Rhapsody

Rhapsody – White Paper

Migrating from a Legacy Integration Engine

Replace an old platform in five easy phases

Migrating from a Legacy Integration Engine

Introduction

To leverage cloud and mobile technologies, healthcare organisations should consider replacing their older integration platforms. By doing this, they can avoid relying on developers with knowledge of specialised algorithmic programming languages (e.g., Monk) and eliminate the need to support platforms that have been discontinued, are no longer supported, or don't support contemporary integration requirements. Based on Rhapsody's extensive experience in migrating customers from legacy engines, this white paper describes a five-step approach to realising this goal.

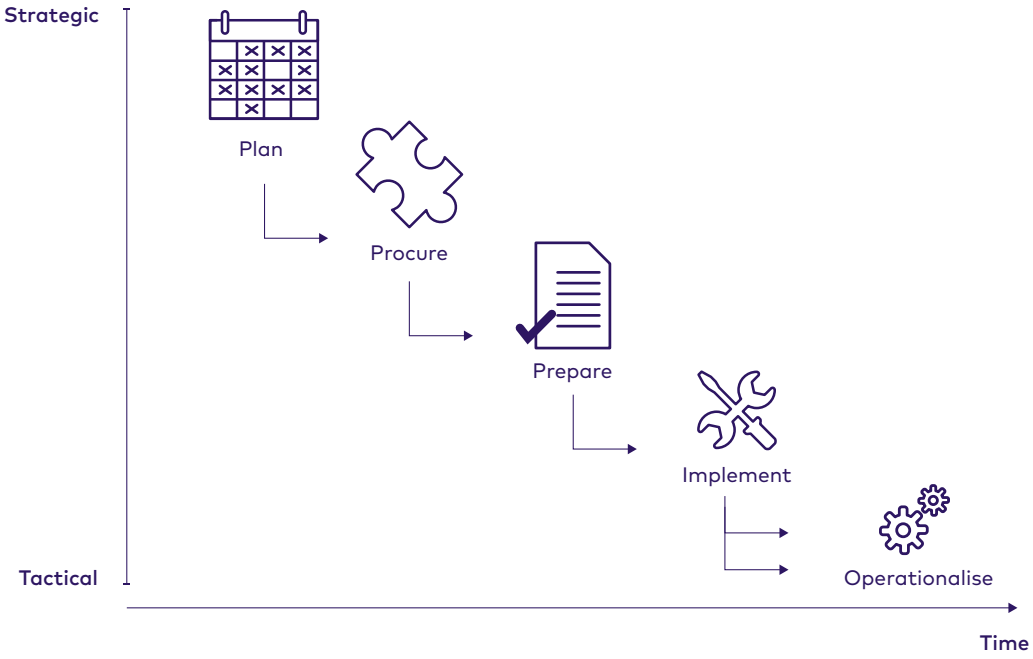


Figure 1: Five Steps Migration Process

Migration Steps

Step 1: Plan

The first migration step should be to formulate a customised plan for the healthcare organisation. The relevant components of the project plan should include the scope, technical requirements, dependencies, a timeframe, and a planning checklist.

Scope

The scope of a migration project may be quite narrow, limited only to replacing discontinued products and re-implementing the affected interfaces.

We find many organisations choose to take advantage of a fresh start on a new product to improve existing interfaces, deploy new interfaces, and/or consolidate interfaces onto a single platform.

Migrating from a Legacy Integration Engine

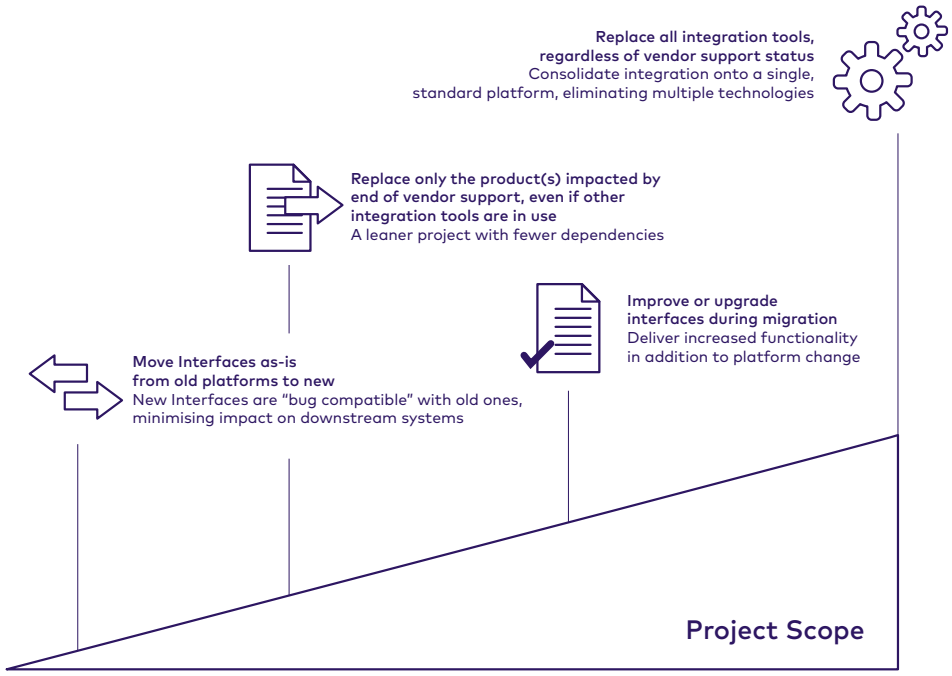


Figure 2: Project Scope

As scoping decisions are finalised, a clear picture will emerge of how many interfaces must be implemented on the new platform. This number will become the most basic measure of the project scope and help an organisation determine the size and expertise required.

Technical Requirements

Defining the functional and non-functional requirements at the beginning of the project will guide the project team's decisions and set the stage for a successful outcome. High-level requirements should answer the following questions:

- Which clinical systems need to connect to which other systems?
- What technical environments will the integration tool run in (e.g., hardware, software, operating system, etc.)?
- What type of, and how many, environments are needed to support the organisation's development process (e.g., development, test, production, etc.)?
- What existing policies around security, auditing, disaster recovery, and uptime will the new technology need to conform to?
- What are the performance metrics in terms of scalability, throughput, and uptime?
- What future considerations need to be taken into account?
- What other technologies could be folded into the new engine (e.g., FTP movers, file listeners, email generators, web service utilities, etc.)?
- What messaging standards do the interfaces adhere to?
- What messaging, protocol, and security standards need to be supported?

Migrating from a Legacy Integration Engine

Listed in the following chart are examples of built-in protocols:

A sample analysis of components is provided below:

Protocol	Secure Option
TCP/IP	Yes
HTTP	Yes
Web Services - Call	Yes
Web Services - Host	Yes
Database	Authenticated
Email	No
JMS, MSMQ, IBMMQ	No
Flat or Zip File	No
FTP Client	SFTP and FTPS

Dependencies

As a project is not executed in a vacuum, it is important to take dependencies into account. Dependencies in a project may result from other projects. A growing trend in the industry involves combining engine migration with other system migration initiatives. We advise clients to roll out their legacy migration first, or concurrently, to eliminate interface re-work.

Planning Checklist

Planning is a critical phase in any project. This checklist summarises the necessary steps for planning your migration.

- ☐ Determine the project's scope—broad, narrow, or in-between—using the number of interfaces to be completed as your guideline
- ☐ Document key technical requirements
- ☐ Determine project dependencies
- ☐ Plan the project completion dates
- ☐ Establish your resource requirements using the number of interfaces and the project timeline as your guideline

Step 2: Procure

Once a project plan is developed, the next step is to identify a replacement engine. In this step, the engine should be evaluated based on product capabilities, service offerings, and vendor experience. Another important factor is the Total Cost of Ownership (TCO) of the new platform compared to your existing platform. Some vendors will charge ongoing support for features that are not required by the healthcare organisation. Additionally, organisations should take into consideration (1) services provided, such as the type of initial and ongoing training, and (2) the ease with which support is obtained.

When looking for a new integration engine, you need to consider ongoing client support. What is included in the support and maintenance agreement? Does the new integration engine vendor provide initial and ongoing training on the product? Is there a certification program available? Is customer support accessed through email, telephone, or online chat? Is there an online ticketing system that makes it easy to trace issues? Ongoing client support is an important factor to consider when comparing integration engine vendors.

Migrating from a Legacy Integration Engine

Evaluation Process

The evaluation criteria should come directly from the requirements developed during the planning phase. To recognise whether a particular solution will meet your requirements, it is best practice to evaluate vendors by asking for a "Proof of Concept" to demonstrate their ability to migrate existing interfaces.

Procurement Checklist

- ☐ Develop evaluation criteria based on project requirements
- ☐ Prioritise requirements into "Must Have" and "Nice to Have" categories so that a best-fit solution can be identified
- ☐ Identify the interface engine
- ☐ Evaluate the integration engine according to consistent criteria
- ☐ Request and evaluate "Proof of Concept" demonstrations

Partnering with vendors that have successfully migrated legacy engines and have performed the appropriate gap analysis on components and functionality will accelerate your migration.

Another issue to consider is the requirements for developers—i.e., should they be proficient in specialised algorithmic programming languages? There are few developers with knowledge of older languages, such as Monk, and more developers with knowledge of modern languages, such as JavaScript.

A sample analysis of components is provided below:

eGate/SRE	JCAPS	Rhapsody
ETD	OTD	Message Definition (.s3d)
Collaboration	Connectivity Map	Route
Eway	External sys (adapter)	Communication Point (Comm. Pt.)
Collab Rule	Java Collab def (JCD)	Mapper definition file (.mdf)
Enterprise Designer	Netbeans	Rhapsody IDE
Enterprise Monitor	Enterprise Manager	Web Management Console (WMC)
ETD Tester	OTD Tester	EDI Analyzer/ Explorer
<programmatic>	<programmatic>	Conditional Connector
Registry Host	Repository	Rhapsody Engine (Config)
Monk Function or Java Method	Java Method	Filters
Shema/ Component export file	Project export (.zip)	.rlc file

Migrating from a Legacy Integration Engine

Step 3: Prepare

Provisioning

The most important preparation migration task is to ensure that proper hardware, software, and licenses are available.

It may be possible to repurpose servers from the legacy engine for the new engine. However, most organisations prefer to run the engine on newly acquired hardware for the following reasons:

- Typically, there is a period of time in which the legacy and new integration engine must run concurrently, creating the need for separate environments.
- If the engine is running on older hardware, implementing a new interfacing solution is a good opportunity to update the hardware at a relatively low incremental cost.
- Another alternative is to move the solution to a virtual environment, thereby utilising existing hardware, yet providing a portable mechanism for easier migration.

Staffing

The scope and timeline form the basis for determining the appropriate experience and number of staff for the entire project. A typical interface project team may include:

- Project managers
- Interface analysts/developers
- Application specialists (often part-time subject matter experts)
- Quality assurance analysts
- Operations analysts

Training

Training for analysts, developers, and operational staff also takes place during the preparation stage. Ensure that the vendor offers a combination of product documentation, online self-paced instruction, onsite training, and continued mentoring.

Prioritisation

Each interface should be prioritised and—optionally—classified according to the difficulty of the project. Often, clients can realise a substantial consolidation when migrating due to differing integration paradigms. It is also advisable to highlight interfaces to be added due to increased functionality provided by the new integration platform.

> "Rhapsody's ease of use combined with powerful functionality allowed us to migrate quickly while simultaneously enhancing integration between PCH applications."
>
> Kevin Allen, Senior Integration Analyst, Phoenix Children's Hospital (PCH)

The project team should develop a strategy for scheduling interfaces for migration. They may prefer to schedule high-priority interfaces first and lower priority interfaces later. Or they may start with easier interfaces and work up to harder ones as they gain familiarity with the new tools.

Preparation Checklist

- ☐ Provision hardware, software, and licenses required for the project
- ☐ Assemble an appropriately sized project team
- ☐ Conduct training for all team members
- ☐ List and prioritise all interfaces in scope for migration

Migrating from a Legacy Integration Engine

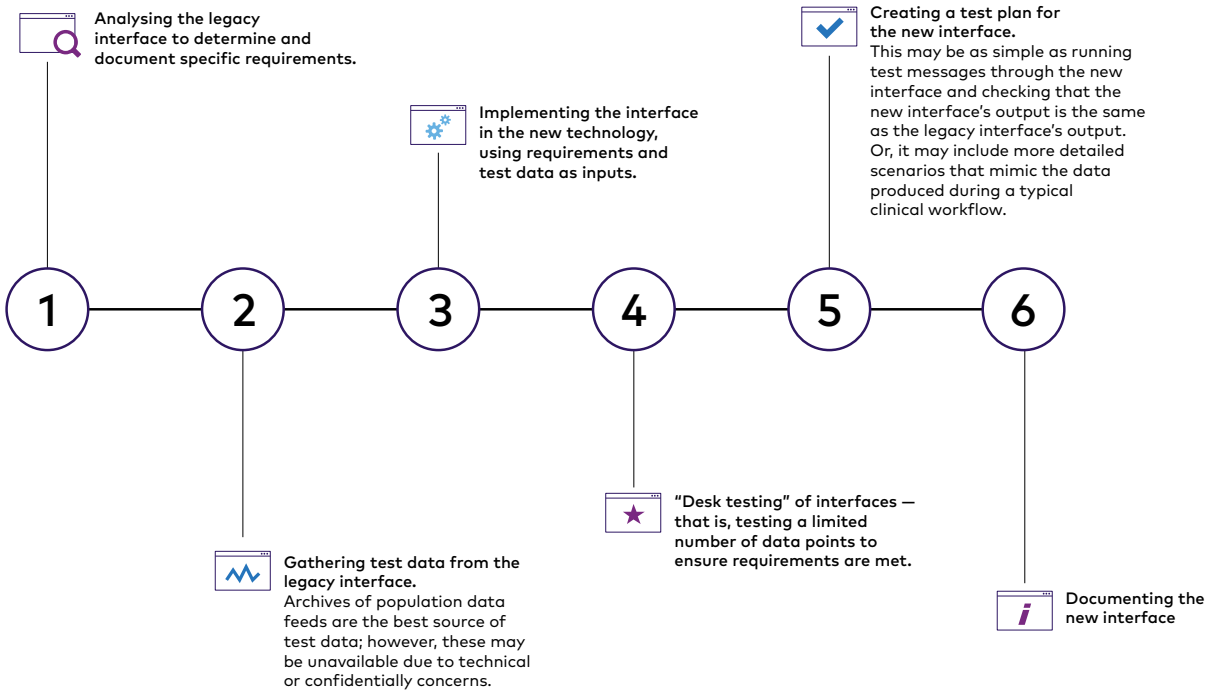


Figure 3: Integration Migration Steps

Step 4: Implement

Migration

Working through the previously created prioritised list of interfaces, the project team can migrate each interface individually or in bulk. Typically, this consists of the steps documented in the illustration below:

Intelligent Mapper

One of the most challenging aspects of an integration engine migration is recreating the custom logic that was built into the legacy interface engine. While some engines, including Rhapsody, offer tools that help translate the legacy code into a compatible language, this approach is problematic at best.

The challenges of migrating logic can stem from a combination of the following:

- A lack of documentation of the legacy logic
- The differing stylistic approaches of multiple developers, which can make the logic difficult to follow
- A minimal-risk approach to not altering existing logic (i.e., appending the end results with additional logic that is both convoluted and inefficient)
- The lack of functionality of many legacy engines (legacy engines require exponentially more coding than modern engines like Rhapsody do)

To address this, we recommend an approach that does not carry forward the same issues into the new engine and capitalises on the vast improvements of logic, workflow, and functionality that come with a modern engine, which would be impossible in a direct translation of the legacy code.

When considering a new engine, look for one that comes bundled with tools that will help your team recreate existing logic. One such example is the Rhapsody Intelligent Mapper. Rather than build logic by going line by line through legacy code and relying on human trial and error, the Intelligent Mapper—using advanced algorithms to reverse engineer the logic—analyses multiple before-and-after messages that have been successfully processed through your legacy engine. This approach removes the need to map outdated languages and/or carry forward the

Migrating from a Legacy Integration Engine

previous inefficient design. Instead, the Intelligent Mapper designs a streamlined approach to create identical message translations within Rhapsody. The end result is logic that is—going forward—easier to maintain, automatically documented, and standardised.

Other Considerations

In addition to migrating individual interfaces, non-interfaces must be implemented. These include such items as:

- Backup and recovery plans in a highly available, clustered, disaster-proofed environment
- Monitoring strategies related to proactive alerting
- Processes to deal with system event notifications
- Tactics for the maintenance of users, roles, and security settings
- Strategies for acclimating to a paradigm shift to modernised engine architecture

> "With eGate some of our biggest challenges were monitoring the interfaces and operational reporting. These pain points have been eliminated with Rhapsody's configurable web management console, which enables us to proactively manage our interfaces and report granular information on each."
>
> Corey Smith, Manager IT Integration, University Hospitals Management Service Center

Testing

To prove that the interface meets the requirements defined at the beginning of the project and to ensure correctness of outputs, the first step in testing an interface is functional testing. Look for an integration engine that streamlines this process by enabling granular component testing, which can eliminate time-intensive, end-to-end test cycles. The next step is to send a high volume of messages through the interface to ensure adequate performance and correct outputs under expected production load volumes.

> "Our organisation has been quite satisfied with the dependability and speed of development turnaround time with the Rhapsody Integration Engine."
>
> Harvey Wittmayer, IT Application Manager, Trinity Health

Migration Methodology

Minimise Risk, Control the Process, and Build Confidence

The most common strategy, although not always necessary, is to purchase new servers alongside a new interface engine. Whether new hardware or virtual machines, it makes sense to future-proof your investment with a server designed for growth. This coordination of hardware and software allows for a smooth transition to the new engine by following a phased cutover approach. The following example assumes three environments—development, QA, and production (active or active/passive)—with the new interface engine being installed on dedicated servers.

The following methodology has many key advantages:

- It's designed for a phased go-live of interfaces, including when one system's data is fed to multiple downstream systems.
- It minimises risk by requiring almost no changes to the existing interface engine, allowing for a rolling back to the old engine in an unexpected scenario.
- It introduces Rhapsody into the flow upfront, which prevents upstream connectivity from bottlenecking/delaying each go-live.
- It utilises "loopback compare" logic to alert the interface team of an unexpected difference between the legacy interface and the replacement Rhapsody interface. This allows the interface team to be the first responders rather than be blamed reactively by the downstream system owners. This builds confidence in the team that extends beyond the IT department.

Migrating from a Legacy Integration Engine

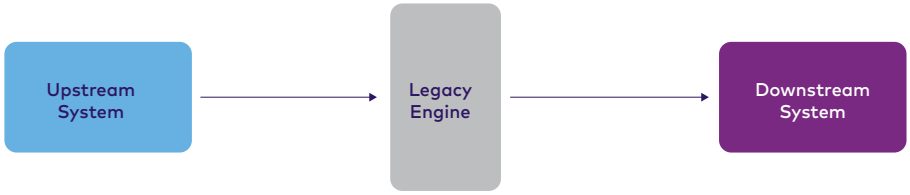
High-Level Steps

1. Install the Rhapsody engines on the new servers, making no changes to the existing integration engines.
2. Create a pass-through route in Dev Rhapsody for each inbound connection in the current legacy engine environment. This will be the foundation of the future completed interface. During the phased go-live, the interface will pass the messages unaltered to the legacy engine. As each completed interface is launched on Rhapsody, this route will be used to feed a copy of the message to be processed directly downstream. Eventually, once all downstream feeds are live in Rhapsody, the pass-through part will be retired, leaving the legacy engine starved of messages.
3. Promote Dev Rhapsody configuration to QA Rhapsody.
4. Point QA applications that send out messages to QA Rhapsody instead of the QA legacy engine. QA Rhapsody is now passing the messages, unaltered, to the QA legacy engine (i.e., its configuration is unchanged).
5. Promote QA Rhapsody configuration to Production Rhapsody.
6. Point production applications that send out messages to Production Rhapsody instead of the production legacy engine. Production Rhapsody is now passing the messages, unaltered, to the production legacy engine (i.e., its configuration is unchanged). This step completes the Bubble Phase (see Step 1 in the images below).
7. Build the interface logic of a single destination system in Dev Rhapsody.
8. Use the provided Message Compare – Bulk with Report logic to test the output. Correct issues as discovered until no differences are reported.
9. Promote the Dev Rhapsody interface to QA Rhapsody.
10. Point the outbound connection of the QA legacy engine to connect to QA Rhapsody on a Message Compare – Real Time route setup for this interface. This step allows the interface to be tested in real time in QA.
11. Promote the QA Rhapsody interface to Production Rhapsody.
12. Point the outbound connection of the production legacy engine to connect to Production Rhapsody on a Message Compare – Real Time route setup for this interface.
    a. This step completes the Live with Loopback Phase (see Step 2 in the images below).
    b. After a period of no issues, the Message Compare logic may be removed.
13. Repeat Steps 6-12 for each destination interface. Once all destinations of an inbound feed have been promoted to production, the pass-through connection from Rhapsody to the legacy engine can be turned off, effectively removing it from the flow.
14. Retire the legacy engine. This step completes the Retire Loopback Phase (see Step 3 on the following page).

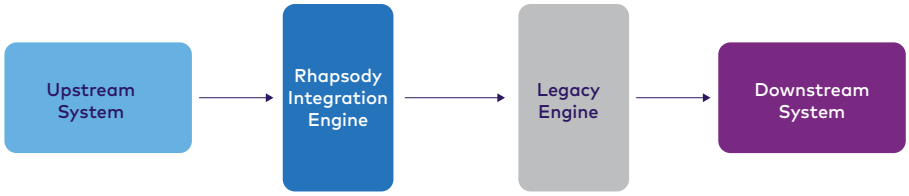
Migrating from a Legacy Integration Engine

Phased approach to introducing Rhapsody into your QA and Production flow

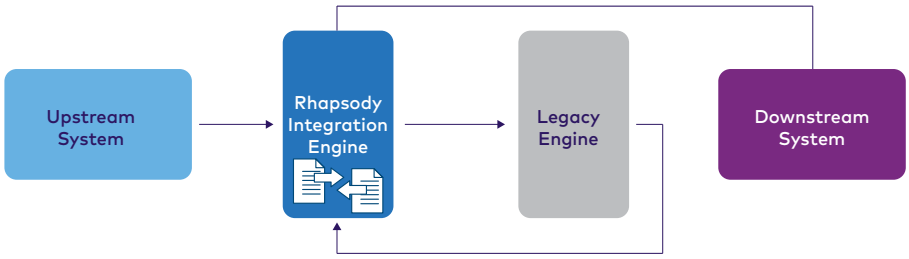
Production Engine Migration: Today



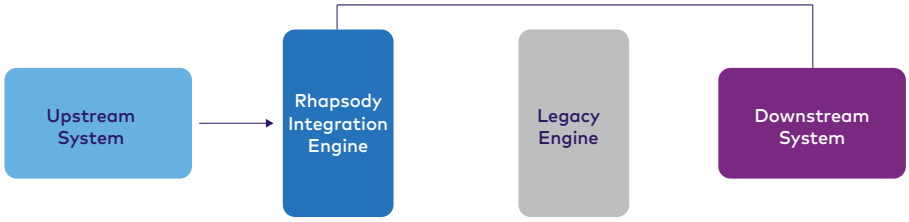
QA/Production Engine Migration: Step 1
Step 1: Rhapsody Bubble (Pass Through Interfaces)



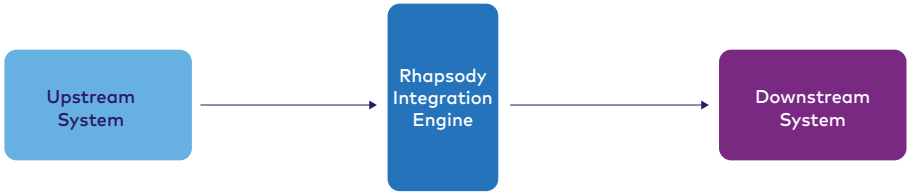
QA/Production Engine Migration: Step 2
Step 2: Legacy Loopback (Compare Outputs)



QA/Production Engine Migration: Step 3
Step 3: Retire Loopback (Turn off Legacy Feed)



QA/Production Engine Migration: Complete
Step 4: Retire Legacy Engine



Migrating from a Legacy Integration Engine

Implementation Checklist

- ☐ Gather requirements, then build, test, and document each interface
- ☐ Conduct testing and defect remediation for all interfaces

Step 5: Operationalise

Promote to Production

To ensure best practices, promotion should be done according to a defined, repeatable process. The contemporary engine's migration solution allows for the promotion of individual interfaces or groups of interfaces, depending on the client's preference.

After the migration is complete, the interface should be monitored closely for a "burn-in" period to ensure that unexpected failures are detected promptly. To ensure that downstream systems experience a seamless transition, the contemporary engine's Message Compare filter is ideal for automating this period.

Legacy Engine Retirement

When the operational team feels certain that the new interface is performing correctly, they may retire the engine by shutting down each legacy connection. After all of the legacy connections have been shut down, it is safe to decommission the engine.

Operational Checklist

- ☐ Promote new interfaces to production according to a defined, documented process
- ☐ Retire interfaces by turning off connections
- ☐ When all interfaces have been shut down, retire the engine

Conclusion

The integration challenge of replacing a legacy interface engine may seem monumental. However, as support for outdated legacy engines decreases, the decision to convert to a modern engine proves to be the most feasible option.

While migrating to any modern engine is better than maintaining the majority of legacy engines, choosing an engine that streamlines that migration will allow your team to focus on providing new value to the business rather than simply replicating the existing functionality of the engine being replaced. Additionally, it will equip your healthcare organisation with the tools for adopting and implementing cloud and mobile capabilities.

By investing in a contemporary platform and implementing the five phases outlined in this paper, your organisation will be future-proof.

Rhapsody-specific Definitions

Compare Filter: The Compare Filter compares two HL7 messages and reports any differences. It can be used to validate (1) that the replication of logic in a new engine is functioning as expected in real time or (2) historical bulk loading of messages.

Message Compare – Bulk with Report route: This sample implementation of the Compare Filter is designed to compare messages previously processed by both the Rhapsody and legacy interface engines. The compare filter is used to match and determine any differences, and a report is generated that consolidates the findings of all of the compared messages. This enables the interface designer to focus on the logic, built within Rhapsody, that is specific to any differences found in the report.

Message Compare – Real Time route: This sample implementation of the Compare Filter is designed to compare messages processed by both the Rhapsody and legacy interface engines during live message traversal. It is generally used as a validation tool during live message flow (e.g., through QA during testing, or production during a burn-in period after go-live).

Rhapsody

To learn more about how your organisation can replace a legacy integration platform with the Rhapsody® interoperability platform

Find out more at www.rhapsody.health

—

Rhapsody® Integration Engine is intended only for the electronic transfer, storage, or display of medical device data, or the electronic conversion of such data from one format to another in accordance with a preset specification as specified in the product manual and/or related documentation. Rhapsody Integration Engine is not intended to be used for active patient monitoring, controlling or altering the functions or parameters of any medical device, or any other purpose relating to data obtained directly or indirectly from a medical device other than the transfer, storage, and conversion of such data from one format to another in accordance with preset specifications. InterOperability Bidco, Inc., doing business as Rhapsody®, its affiliates and subsidiaries makes no warranties and the functionality described within may change without notice. ONC Health IT Certification (2014 Edition) Rhapsody Integration Engine and Rhapsody Connect attained 2014 Edition Modular Ambulatory EHR Certification and 2014 Edition Modular Inpatient EHR Certification from the ICSA Labs ONC Health IT Certification Program. This EHR Module is 2014 Edition compliant and has been certified by an ONC-ACB in accordance with the applicable certification criteria adopted by the Secretary of Health and Human Services. This certification does not represent an endorsement by the U.S. Department of Health and Human Services. For more information, please see www.rhapsody.health/meaningful-use. Rhapsody® is a registered trademark of InterOperability Bidco, Inc., manufactured in New Zealand, by InterOperability Bidco, Inc. All other trademarks displayed in this document are the property of InterOperability Bidco, Inc., doing business as Rhapsody®, its affiliates and subsidiaries or their respective owners, and may not be used without written permission of the owner. Rhapsody Integration Engine is not intended to be used for diagnostic purposes, or to replace clinical judgment or responsibilities. All patient information shown in any imagery is for representation and demonstration purposes only and is not related to a real patient.

Copyright © 2018 HealthInterOperability Bidco, Inc. doing business as Rhapsody® group of companies | All rights reserved | www.rhapsody.health