Implementation Plans for Agreed Requirements of HL7 Query Module

This page details the implementation plans for the Improvements to the HL7Query module project of GSoC 2013.

Step 1 : Capturing Events

Building a generic listener that solves the specific first use case of listening for ORUR01 and ADT related messages (events).

At the moment, we will only build support for ORUR01 events. However, during the design phase we should always consider the need to be able to expand on our design to support other events such as ADT messages.

The OpenMRS AtomFeed module uses the OpenMRS Event Module to identify Patient, Obs, concept and encounter related events. This is done via a generic GeneralEventListener class (one listener to identify all). One option open to us is to write a similar eventlistner which we could use to identify all Encounter related events. For this we will be (initially) building a special listener that is inspired by the design of the Atom Feed Module. As in the Atom Feed Module there will be a generic listener to listen to the Patient and Encounter related events. For our purposes, we are only interested in CREATE and UPDATE events, and not DELETE events.

We will build an enum of all the specific events we will identify during the current phase of development. This enum will currently contain only Encounter.CREATE and Encounter.UPDATE. When we are building support for Patient events, support for Patient.CREATE and Patient.UPDATE will also be added.

The contents off this enum will be displayed on the HL7 event tracking configuration page (Image A), where users will be allowed to pick which ones to follow.

Point of discussion : Will we identify all patient and encounter events, and filter them based on the selections the user makes, or can we come up with a process where the selections the user make automatically filter data in the eventlistner class? If we cant filter applicable events inside the eventlistner class, and must listen for all, then (in the interests of reducing overhead) we should restrict the module to listen to only encounter related events at this point.

Step 2: Creation of a Sender Profile

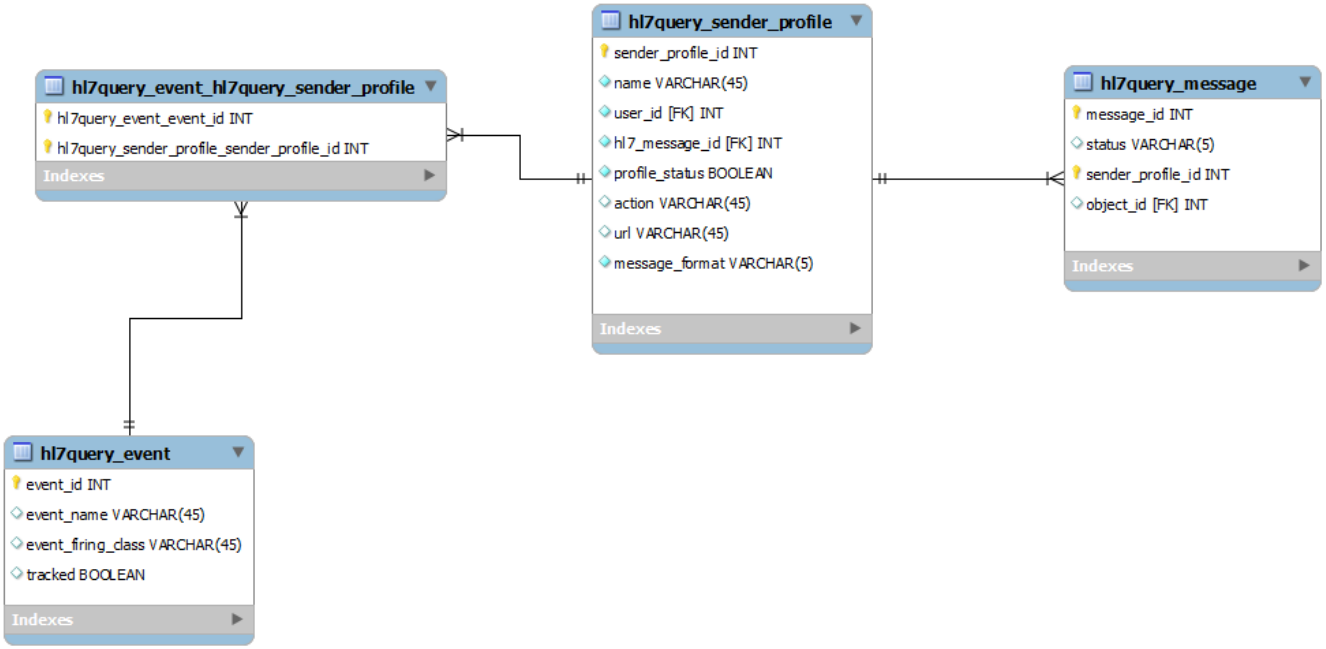
In terms of Image B (The creation of sender profiles used to manage message processing data) The Sender Profile creation page consists of a drop down where users can pick which particular event that the profile will track. The issue I foresee here is that we have not provided an appropriate measure for users to select the data to populate this drop down. If developer was to use all the data in the hl7query_template table, he would end up with a lot of unnecessary sub-templates that are used to build a parent template.

So, I propose that we introduce a new column to the hl7query_template table which allows us to identify only the parent templates. Currently, there will be only one value (Generic ORUR01) in this dropdown. However, others will be added when ADT messages are supported.

Now, the drop down list in the sender profile template creation page will list ONLY the templates that are tagged as 'parent'.

Persistence or Sending of messages

Some sender profiles will persist the message for sending later, while others will POST or PUT (depending on whether the action is save or update) the message immediately. If the message is to be persisted, we need to persist only the id of the object, and not the entire hl7 message. We took the decision not to persist the entire message as it can often be very large.



In some cases, authentication may be necessary to send the message, in such an event, the credentials should be persisted as a global property, and then used. I also propose that the message sending is incorporated into a queue, and managed therein. It may also be useful to display details of all messages that were sent on a separate list, so that users can track which were successful, and which failed.