# **OpenHIE Interoperability Layer Call Notes**

**Meeting purpose: Community Call for OpenHIE IL**

**Date: 23-07-2013**

**Attendees:**

* Ryan Crichton
* Linda Taylor
* Shahid Khokhar (Regenstrief)
* Larry Lemmon (Regenstrief)
* John Lajiness (Regenstrief Student)
* Mark Tucker (Regenstrief)

**Agenda**

1. Discuss of Interoperability Layer architectural concerns
	1. ESB paradigm concerns
	2. Central component vs. multiple tools/systems

 2. Any other business

 stack of standards ?

**Call Recording file # 87528201**

**Meeting Notes:**

***Discuss of Interoperability Layer architectural concerns***

***ESB paradigm concerns***

RC re-capped process to date

OpenHIM looked like the best option but have also identified some architectural concerns for discussion

One of key points is use of ESB for IL

ESB does transformation, complex orchestration

MT-assume registries exist with functions with a RESTful interface and a Java API

The CR group has provided a clean API for me to use

can also use a simple RESTful call

in OpenHIE world some of these may be pushed to edge node

in which case do I need orchestration? if syntactic trivial can use Mirth, or complex ones i.e. code-mapping , is easy to write if I have clean interfaces - not interested in using an ESB for this

take “dirty” messages and make them acceptable for our SHR, the tashk is hard so need to fix message in Java without access to registries. What’s left for an ESB to do?

Don’t view this as a central component - messages arrive in queue - a raw queue to pre-processed queue with pre-processor in the middle - to SHR

Has a very complex workflow

Problem not so much concept of ESB but that the ones in existence with graphical mapping interfaces

Agree that we don’t want everything in one monolithic system - existing ESBs often are but doesn’t have to be - can be made up of distributed components. Should specify what we do/don’t want: e.g.

* We want something easily distributed over different processes
* We don’t want to rely on graphical interfaces to code workflow

Need to define clean interfaces and clean interpretations of messages within our architecture

If edge nodes wanted to consult CR would it do it independently or must it go through the IL?

If there was no ESB how would we do it? Use a security interface that was well thought out

Must put a common security and logging style on it which registry must use when exposed

Then we would have to define these interfaces and process to be used. The code would be in registry itself.

By creating security layer for each layer must on-board each new registry / replicate security layer - using a central layer will do away with this need. Must balance scalability with performance

The edge nodes should not have to know where all the registries are. Issue of location transparency - don’t need to know where registries are, just need to know where central IL is

IL provides a routing mechanisms to the registries

2 things that we need:

* central component to deal with security and logging
* some processes to do orchestration for some sort of workflow to enrich message before it goes to SHR

Agree that this makes sense

2 pieces:

* stream of V2(LLP) messages to processers to SHR (caters to high volumes)
* regarding registries - single Apache can handle these

Have interface and systems component - passes to a mediator

if we separate these into two individual systems with different mediators that are basically processors would this work?

Would like to see V2 messages - very simple - and a single Apache. May be a role for an ESB if other functions required that not catered for

Would not mix the two : Interface vs persistence are radically different

Have read-only access to registries and message queue going mainly to SHR - two different groups

Only data to be stored in SHR would go through “message pipeline”

Maybe a query-response model for 1st set and then for SHR would have a message queue

also responses going back out

MT - See document at:

 [https://wiki.ohie.org/display/SUB/Primary+Use+Cases+of+OpenHIE](https://wiki.ohie.org/display/SUB/Primary%2BUse%2BCases%2Bof%2BOpenHIE)

For RHEA implementation, have asynchronous model

In Candian model, can poll systems that are not always “online”

Have identified two major compoents - RC will write up and send out by email - asked for comments in email thread

**Action Items:**

Ryan C will write up and send out by email - asked for comments in email thread

RC will touch base with Chris Ford and Derek Ritz for their comments as well