# Meeting/Call Notes

**Meeting purpose:** Community Call for OpenHIE Interoperability Layer

**Date:** 09-07-2013

**Attendees:**

* Rhonwyn Cornell (Jembi)
* Hannes Venter (Jembi)
* Kari Schoonbee (Jembi)
* John (Regenstrief)
* Larry Lemmon (Regenstrief)
* Shahid Khokhar (Regenstrief)
* Derek Ritz
* Daniel Futerman (Jembi)
* Mark Tucker (Regenstrief)

**Agenda**

1. Feedback from WorldVista Call - Hannes
2. Review recommendation for Interoperability Layer
3. Discuss standards “stack”
4. Any other business

**Call Recording file** *# 39774801*

<http://www.conferenceplayback.com/stream/96774894/39774801.mp3>

**Meeting Notes:**

* WorldVista Call
	+ HV: Not too interested in it as it uses MUMPS but wanted to investigate and learn more about the system and gather lessons learnt. Met last week and were taken through some of the interoperability profiles they’re using on WorldVista. Data model set up is interesting. They use HL7, set up using node JS system – Java. Use node JS to deal with high transactions. It’s like a noSQL database, data model is dynamic no validation of data going in. Bottom line: we will be looking at again to feed into IL and SHR going forward. DR: Is this similar to Mohawk approach or is it a post processing system not real time. HV: not sure how Justin et al store data. Acts like a document store, an object database. Larry: how do you do global merge of data, it would seem like it would be more difficult to do. HV: it’s for performance they do it this way, because don’t have relationships to resolve it’s quicker. noSQL uses ----- map reduce algorithm on noSQL side, Vista it’s all MUMPS. Larry: is Medsphere involved. HV: no. OpenVista not the same as commercial one and doesn’t have lots of the features the commercial one does.
* Interoperability Recommendations Review
	+ HV: Ryan is unfortunately away. Had a look at several tools; Aurien and Axial 360 no go. # option are ; OpenHIM, Mirth Connect, Connect 4.0. OpenHIM main tool in Rwanda, developed specifically for RHEA and gives Jembi a bit of a bias towards it was we have the knowledge and skills for it. Mirth has some drawbacks for our use cases. Connect 4.0 designed for USA
		- OpenHIM – developed for RHEA by Ryan and has been running in pilot for a few months. Very immature, there isn’t really a community around it. Based on community requirements there are a number of deficits in the software that will need to be addressed. We can however develop according to our requirements don’t have to adapt an existing system to meet our requirements. Deficiencies:
			* Coded for RHEA usecase, assumed have SHR and CR. A lot of interfaces developed for specific interfaces – FR = resource map. Needs work to make it more generic. DR: they are on ramps and off ramps right. We can choose to favour standards based profile/ HV: yes that’s right. Does follow this flow. It’s not hard coded but can’t just plug in new interface without new code. CR is a bit more generic – PIX profiles. Mule is designed for this sort of thing – multiple workflows etc. DR: Custom interfaces are often used for performance reasons. We should easily be able to switch; Standards vs. custom interfaces. HV: should there always be standards use is a good questions for discussion. DR: key is to architect for it. HV: always code base on policies. Mark: whats the difference between on ramp off ramp with dependencies injection? Do we have to choose one or the other? Which one you use depends on message content. HV: dependencies injection happens at the beginning. DR: don’t want it on a message basis but in the architecture. Mark: do we what an ESB at all? Comes form TS call – how does normal operation of the system interact with TS. two ways of thinking about code mapping. A) SHR do rudimentary code mappings. At run time you’re never talking to TS B) SHR doesn’t do term message, on every message I make a call out to TS. Path A) not caring about TS performance. Big difference between person and computer querying TS. My bias, SHR should do rudimentary code mapping, there’s a huge benefit at message time. 2 use cases, 1) patients seen at edge nodes and patient info dribbling in as HL7 messages, goal is HL7 go into SHR. Edge nodes use local codes for mapping, assume edge nodes already have term mappings; Edge nodes preconfigured with doctors details/number. Don’t want/need orchestration within the HIM. HV: argue path A) doesn’t preclude the use of an ESB. Like the idea SHR does a bit of local mapping. Don’t think this precludes use of ESB. A central point solves mediation problem. Mark: has same advantage. Everyone sends their messages to same point, a central node, but not necessarily an ESB DR: so central node is SHR? DR: compelling argument this is. British Columbia followed same approach, was brought in after BC derailed: assumption that someone at the edge of the network is going to come up with the perfect message, you start the clock at when person starts constructing perfect message not when message is sent. Traffic from outside is always faster than doing inside. At steady-state you can assume success and only throw exception on error. System becomes faster at steady-state. Codes become cached. Role of TS; As soon as you want to do clinical decision support you have to be able to see decision tree. Cannot do CDS without TS and decision trees. Always more use cases. Mark: Agree CDS requires deep knowledge of terminology. DR: Never realized what a compelling argument it was. Understands why BC was compelled. However no other province in Canada has done this. BC has rolled back. Mark: feel like this is what we’re doing in Indiana and we’re not being compelled to roll back. If we have registries people will use them. The perfect message is what you do, the only thing you don’t know at the edge node is the patient. You already know your doctors, your terms. The message is already as perfect as it’s going to be. Expose CR with thin layer to clients. Thin layer is authentication. Central node great for authentication, great wrapper for our registries. But only CR gets used heavily. Advocating design that says main line of messages come in and get sent to the SHR with no fuss. We use Mirth as simple message queue. If we own edge nodes, we can build perfect messages. Dirty on the outside, clean of the inside. How much must my preprocessor do? Withering away of preprocessor ultimate goal. Larry: general design question: are we going to have a reflection of all the registries at the edge nodes? (a copy of them, PR, TS, FR). DR: will be a real challenge. Have to worry about cache synchronizing. Larry: can do once per day. Not patients, thinking about PR, TS and FR so forth. Derek: security reasons for not sending out ECIDs etc. out to clients. But caching has benefits. Secrecy doesn’t always work out. Cached opportunistically. Mark: doesn’t see need for TS at edge node. They know what code they want to use. DR: Can engineer for perfect messages. Cache golden keys at edges. Perfect messages are very fast.
* Highlight Marks argument – design for a system that over time will produce perfect messages

Action items

* Mark to post documentation on perfect messages

Future discussion topics stemming from call

- engineering for perfect messages

- (ongoing) role the interoperability layer. Do we need an ESB?

***Next Community Call***

The next call will be on Tuesday 16th July 2013.