TRANSITION

GAPS
INTEGRATION
ARCHITECTURAL
STUDY

January 1, 2016

Prepared for:
New World Health

galenhealthcare.com

GALEN
HEALTHCARE SOLUTIONS

GAPS Integration Architectural Study

Table of Contents

Executive Summary 2

Current State Assessment 3

Future State Architecture 6

Statement of Confidentiality

for verifying the message definitions against a cache of production messages to ensure that the existing messages can be parsed correctly by the new Health Connect message definitions. A conservative estimate is that a difficult definition can take up to 1 day to complete (build and test).

Two items to note:

1. The Health Connect message parser follows the HL7 standard in all aspects. The HL7 standard dictates that if a composite field is null then the lower level delimiters are not placed in the message. Some other engines, such as eGate do not follow this convention.

This is only an issue when the receiving system(s) interface has been programmed to expect the lower level delimiters and will experience an error when receiving a message without the lower level delimiters. If this should occur there are two general courses of action. 1) Reprogram the receiving system to no longer require the delimiters in the message. 2) Configure the Health Connect interface to add the delimiters to the output messages (this would need to be done manually with code as this is not a setting or option).

2. It was noted during the GAPS analysis that many manipulations within JCAPS / SeeBeyond were being accomplished by manipulating the entire message as a string. This was method was utilized because of both the comfort of the programmers and the difficulty in creating message definitions with the current product(s).

Galen highly recommends using the features and functionality built into Health Connect to accomplish message manipulation. This means parsing the message and accessing segments and fields directly. It is by far the most efficient and recommended approach to take. Since most filters and routes in Health Connect are designed to parse the message immediately, a filter such as the mapper will not function correctly if it cannot parse the entire message. Therefore, it is vital to have solid message definitions for each system you will be interacting with. If there is reason to need to manipulate a message without parsing it, a javascript filter would likely be the easiest method. Setting any var equal to input[0].text will return the message body as a string. One example of a valid reason to do this would be to prepend or append some characters to the entire message. It is not recommended to use this feature to manually split the message by "|".

Publication and Subscription Model

The current integration platform employs a publication and subscription model for passing messages from the input connectors through the translation layer and to the output connectors. This implementation utilizes a JMS queuing layer that lies between the publisher and subscriber and effectively decouples all the components. In some cases this can present a problem when migrating to the Health Connect "push" model where messages are passed directly from component to component creating a more holistic view of the lifecycle of the message as it passes through the interface.

The results of this analysis yielded no significant use of the complex capabilities of the publication and subscription model or direct use of the standard JMS queue technology to route messages. This straightforward implementation will present no migration issues when deploying the routing of messages in the Health Connect environment.

Another area of concern when analyzing eGate configurations is accounting for the ability of the collaboration to publish different event types during the course of processing a single message. This can also create significant challenges when migrating to Health Connect as the mapper filter in Health Connect is limited to a single message type for the output of any given message. There are a few instances where this functionality is utilized in the current eGate interfaces. Those interfaces are identified in Appendix A. Although

translating this functionality to Health Connect can be difficult at times, the use of this feature at Hershey Medical Center is fairly straightforward and can be replicated in Health Connect through the additional use of a JavaScript filter.

Connectors

The majority of the connectors/eWays used in the Hershey Medical Center interfaces are standard TCP/IP and file-based with no significant custom configuration required. There are comparable Health Connect com points that support these protocols so creating the same connectivity as part of the migration is a straightforward configuration requiring basic Health Connect skills.

One significant difference that must be accounted for is the way that message acknowledgments are handled with TCP/IP interfaces. The TCP HL7 Monk eWay handles the HL7 acknowledgment directly from the connection slice of the eWay. When an HL7 message is received in eGate the ACK is generated immediately from the eWay completing the interaction with the sending system. In Health Connect, the acknowledgment is abstracted away from the connector and into the Route component. The ACK is generated by the ACK filter and sent back to the original sender. While the vast majority of situations will experience no noticeable difference in transactions, some additional attention should be given to message processing during development and support of the interfaces. Since the ACK is treated as a message in the Health Connect Web Management Console and can be tracked accordingly during the support process. During development the ACK pathway must be added deliberately and the associated components configured accordingly to ensure the proper acknowledgment handling procedures are in place.

File Lookups

Several of the Monk collaborations look up values from local, external files using the file manipulation commands. While the datamap function in the Health Connect was developed to function in a similar manner as the look up functions in Monk, the use of the Health Connect look up tables is preferred. The Health Connect look up tables utilize a multi-column table in the engine rather than an external file. This allows for the table to be captured as part of the configuration when exporting and importing the interface configurations.

Key Observations

- No significant occurrence of complex interface logic contained in the collaborations
- No significant customized event type definitions
- Publication and subscription implementation will not create issues when recreating the message flow in Health Connect
- No non-standard or highly complex use of any eGate technology that cannot be replicated in Health Connect

Engine Notes

eGate:

Most .tsc files can be used as a baseline to create a mapping definition file for Health Connect.

After analyzing the .tsc files the following items were identified:

- crs_Star11AdtFilter.tsc, MedAsset_StarAdtFilterV22B.tsc, StarAdtFilterV22B.tsc, StarGenAdtFilter.tsc have many iq-put staements. These .tsc files are being used to define rules about messages being delivered to multiple queues if criteria is met. This type of logic can be easily implemented in Health Connect but using conditional connectors.

- There are 14 .tsc files which use the data-map functionality of eGate. To convert these translations into Health Connect, a lookup table will replace the eGate .dat file referenced in the following colaborations:
  - crs_Cerner_Scheduling.tsc: 1 data-map
  - crs_Cerner_Scheduling_test.tsc: 1 data-map
  - crs_PowerWorks_CoPath_ORU.tsc: 1 data-map
  - crs_PowerWorks_CoPath_ORU_old.tsc: 1 data-map
  - crs_Resub.tsc: 1 data-map
  - crs_StarADT_BedStat1_Cerner.tsc: 1 data-map
  - crs_StarADT_BedStat2_Cerner.tsc: 1 data-map
  - crs_StarADT_BedStat3_Cerner.tsc: 1 data-map
  - crs_StarADT_Cerner.tsc: 6 data-maps
  - crs_StarADT_Cerner_prod.tsc: 3 data-maps
  - crs_StarADT_Cerner_save0921.tsc: 3 data-maps
  - crs_StarADT_Cerner_win_email.tsc: 6 data-maps
  - crs_StarADT_Cerner081408_backup.tsc: 6 data-maps
  - crs_StarADT_Cerner102307.tsc: 6 data-maps

ADT Migration Estimates

An estimate of the time required to migrate the existing ADT interfaces from eGate to Health Connect was requested by NWH. The goal is to migrate ADT interfaces to Health Connect by an October timeframe.

Based on the components analyzed in the table in Appendix at the amount of time required to migrate all interfaces to Health Connect conservatively totals approximately 600 hours.

By this estimate migrating the existing eGate ADT interfaces (a subset of the interfaces in Appendix A) by an October timeframe is feasible.

To further ensure that this timeframe is met it is recommended that more than one resource be dedicated to the effort to shorten the calendar time. It is also recommended that additional time be contracted with Galen to review work to date in the July/August timeframe to ensure that timelines are being met and development best practices are being followed.

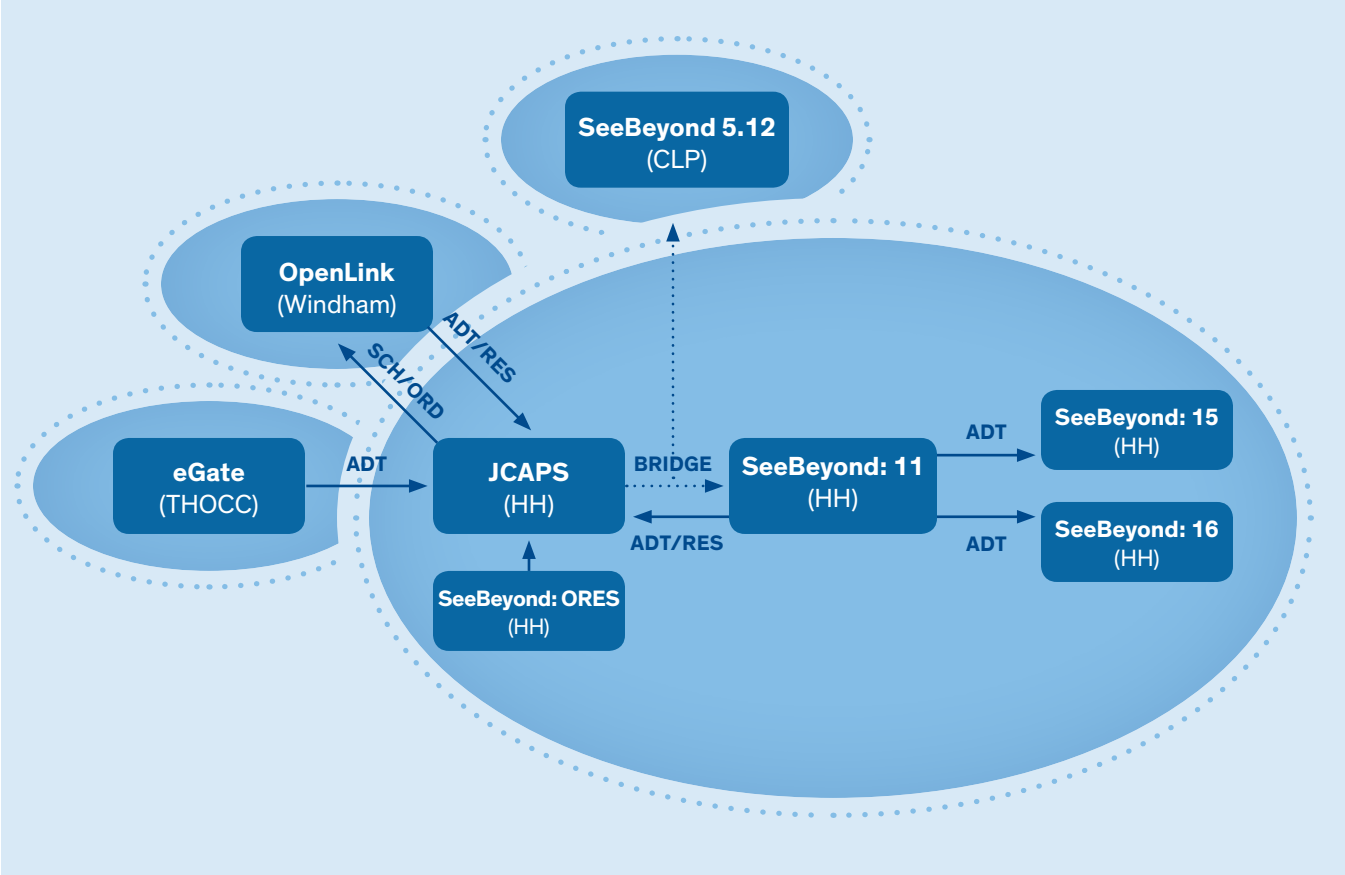
Future State Architecture

The future state Health Connect architecture, once implemented, will provide all the existing interface functionality with a modular design to facilitate future change and growth. An interim architecture can be put in place in advance of the completed migration to aid in the move to Health Connect. The “Health Connect Bubble” is an interim architecture which inserts Health Connect connectivity between the current integration platforms and the external system. This helps build project momentum and validate connectivity using production feeds with minimal impact on the existing interfaces.

Ultimately the final architecture will provide:

- Simplified operational support and administration
- Limited points of failure
- Minimized impact on environment when modifying or implementing new interfaces
- A more “visual” layout of the interface landscape

GAPS Integration Architectural Study



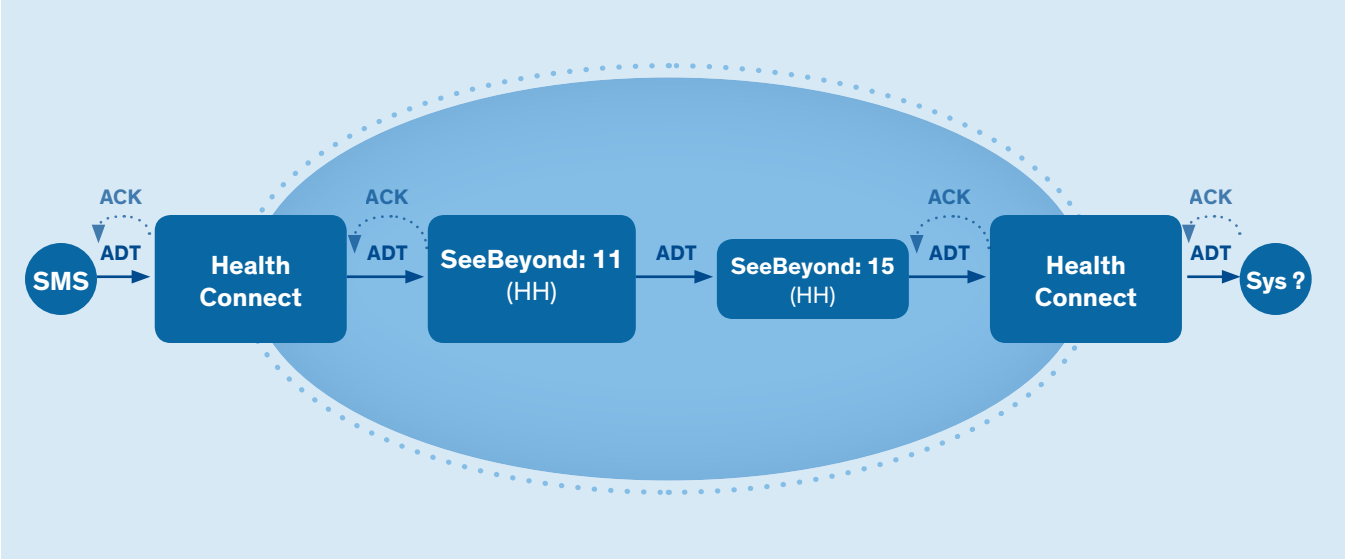
Health Connect Bubble

The "Health Connect Bubble" encapsulates the current integration environment within a Health Connect configuration by inserting Health Connect pass-through interfaces between the current interfaces and the external systems. It is recommended that this "bubble" be implemented first in a test environment.

The current New World Health environment is complex, having 8 integration engines. The message flow was analyzed to determine the appropriate method to implement the "bubble". It is important to build Health Connect's framework correctly from the beginning to minimize the amount of rework and changes needed to completely remove the legacy engines. Galen recommends approaching this project in pieces which can be accomplished simultaneously with no adverse impact if desired.

The drawing above represents the current integration environment overlaid with Health Connect "bubbles". Because the current OpenLink, eGate, and CLP SeeBeyond largely function independently, their migration can be approached independent of each other. However, the JCAPS and the four SeeBeyond engines are part of a system of message flow that should not be decoupled. Therefore, it will not be necessary to place Health Connect in-between any of these engines, but rather treat them as a system working as one.

GAPS Integration Architectural Study



The basic design places an inbound route that receives messages from external sending systems that has a communication point configured to receive the message and send the acknowledgment back to the original sender. The output of the route is a communication point that sends the unaltered message to the existing legacy engine interface via the same communication protocol (i.e. TCP/IP) and accepts the resulting acknowledgment. This configuration requires no change to the legacy engine interface. The only additional concern when creating this configuration is that the sending system will need to be reconfigured to send to the Health Connect server and port. This may require vendor support for that specific application.

The design of the "bubble" (see next section: Modular Design) is such that almost no modification is needed to the existing working environment. This is accomplished by inserting Health Connect in-between the sending systems and the current interface engine. Additionally, Health Connect is inserted in-between the current interface engine and messages going to the external systems. This means that messages will flow into Health Connect, which will generate an ACK back to the sending system. Health Connect will then pass the unaltered message to the existing interface engine. The existing engine will continue to perform as expected but instead of sending the message directly to the destination it will pass it back to Health Connect. Health Connect will then ACK the message back to the existing engine and pass the message on to the destination, again unaltered. This now inserts Health Connect into the picture and the next step is to build the translation logic into Health Connect and pass the messages directly to the destination, bypassing the existing engine(s). The only changes that need to be made to any existing systems is changing the IP address / Port that the sending system is connecting to (it needs to be Health Connect now) – the only exception is if the current interface sends to a system that resides behind a firewall. The receiving system's firewall settings may need to be configured to receive data from the Health Connect server and port.

It is possible to have Health Connect be a synchronous pass through and shuttle the ACK received from the downstream system back to the sending system (rather than generating an ACK of its own). However, this is not the "bubble" approach and to truly decouple the current environment from the external systems, it is recommended to NOT use the synchronous configuration.

There are two significant benefits to using the "Bubble" configuration:

- **Connectivity**–Establishing and testing connectivity before the interface logic is tested. This two phased testing will make it easier to identify connectivity issues and interface business process issues separately. This bubble phase requires the creation of a working message definition that can parse the incoming messages. Since Health Connect uses a message definition to create acknowledgements a by-product of this phase is the foundation which you will use to build all of the configuration logic relating to this system.
- **Monitoring**–passing data through the Health Connect engine to and from the external systems provides a facility to gain familiarity with the administration and operational support of the Health Connect integration environment. The message archive will be populated and can be searched upon and the routes and communication points that comprise the "bubble" can be monitored through the Health Connect Web Management Console. This also provides an opportunity to fine tune the alerts and notifications to minimize false positives.

Pre-Build (Server Install, Business Requirements, Documentaion) → **Health Connect Build and Design** → **UAT, Migration, Go-Live**

Health Connect Migration Time Estimates

Engine	Msg Definitions	Map: Hard	Map: Medium	Map: Easy	Total Hours	Total Days	Total Weeks	Total Months
CLP	55	9	24	55	1552	194	38.8	9.0
OpenLink	46	0	6	40	784	98	19.6	4.5
eGate	24	0	14	65	936	117	23.4	5.4
SeeBeyond 5.12	60	13	38	59	1976	247	49.4	11.4
JCAPS	51	11	35	88	2024	253	50.6	11.7
Total	236	33	117	307				
Total Man Hours	1888	1056	1872	2456				

Adjustable Variables	
Total Team Members	7 interfaces engineers
Time on project (each)	4 hours/day
Msg Definitions	8 hours each
Map: Hard	32 hours each
Map: Medium	16 hours each
Map: Easy	8 hours each
Error Margin	10%

Health Connect Build Totals	
Total Hours	7999.2
Total Days	999.9
Total Weeks	199.98
Total Months	46.1
Days to Complete	286
Weeks to Complete	57
Months to Complete	13.2