Automating the Migration of UKHC's Interfaces

Background

There are over 300 interfaces supporting the integration of UKHC's portfolio of clinical and administrative applications. This number is increasing over time, as digital interoperability within the hospital and between hospital partners also continues to increase. In order to assure cost sustainability over time, enable updates to the technology stack supporting the integration platform, and assuring minimal to no disruption to hospital operations, the current UKHC integration platform was designed in a manner to allow for integration interfaces to be automatically migrated to alternate technology stacks.

In addition to the requirements of the integration platform in this RFP, maintaining an interface implementation approach that continues to enable an automated migration to alternate technology stacks and being able to incorporate an automated migration of the current interfaces, would be value add components of a response to this RFP.

In order to allow offerors to assess their ability to support an automated migration of UKHC's current interfaces, as well as to propose an innovative strategy to future proof interface implementation to maintain this ability, this section of the RFP describes the current implementation of these interfaces in further detail.

Common Orchestration Flows

UKHC's Integration Platform is designed around the concept of Common Orchestration Flows that are not specific to any given system sending in data or any given endpoint receiving data from the exchange. These Common Flows do not need to be "automatically migrated" as they are considered the underlying orchestration flows common across a wide majority of separate interfaces going through the platform. The main flows include (1) Publication Flow which is an orchestration to receive HL7 messages from any number of receiving systems and publish them on a queue, (2) Notification Flow which is an orchestration allowing any number of outbound notification interfaces to consume messages from an endpoint specific queue and route them to a given destination, and (3) Synchronous Flow which is an orchestration allowing for a synchronous flow between sending system and an endpoint system with inbound message directed to the endpoint system and the response from the endpoint system directed back to the sending system synchronously.

These Common Orchestrations are similar in scope to the implementation of three (3) complex HL7 interfaces, in that they provide underlying common functionality, logging, error handling, that apply across all interfaces. They also provide for existing "Policy Enforcement Points" which are the configurable sections of the flow that can be tailored to a given sending system or endpoint system through dynamic configuration loaded in the form of XML based policy files. It is in the form of these policy files, where most of the "automation" would be directed to migrate legacy interfaces into the new technology stack, and the design for these policy files are described in the following section.

Common Flow SLAs and Policies

The vast majority of migration of UKHC's current integration, to a new technology stack, would be in the form of porting the current policy files that are applied across sending system traffic or to endpoint delivery to handle the specific variations and requirements of the integrating systems. These policy files are XML based files that are associated to a Service Level Agreement (SLA) which in the current solution acts as a "contract" for a given integrating system, and bundles the specific policy files needed to inform the runtime as to what to do with the transaction from or to the integrating system.

Currently, within the Integration Platform, there are 315 SLAs implemented across the following categories including 9 SLAs for receiving information (Publication Flow), 303 SLAs for sending out information to endpoint systems (e.g. Notification Flow), and 3 SLAs for synchronous query flows. These numbers provide a good idea as to the number of distinct systems interacting with the Integration Platform in terms of sending, receiving, or querying.

These SLAs will bundle a dozen or more policies, however the vast majority of the migration work involved in porting existing interfaces to a new technology stack would involve the Transformation Policy. For the purposes of this RFP, offerors providing a solution that automates simply the migration of the Transformation Policy, and respective QA/Test strategy to assure that migration has no defects, will be sufficient given this handles the majority of migration concerns at this time.

Technical details on the Transformation Policy has been provided as follows.

Technical Specification to the Transformation Policy

The majority of the interface work when onboarding a system to the Integration Platform involves message specific processing, such as correcting message schema, removing specific message elements that may not be supported by a given endpoint system, or other transformation related tasks.

Transformation can occur across all flows, including Publication (inbound messages being “normalized” before they hit the queue to be routed to subscribers), Notification (outbound messages being “localized” to the messaging standard of a given endpoint system), and Synchronous (inbound transformation between a sender and endpoint as well as outbound transformation between the endpoint and sender).

A common architecture for Transformation has been used in the platform, as applied to SLAs that relate to Publication, Notification, or Synchronous flows. Automating the migration of these transformations would account for perhaps 95% of the overall migration effort to port the interfaces to a new platform.

An example of a transformation policy is supplied as follows:

```
<wsp:Policy>
  <wsbm:Transform>

<wsbm:TemplateName Type="ESQL" Param='HL7.PD1."PD1.(1-13)", HL7.PID."PID.(1-
40)"'>com.ukhc.eii.format.esql.Transform_Enforce_Fields_Terminate</wsbm:TemplateName>

 <wsbm:TemplateName Type="ESQL" Param='HL7.PV1."PV1.(1-59)", HL7.DG1."DG1.(1-
18)"'>com.ukhc.eii.format.esql.Transform_Enforce_Fields_All_Paths</wsbm:TemplateName>

<wsbm:TemplateName Type="ESQL" Param='HL7.PID."PID.10"'>
com.ukhc.eii.format.esql.Transform_Subcomponent_to_Repeating</wsbm:TemplateName>

<wsbm:TemplateName Type="XSLT">REQ_HQ_XFORM</wsbm:TemplateName>

  </wsbm:Transform>
</wsp:Policy>
```

Note, in this example transform, there are a number of “TemplateName” elements and each of these elements represents a specific action to be done on the transaction. These actions are executed in declarative order from top to bottom, or first child element to the last child element.

Transformation Actions come in two flavors, with a first flavor being a “Function Call” as highlighted in Yellow in the policy file and marked as “ESQL” type. This action type essentially invokes a pre-built function within the Integration Platform, passing in “Parameters” that are defined within the policy itself, and these pre-built functions then operate on the transaction accordingly. For example, in the above example, the “Transform_Enforce_Fields_Terminate” is a function that can go through specific message paths, and enforce a given number of fields in that path, so in this example, passing in the parameters of “'HL7.PD1."PD1.(1-13)", HL7.PID."PID.(1-40)" as identified in the policy, which is a comma-delimited list of parameters so HL7.PD1.PD1.(1-13) being a first message path, and HL7.PID.PID.(1-40) being a second message path, and the function will iterate through these message paths, and assure that fields 1 through 13 for the PD1 message segment are all enforced in the message (if some fields are missing they will be added), and fields 1 through 40 for the PID segment will be similarly enforced and added if needed.

There are approximately 35 ESQL functions in use across interfaces, and these underlying functions do not need to be “migrated” automatically, they can be ported through coding effort to the new platform, but the policy that calls these functions at runtime across the common orchestration would need to be automatically migrated and behave in a consistent manner to the current platform.

As shown in the policy example, there is also an action type for “XSLT”, in this case, loading the XSLT associated with the REQ_HQ_XFORM identifier. The current platform maintains a cache of XML documents, including XSLT documents, and are identified by string identifiers. The transformation policy can specify an XSLT to execute at runtime against the transaction, allowing for a wide range of transformation to be applied against the transaction through the configured XSLT file associated with the policy. The XSLT files within the solution are all compliant with XSLT 2.0.

The existing platform also uses a common XSLT file to encapsulate some platform wide functions available across all XSLTs, so the XSLTs must be able to work with XSLT inclusions (e.g. <xsl:include>). Most XSLT 2.0 compliant parsers will support inclusions, but whether an inclusion resolves at runtime from an in-memory cache or whether the files would need to be flushed to disk by the runtime layer, are important considerations to assure the XML inclusions work at runtime.

XSLT can only be applied to XML formats, so currently all traffic flowing through the Integration Platform is normalized to XML. This normalization occurs at the edge of the current solution (e.g. HL7 MLLP transformed to XML equivalent at the front door of the solution).

Summary of Migration Automation

The UKHC Integration Team is open to alternative strategies that can be used to load transformations in a more protocol agnostic manner (e.g. XML and JSON) and load transformations that can be dynamically configured for a given sender or endpoint. This could be done, for example, by extending the transformation policy concept with a third action type, that supports a different transformation template than XSLT, and depending on its characteristics, it could become a preferred option for new interfaces as long as it maintains that ability to “automate migrations” of interfaces across future technology stack upgrades.