2009 Implementers Group Meeting Program Synchronization

Organizer: Ben Wolfe

Problem: handling disconnected data entry

Current (and only standard) option: RemoteFormEntry
Works to enter data in form at remote site, make available in target site, with the mechanism being to send/transfer partial SQL dump
disadvantages: 1) no real database exists at remote form entry site, and 2) there can be an unintended side effect of remotely created patiends are removed temporarily as a result of central site update

New (upcoming) option: Synchronization Module
basic principle is to track actual changes made through the API, package these changes for transport, send to target instance, and re-execute changes on the target system

This will allows correct handling of a patient visiting multiple sites that have a "sync" relation

Synchronization module demonstration on OpenMRS 1.5 (or maybe the eventual 1.5.1)
Depends on UUID feature in core, introduced in R1.5
Targeting a release of the module in 6 weeks (End of October?) but input/help from developers needed to resolve outstanding tickets

Synchronization methods

1. Web (Network), using transport over the network
2. File, using transport using physical media

Synchronization section in Admin page:

- Overview
- Configuration
- History of Changes
- Help

Points made during the presentation/discussion:

for synchronized systems, it is not recommended to execute direct DB changes because sync tracks API activities. DB changes*will get lost

- primary keys will be different for the same object in child/parent, but UUID handles this. object lookup is through UUID, not through primary key, "UUID acts as the primary key in the synchronized world"
- Side effects of UUID introduction is that primary keys for identical objects in different databases will be different. FormImportExport uses primary keys (and would be affected by sync). But: sync module can also be used to handle forms synchronization
- A new child server is created from the parent node.
- If syncing after independent updates have introduced the same object will require manual cleanup to remove duplicates. No tools exist to handle this automatically.
- A node can have both parent and child roles in a multilevel synchronization hierarchy
- in some settings it may make be helpful or more efficient to do synchronization between siblings. however, this is not supported with the current architecture
- Currently synchronization does not track the timestamp of a change in the data and when that change is synchronized.
- successive one way synchronization steps, without incorporating reply/acknowledgment is not tested (but should work)
- on the use of UUID versus URN/URI: the latter requires name space management
- on the use of database replication for synchronization: MySQL database replication cannot easily solve the same problems that the sync module addresses. it needs handling of primary key offsets to accommodate multiple sync updates to avoid duplication of primary keys.
- The module cannot yet handle synchronization of subsets of patients. This requires an architectural revision based on of aspects such as location based privileges, and reconsider child/parent as publisher/subscriber
- Sync enabling existing separate databases (synced merge?) requires manual editing to remove any duplicate information
- on efficiency: with many child sites, each childe can get a lot of synchronized data that is not of interest to them
- on database size: gets big with tracking the change table. Change table probably needs to be purged at interval. The UUID column adds 15% size (38 chars for each entry), File size to transport can become large.
- on the business case for synchronization: origins for the module are PIH needs in Rwanda, with shared clinical staff, data management staff, and patients that visit multiple sites. Not intended for country level synchronization