Merging and Restoring patients

Table of Contents

Overview

The purpose of this document is to discuss the requirements and design specifications for merging and restoring patients stored in the SHR.

1) The Patient Merge process refers to the act of retiring Patient B in favour of patient A. In such an event, the encounters, observations and other underlying objects which belonged to patient B will be transferred to Patient A, and Patient B will be retired. In a multiple patient merge process, Patients A, B and C may be retired in favour of patient X. Underlying objects while belonged to patients A, B and C will also be transferred to patient X as specified earlier.

2) The Patient restoration process refers to the act of restoring an already retired patient. Assume that Patient B had been retired in favour of Patient A. In such an event, Patient B can be restored, and encounters, observations and other underlying objects, which were assigned to Patient A restored back to Patient B. In a multiple patient restore process, Patients A, B and C may be restored from patient X. Underlying objects while belonged to patients A, B and C will also be transferred back from patient X.

   It is important to note that while multiple patient merge / restore actions are possible, a single OpenEMPI request will address only a one-to-one patient action. Therefore, multiple patient merge / restore tasks will be represented by multiple OpenEMPI requests.

The OpenMRS core already provides patient merge features. However, we cannot use it for our use cases due to the following reasons,

1) The OpenMRS Patient merge is uni-directional only. It does not support the restoration of merged data.

2) The OpenMRS merge feature automatically re-assigns all person names and identifiers of the patients being retired to the surviving patient.

3) Since our use case requires patient restoration features, we will need to log additional information on the patient merge process

Due to these reasons, an alternative patient merge solution had to be designed.

However, it was decided that the RHEA patient merge process should duplicate the core merge features in all other aspects. For example, we will follow the existing methodology of transferring ownership of related objects between different patients.

SHR event processing logic

OpenEMPI has been designed to only notify clients of a change in a single patient's identifier. It will also include a basic attribute to state the event which caused the patient identifier to be changed. As such, the SHR will need to use this information to determine what events occurred in the MPI. The event types can be a merge event, a restore event or an identifier update event.

The following explains the logic that the SHR should execute to determine what event it needs to perform.

- The SHR receives a patient identifier change notification for a single patient. This contains pre-update identifiers and the post-update identifiers after the event occurred.
- The SHR reads the event type attribute to see what event caused this patient identifier change

A) If the event type is a merge or link,

   - The SHR looks up the patient with the pre-update ECID (if not found, an error is thrown)
   - The SHR looks for a patient with the post-update ECID
     - If not found, update the pre-update ECID to the new value as this is just an identifier update. Otherwise, merge the patient with the pre-update ECID into the patient with the post-update ECID. This involves retiring all identifier, demographic and encounter information for the pre-update patient and re-creating all encounter data and reassigning it to the post-update patient.

B) If the event type is restore or un-link

   - The SHR looks up the patient with the pre-update ECID (if not found, an error is thrown)

- ▪ The SHR looks for a patient with the post-update ECID in its log of retired patients
    - If not found throw an error. Otherwise, restore the patient with the post-update ECID and move the encounters that belonged to that patient from the patient with the pre-update ECID

C) If the event type is just an update of the patient identity (ie. Not a merge or restore action)

- The SHR looks up the patient with the pre-update ECID and changes that patient identifier to the post-update identifier.

Overview of the RHEA Patient Merge workflow

Explained below is the high level workflow for the patient merge process.

- The SHR receives a patient identifier change notification from OpenEMPI via the interopability layer.
- The SHR determines that this event is a patient merge event using the logic described above.
- Once the patient merge request is received, the SHR will carry out the merge process. Assuming that patient B is to be retired in favour of patient A, this process will comprise of the following steps.
    1) Identify all the underlying objects belonging to patient B. Transfer ownership of the underlying objects from patient B to Patient A
    2) Audit the encounter, obs and other data which were updated in this manner
    3) Retire Patient B
    4) Audit the patient retire process

*When a patient is retired, we will also retire its identifiers / names and demographic data. They will not be copied over to the remaining patient. This is a key deviation, which changes our merge process from the OpenMRS core merge process.

Overview of the RHEA Patient restore workflow

Assuming that patient B has earlier been retired, and is now to be restored, the patient restoration process would include the following steps,

1. Restore the retired patient B
2. Restore Patient B's identifiers and demographic data
3. Identify the encounters, obs and other objects which were copied over to Patient A at the time Patient B was retired
4. Reassign these objects from Patient A to Patient B
5. Audit the restoration process

Implementing the merge process

The HIE layer will intercept the messages sent from OpenEMPI. It will split the message into individual notifications, and send each of these to the SHR via a single request.

A sample is show below:

```
<identifierUpdateEvent>
        <dateCreated>2013-04-25T00:00:00-04:00</dateCreated>
        <identifierUpdateEventId>255</identifierUpdateEventId>
        <postUpdateIdentifiers>
            <postUpdateIdentifier>
                <identifier>9876543210987654</identifier>
                <identifierDomain>
                    <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                    <identifierDomainDescription>ECID</identifierDomainDescription>
                    <identifierDomainId>10</identifierDomainId>
                    <identifierDomainName>ECID</identifierDomainName>
                    <namespaceIdentifier>ECID</namespaceIdentifier>
                    <universalIdentifier>ECID</universalIdentifier>
                    <universalIdentifierTypeCode>ECID</universalIdentifierTypeCode>
                </identifierDomain>
            </postUpdateIdentifier>
        <postUpdateIdentifier>
            <identifier>CY-123AB</identifier>
            <identifierDomain>
                <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                <identifierDomainDescription>OMRS555</identifierDomainDescription>
                <identifierDomainId>10</identifierDomainId>
                <identifierDomainName>OMRS555</identifierDomainName>
                <namespaceIdentifier>OMRS555</namespaceIdentifier>
                <universalIdentifier>OMRS555</universalIdentifier>
                <universalIdentifierTypeCode>OMRS555</universalIdentifierTypeCode>
            </identifierDomain>
        </postUpdateIdentifier>
    </postUpdateIdentifiers>
    <preUpdateIdentifiers>
        <preUpdateIdentifier>
            <identifier>1234567890123456</identifier>
            <identifierDomain>
                <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                <identifierDomainDescription>ECID</identifierDomainDescription>
                <identifierDomainId>10</identifierDomainId>
                <identifierDomainName>ECID</identifierDomainName>
                <namespaceIdentifier>ECID</namespaceIdentifier>
                <universalIdentifier>ECID</universalIdentifier>
                <universalIdentifierTypeCode>ECID</universalIdentifierTypeCode>
            </identifierDomain>
        </preUpdateIdentifier>
        <preUpdateIdentifier>
            <identifier>CY-123AB</identifier>
            <identifierDomain>
                <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                <identifierDomainDescription>OMRS555</identifierDomainDescription>
                <identifierDomainId>10</identifierDomainId>
                <identifierDomainName>OMRS555</identifierDomainName>
                <namespaceIdentifier>OMRS555</namespaceIdentifier>
                <universalIdentifier>OMRS555</universalIdentifier>
                <universalIdentifierTypeCode>OMRS555</universalIdentifierTypeCode>
            </identifierDomain>
        </preUpdateIdentifier>
    </preUpdateIdentifiers>
    <source>LINK</source>
    <transition>JOIN</transition>
</identifierUpdateEvent>
```

The SHR will identify appropriate data from this message, and carry out the patient merge process.

Managing updates which occur after a patient merge action

Assume patient A and patient B are merged at time t into patient A. Then encounters are stored in the SHR against patient A at time t+y. Subsequently at time t+y+1, patient B is unlinked and the SHR restores patient A. When restoring patient A, to which patient do we assign the encounters created after the original merge.

I propose we store in a separate table in OpenMRS, for each encounter submission, the encounter id in OpenMRS and the local point-of-care patient id. This will allow us to identify which patient the encounters need to be re-assigned to in the scenario above. Note that this is needed since we will lose the local patient identifier information for the retired patient when we perform the original merge.

Auditing the merge process

When a request to merge a patient is received, the SHR will,

- Retrieve the specified patients as well as the encounters, obs and other objects belonging to them.
- Persist the patient id and underlying object ids into the database along with a timestamp and process owner id.

Implementing the patient restore process

The HIE layer will intercept the messages sent from OpenEMPI. It will split the message into individual notifications, and send each of these to the SHR via a single request.

```
<identifierUpdateEvent>
        <dateCreated>2013-04-25T00:00:00-04:00</dateCreated>
        <identifierUpdateEventId>255</identifierUpdateEventId>
        <postUpdateIdentifiers>
            <postUpdateIdentifier>
                 <identifier>1234567890123456</identifier>
                <identifierDomain>
                    <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                    <identifierDomainDescription>ECID</identifierDomainDescription>
                    <identifierDomainId>10</identifierDomainId>
                    <identifierDomainName>ECID</identifierDomainName>
                    <namespaceIdentifier>ECID</namespaceIdentifier>
                    <universalIdentifier>ECID</universalIdentifier>
                    <universalIdentifierTypeCode>ECID</universalIdentifierTypeCode>
                </identifierDomain>
            </preUpdateIdentifier>
            <preUpdateIdentifier>
                <identifier>CY-123AB</identifier>
                <identifierDomain>
                    <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                    <identifierDomainDescription>OMRS555</identifierDomainDescription>
                    <identifierDomainId>10</identifierDomainId>
                    <identifierDomainName>OMRS555</identifierDomainName>
                    <namespaceIdentifier>OMRS555</namespaceIdentifier>
                    <universalIdentifier>OMRS555</universalIdentifier>
                    <universalIdentifierTypeCode>OMRS555</universalIdentifierTypeCode>
                </identifierDomain>
            </postUpdateIdentifier>
        </postUpdateIdentifiers>
        <preUpdateIdentifiers>
            <preUpdateIdentifier>
                <identifier>9876543210987654</identifier>
                <identifierDomain>
                    <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                    <identifierDomainDescription>ECID</identifierDomainDescription>
                    <identifierDomainId>10</identifierDomainId>
                    <identifierDomainName>ECID</identifierDomainName>
                    <namespaceIdentifier>ECID</namespaceIdentifier>
                    <universalIdentifier>ECID</universalIdentifier>
                    <universalIdentifierTypeCode>ECID</universalIdentifierTypeCode>
                </identifierDomain>
            </postUpdateIdentifier>
            <postUpdateIdentifier>
                <identifier>CY-123AB</identifier>
                <identifierDomain>
                    <dateCreated>2009-06-17T11:45:54-04:00</dateCreated>
                    <identifierDomainDescription>OMRS555</identifierDomainDescription>
                    <identifierDomainId>10</identifierDomainId>
                    <identifierDomainName>OMRS555</identifierDomainName>
                    <namespaceIdentifier>OMRS555</namespaceIdentifier>
                    <universalIdentifier>OMRS555</universalIdentifier>
                    <universalIdentifierTypeCode>OMRS555</universalIdentifierTypeCode>
                </identifierDomain>
            </preUpdateIdentifier>
        </preUpdateIdentifiers>
        <source>UNLINK</source>
        <transition>JOIN</transition>
    </identifierUpdateEvent>
```

The above message represents a restore request for a single patient object.

When a patient is retired, the SHR Adapter module will log the retired patient, the surviving patient, as well as the encounter, obs and other ids which were transferred over. When a request to restore a patient is received, we will consult the patient merge log table to identify if the specified patient has actually been merged. If so, we will have access to the encounters, obs and other objects which were transferred over during the merge. Using these ids, we can identify which of the updated objects should be transferred back.

Error message codes

The following error codes will be returned as appropriate,

- Patient merge ok → 200
- One or both patients not found → 404
- If requested merge has already been completed → 200

When a request to restore a patient is received,

- Obtain the details persisted for that patient when it was first merged.
- Using the object ids contained herein, identify the actual encounters, obs and other objects copied over to the surviving patient.
- Retire these, and copy them over to the newly reactivated patient.

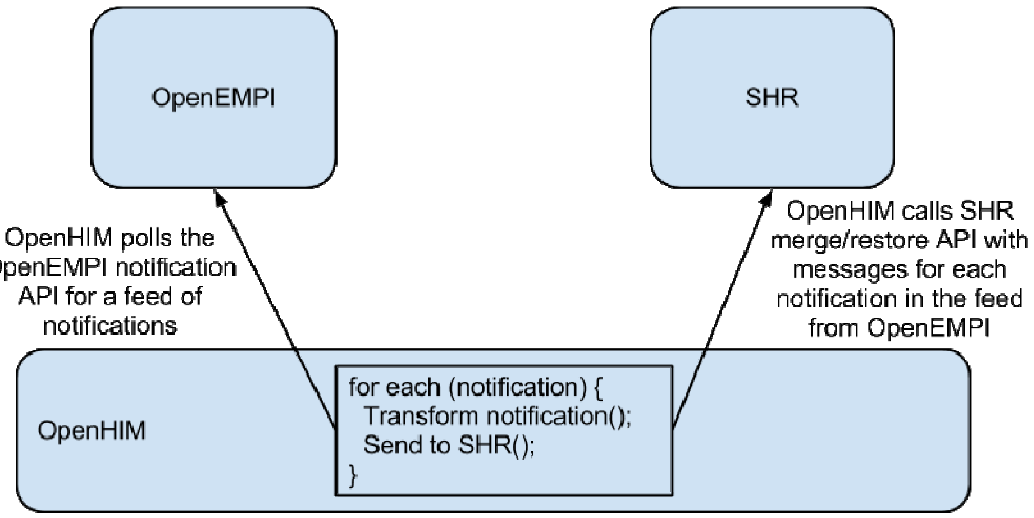
Auditing the patient restoration process

For the patient restoration process, the following information will be logged,

- The patient id of the patient to be restored
- The patient id of the surviving patient, from whom these data will be copied
- The object id's of each child object transferred over during the restoration
- The time of restoration
- The user who carried out the restoration

OpenHIM Design

The OpenHIM will act as an intermediary between OpenEMPI and the SHR to more loosely couple the workflow between these two applications. The overall architecture is shown in the diagram below.


OpenEMPI
SHR
OpenHIM polls the OpenEMPI notification API for a feed of notifications
OpenHIM calls SHR merge/restore API with messages for each notification in the feed from OpenEMPI
OpenHIM

```
for each (notification) {
  Transform notification();
  Send to SHR();
}
```

In the OpenHIM the following needs to be implemented:

- Develop a connector that polls the OpenEMPI notification API on a specific time frame
- Develop a normalization transformer that can transform the notification feed from OpenEMPI to individual events in the format expected by the SHR.
- Develop a de-normalization component that allows the OpenHIM to send the event to the SHR.
- Normal auditing and logging of the message in the OpenHIM

Patient Merge scenarios

1) Missing identifiers

   Assume that the Patient merge request contains the following identifiers,

   a. Pre update identifiers:

      ECID : 1234

      OMRS: 2345

   b. Post update identifiers

      ECID: 7890

      OMRS:3456

      EACD: 5678

Based on the above, the patient to retire and the patient to survive were identified using their ECID numbers.

However, it was also noted that the surviving patient does not have an OMRS Id.

Therefore, a new OMRS id will be created, and assigned to the surviving patient, so that its patient identifiers are identical with those provided in the post update identifiers list.

2) Normal patient merge action

Assume that the patient to be retired contains two encounters with three obs each, while the surviving patient contains X number of encounters and Y number of obs.

Once the merge is completed, the patient who was to be retired is retired, and its two encounters (along with their obs) passed over to the surviving patient. The transfer of encounter and obs is managed by updating their patient id's to point to the surviving patient.

Patient restore scenarios

1) Missing identifiers

Assume that the Patient restore request contains the following identifiers,

   a. Pre update identifiers:

ECID : 1234

OMRS: 2345

b. Post update identifiers

ECID: 7890

OMRS:3456

EACD: 5678

Based on the above, the patient to restore was identified using its pre update ECID number.

Once this patient was restored, it was noted that its patient identifiers did not match with those listed as post update identifiers on the OpenEMPI request.

Therefore, new identifiers will be created and assigned so that the patient identifiers of the restored patient matches with those listed in the OpenEMPI request.

Logging of patient merge/restore transaction

The patient merge/restore action will be logged in the single table, as opposed to two separate tables for merge and restore actions. This decision was made on the grounds that a restore is basically a reversal of a patient merge transaction. Therefore, each time a restore occurs, an existing record will be updated.

1) For each patient merge request, the following data will be persisted.

   - The patient who is retired
   - The patient who survived
   - The merged data (this is a serialized object representing the encounter and obs uuid's which have been moved over to the surviving patient)
   - Created date
   - Created user
   - set 'restored' flag to false

When a patient merge is being reversed (a restore occurs), the following changes will take place in the above transaction record,

- Assign the restored date
- Assign the restored user
- Set the ‘ restored’ flag to true

Tasks completed until the 18th June 2013

To date, the following features have been implemented and tested,

1) The Patient Merge workflow
2) The patient restore workflow
3) Underlying data model changes / additions
4) Error message notification features
5) Auditing of each request / response

Outstanding tasks

As specified above, all functional requirements in terms of the SHR have already been completed. However, these features have only been tested via mock OpenEMPI requests made via SoapUI. They need to be re-tested end to end with real OpenEMPI requests once the Interopability layer has been connected with OpenEMPI, and is in a position to start sending valid requests to the SHR endpoint.