to obtain a valid and preferably registered OID, for more details on OIDs and obtaining one for your organization go to HL7 OIDs page. An identifier domain in OpenEMPI is analogous to a patient identifier type in OpenMRS, also the universal identifier field of an identifier domain in OpenEMPI is equivalent to the name of a patient identifier type in OpenMRS.

1. Create new identifier domains in OpenEMPI see the Manage Identifier Domains page for details, alternatively you could work with an existing one, the *Universal IdentifierId Type* values for the identifier domains must be set to **ISO** and the values of the *Universal IdentifierId* field must be valid OIDs, the SHR requires this, as you can see from the other existing identifier domains, the *name* and *namespace identifier* field can be set to be the same as the universal identifier value
2. Agree on an enterprise/global patient identifier domain to use and set it up in OpenEMPI as below
    a. On the machine where OpenEMPI is installed, open the ***/usr/share/openempi/openempi-entity-3.3.0c/conf/mpi-config.xml*** file, you should see a ***global-identifier*** tag, change the values if its nested tags to match those of the global ID you came up with.
3. Mediator configurations: Open the ***xds-mediator.properties*** file that you created earlier above and edit the values of the properties below.
    a. ***client.requestedAssigningAuthorityId*** **:** Specifies the patient identifier domain to convert to when enriching a document with enterprise IDs, when a document is submitted to the HIE, it normally has local patient identifiers from the source, this tells the mediator in OpenHIM which identifier domain to convert to before forwarding the document to the SHR.
    b. ***client.requestedAssigningAuthority*** (Optional) **:** Specifies the name of the patient identifier domain to convert when enriching a document with enterprise IDs.
    c. ***pnr.patient.autoRegister :*** Set it to true if you want patients to be auto registered if they don't exist in the Client Registry i.e if the mediator can't find them in the client registry by their local ID from the source.
    d. You **MUST** restart the mediator for the changes to take effect.
4. The patients in the point of care OpenMRS instance and the SHR need to have identifiers matching the format specified by the values of their respective ***shr-cdahandler.id.format*** global properties, by default they are set to ***%2$s^^^&%1$s&ISO*** and there should hardly ever be a reason to changed them, only advanced users who know what they are doing should alter them.
5. Point of care OpenMRS instance configurations:
    a. Go to ***Home*** -> ***Case Reports*** -> ***Configure***, set the values of the listed configurations, they have fairly good descriptions, please refer to the CBR module's documentation on how to set them.
    b. The user that submits case reports is required to have a provider account that is linked to their person record.
    c. Go to ***Home*** -> ***Case Reports*** -> ***Identifier Mappings*** and map each local identifier type to one of the identifier domain you created in step 1 above, the values must be the *Universal IdentifierIds* of the identifier domains in OpenEMPI you are mapping to.
6. SHR configurations:
    a. Click on ***Administration*** in the green bar at the top of the page, under ***Maintenance*** select **Settings**, select **Shr - CDAhandler** from the left panel and set the values of the global properties that start with ***Autocreate*** to true or false based on your preferences.
7. OpenXDS configurations:

a. In the ***/usr/share/openshr/openxds/openxds.properties*** file, there is a ***validate.patient.id*** property, when a document is submitted and the patient doesn't exist in OpenXDS' PIX/PDQ registry which will be the case for new patients, the document processing will fail unless this property is set to false.

Integration with DHIS2 Tracker

The purpose of this integration is to be able to push case report data to a DHIS2 instance from the shared health record (SHR), whenever a case report is received in the SHR we want to be able to forward this data to DHIS2 tracker, i.e. the patient gets registered and enrolled in a specific program and then submit events to the same program whenever subsequent case reports are received for the same patient, how to set up programs in DHIS2 is beyond the scope of this documentation.

The DHIS2 tracker module needs to be installed in the SHR to provide this functionality and below are details on how to set it up.

- Note that the module has a dependency on the event module, you will need to download the latest build of the module, here is a copy you can download and install.
- Download and install the DHIS2 tracker module.
- It's important to restart the SHR (OpenMRS) instance after installing the above modules for proper functioning.
- Click on ***Administration*** in the green bar at the top of the page, under ***Maintenance*** select **Settings**, select **Dhis 2 Tracker** from the left panel and set the values of the properties according to the descriptions. To find these values in dhis2, go to apps (looks like a phone pad) and maintenance. Under the program section you will find the areas of interest to the DHIS2 tracker module (Program, Tracked entity type and Tracked entity attribute). Once in these sections click actions (3 vertical dots to the right) and select Show Details. You will find the ID that is needed in tracker module for each attribute.

The assumption is that you've already setup a program in your DHIS2 instance with a tracked entity type that represents a Person along with the necessary entity attributes, for each of the global properties that have a ***UID*** suffix, the value is the UID of the corresponding entity attribute in DHIS2. The values for the Female Option Code and Male Option Code are the codes and not the UIDs of the corresponding options in DHIS2.

Please pay extra attention to the description of the ***Case Report Encounter Type Name*** global property

You will need to create or update an organization unit in DHIS2 and map it to your OpenMRS instance that will be submitting case reports, if you plan on submitting case reports from multiple instances then you will have to do it for each of the instances, please refer to the DHIS2 documentation on how to create/update organization units. An organization unit has a code field, to map your submitting OpenMRS instance to an organization unit, you need to do the following in the SHR

- Create a new location , Click on ***Administration*** in the green bar at the top of the page, click ***Manage Locations*** and click ***Add Location***
- Set the value of its **DHIS2 code** location attribute type to match the value of the code field of the mapped location in the DHIS2 instance.
- Set the value of the ExternalId location attribute type to ***ORGANISATION_EXTENSION***^^^&***ORGANISATION_OID***&*ISO* where ORGANISATION_EXTENSION and ORGANISATION_OID match the values of the organisation OID and extension configurations in the submitting OpenMRS instance's respectively, typically they are values of the global properties named *casereport.organisationOID* and *casere port.organisationExtension*.

Things to keep in mind:

- For this to work, when submitting a case report for a patient, one of the triggers **MUST** be *New HIV Case* for the SHR to register and enroll the patient in the configured program in DHIS2.
- The DHIS2 instance MUST be in the same time zone as the the OpenMRS instances submitting case reports, otherwise depending on the time of the day you submit a case report you might end up with a failure with an error message in the OpenMRS instance's logs saying the event or incident date is in the future, this is because the rest API exposed by DHIS2 tracker doesn't support the time component and time zones for dates.

To view the details in DHIS2, open the Tracker Capture app and on the right panel you should select the org unit mapped to the OpenMRS instance from which you submitted the case report and you should see the patient's record.