

Why Do I Need an Interface Engine?

Evaluating Two Approaches: Point-to-Point and Interface Engine

Overview

Introduction

Clinical care facilities typically use a variety of complex software applications from different vendors. Because these applications are created by different software teams, these applications need to exchange data and typically do so via interfaces. The de facto “language” used to move this data between applications is HL7, which is an international message standard.

When facilities build HL7 interfaces, they have a choice of constructing them point-to-point or by using an HL7 interface engine.

There are several challenges to building HL7 interfaces. An HL7 interface requires a sending and receiving module. These modules are created by the software vendor who programmed the application. Even though both applications use the HL7 message format, they rarely agree on the specific HL7 format that is used.

In order to bridge the differences in HL7 format, modifications need to be made to the sending or receiving modules or an interface engine is used in the middle to translate the messages.

The software vendors charge money to the clinical care facility in order to create the sending and receiving modules. Additional money is required for each additional module or for any modifications to the modules.

The presence of an HL7 interface engine in a healthcare environment gives more control to your organization and saves money and time by:

- Reducing the required number of export and import endpoints
- Allowing for reuse of data between applications
- Providing an easier method to interface a new or replaced application
- Providing the ability to monitor the entire system at one time
- Providing the ability to proactively notify interested persons using visual display and e-mail, when problems arise

Purpose

This white paper provides details on how an HL7 interface engine provides more control and saves time and money in a clinical or healthcare environment. It includes a comparison of an interfaced environment using point-to-point communications and an interfaced environment using an interface engine.

Problem

Introduction

The typical clinical facility or hospital has several HL7 enabled applications and devices. To reduce data entry time and increase overall efficiency of the facility, these applications or devices need to communicate with each other.

There are two basic ways this can be accomplished:

1. Point-to-point where each pair of applications communicates independently of other applications.
2. Using an interface engine that is placed between all the applications to aid in information exchange and monitoring.

Creating communication

To facilitate communication between two HL7 enabled applications, an HL7 interface is created. An HL7 interface includes:

- An export endpoint for the sending application
- An import endpoint for the receiving application
- A method of moving data between the two endpoints

The following figure shows an interface between the HIS system and the lab. The blue box represents the export endpoint for the HIS system and the yellow box represents the import endpoint for the Lab system.

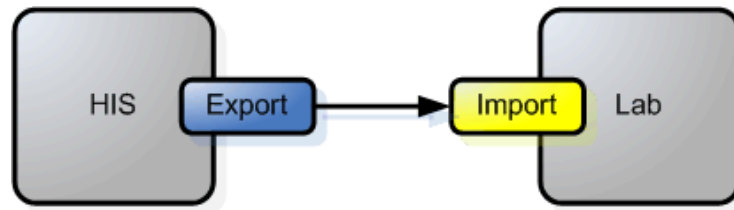


Figure 1: HL7 Interface

Facilities with point-to-point communication

Facilities that use the point-to-point model find that:

- It is very expensive to initially implement communications
- It is expensive and time consuming to add new or replace existing applications
- They spend considerable time and money to monitor and maintain the connections over the life of the applications

Facilities with interface engines

Facilities that use an interface engine model find that:

- It is much less expensive and takes less time to initially implement an interface because an engine allows for leveraging of data and an engine is flexible in its acceptance of data
- The cost and time required to add new or replace existing applications is frequently less than half that required in a point-to-point model
- It requires considerably less time and money to maintain and monitor the interfaces because of the availability of centralized monitoring

Data flow

The following figure shows workflow diagrams for point-to-point and interface engine model in a typical small acute care setting:

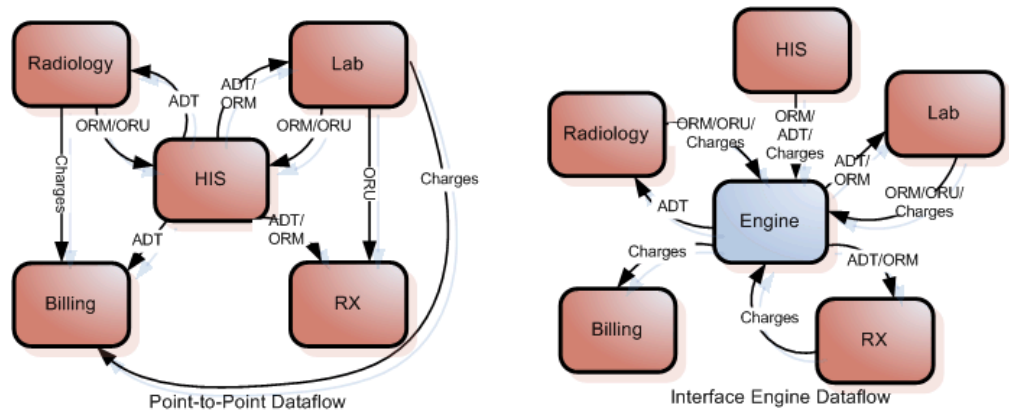


Figure 2: Dataflow for HL7 Interfaced Environments

Example workflow

The workflow of an example small acute care setting is shown in the following table:

Information Type	Workflow
Demographics	Entered in registration system and sent to all ancillary systems.
Orders	<p><u>Lab and Pharmacy</u> – Orders entered on order entry system (HIS) and ORM messages delivered electronically to both systems. An order response is sent from the Lab back to HIS. No response is sent from the Pharmacy.</p> <p><u>Radiology</u> – Entered on HIS but printed and re-entered into radiology. Once typed into radiology an electronic confirmation is sent to the HIS.</p>
Results	<u>Lab</u> – Sent to HIS and pharmacy

Information Type	Workflow
	<u>Radiology</u> – Sent to HIS
Charges	<u>Radiology, Lab, HIS, and Pharmacy</u> – all deliver electronic charges to Billing.

Point-to-Point Communications

Overview

The point-to-point communications model has each pair of applications communicating independently of the other applications in the environment through the use of export and import endpoints.

What are export and import endpoints?

Export and import endpoints are attached to an application to allow it to send and receive HL7 messages.

Typically, the export and import endpoints are required to:

- Filter data being sent or received
- Change the format of data to match the receiving application's needs
- Write the data into the receiving application's database

What do export and import endpoints cost?

An endpoint costs both time and money to implement including:

- \$5K to \$25K in fees and project costs to create each endpoint, depending upon the vendor

- After getting in the implementation queue for all involved vendors, an average of three months to configure and test but varies depending upon the application vendor

How many export and import endpoints are required?

In a point-to-point model, applications that are going to communicate must have an export and import endpoint designed specifically to interface with the other application. The number of export and import endpoints varies depending upon the type of communication required between the applications.

The following table shows how the number of endpoints required are calculated in a point-to-point model.

Type of communication	Number of Endpoints Required for Each Application	Total Number of Endpoints Needed	Cost of Endpoints at \$10K Each	Examples of Use Outbound (OB) Inbound (IB)
Bidirectional	Each application has: 1 export endpoint and 1 import endpoint	4	\$40,000	ADT data sent to pathology and results returned: <ul style="list-style-type: none"> • HIS needs 2 (OB ADT and IB Orders) • Pathology needs 2 (IB ADT and OB Results)
Unidirectional	Sending application - 1 export endpoint Receiving application - 1 import endpoint	2	\$20,000	ADT data sent to dietary: <ul style="list-style-type: none"> • Dietary needs 1 (IB ADT) • HIS needs 1 (OB ADT)

Example point-to-point figure

The following figure is an example of how the previously described workflow with HL7 enabled applications communicates with each other without the benefit of an interface engine.

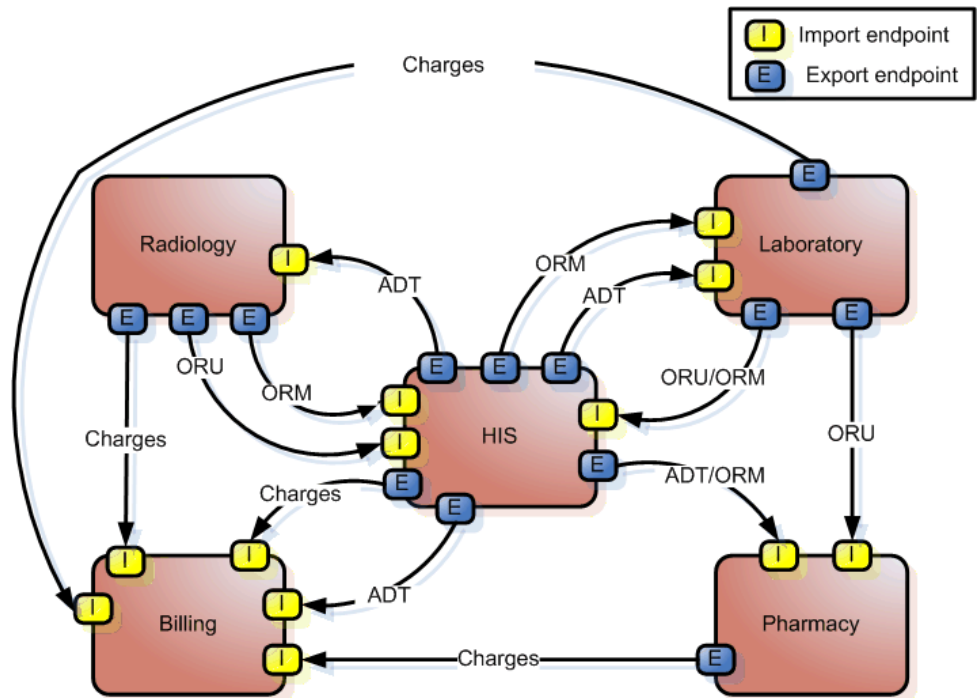


Figure 3: Point-to-Point Model Requires 26 endpoints at approximate cost of \$260K

Cost of implementing point-to-point model

The costs to implement the previous point-to-point example are:

- 26 endpoints at \$10K each = \$260K
- Effort: If each endpoint takes three calendar months to complete, worst case is 75 calendar months. However, if all endpoints were built at once, there would be significant schedule overlap. A reasonable estimate is a total of 24 calendar months. A primary driver in the schedule is the waiting time for each vendor – both end points of the interface must be constructed for each interface to go live and the implementation queue is often long.

Replacing or Adding Interfaces in Point-to-Point Model

Overview

Adding, changing, or replacing applications in a point-to-point interfaced environment affects not only the ancillary with the new application but potentially the entire system.

When a changed or updated application is introduced in an interfaced environment, all the applications that interface with the updated application are affected. Typically, this means all the endpoints for the updated application must be created or changed to continue communication. In addition, all the software vendors who have interfaces attached to the application have to replace or modify their endpoints.

When a new application is introduced in an interfaced environment, that software vendor must create interface endpoints for each application with which it will communicate. In addition, each of those software vendors will have to create an interface endpoint to use in communicating with the new application.

Process to replace applications

If an existing application is replaced:

- The export and import endpoints attached to the replaced application have to be created
- Most of the endpoints for applications that the replaced application interface with will also need to be modified

Example of replacing applications

The following figure illustrates replacing the lab application in the point-to-point model along with the new endpoints required:

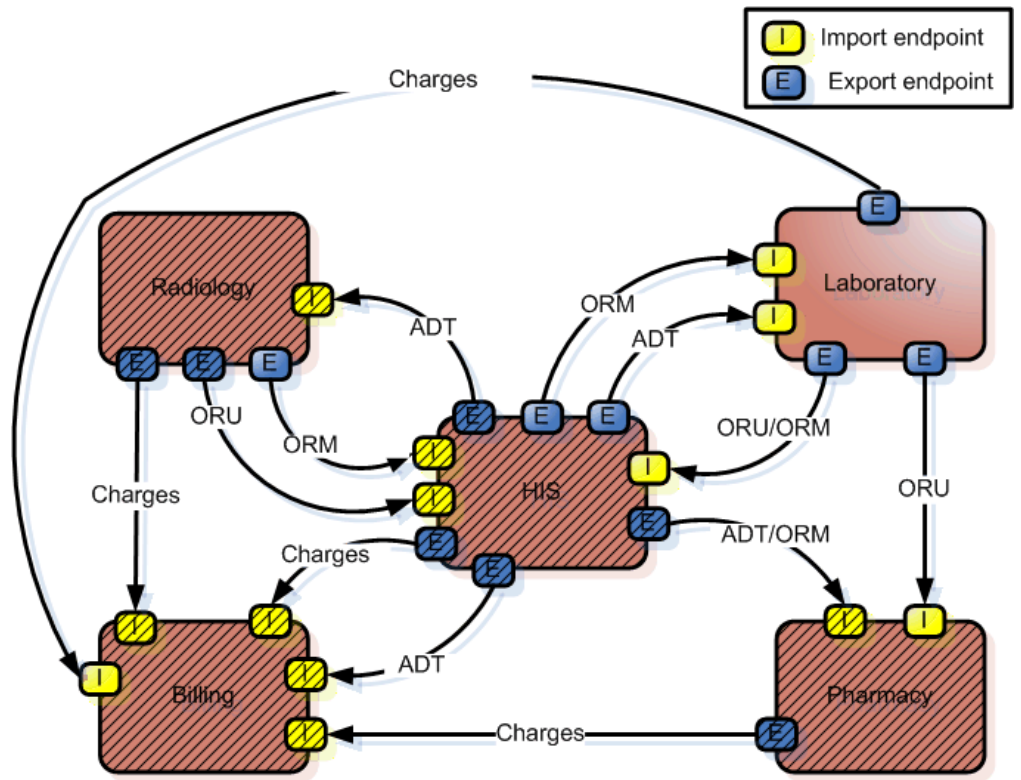


Figure 4: Replacing Point-to-Point Model Requires 10 interfaces at approximate cost of \$100K

Cost of replacing application example

The costs to replace the application in the previous point-to-point example are:

- 10 endpoints (not shaded) at \$10K each = \$100K
 - 5 new Laboratory endpoints
 - 3 updated or replaced endpoints for HIS
 - 1 updated or replaced endpoints for Pharmacy
 - 1 updated or replaced endpoints for Billing
- Test and workflow issues - \$25K
- Effort: Time waiting for four application vendors to create and update endpoints to communicate with the new application at 3 months each – 15 to 30 months counting for overlap and testing once all endpoints are created. As previously mentioned, a key driver is availability and scheduling of vendor staff.

Process to add applications

The cost to add a new application is inordinately expensive because existing endpoints can not be used without modification. In addition to the expense of a new application's endpoints, all applications which communicate with the new application must have additional endpoints.

Example of adding a new application

The following figure illustrates adding a new surgery application to the point-to-point example which needs bidirectional communications with the HIS and one-way communication with the billing system and laboratory.

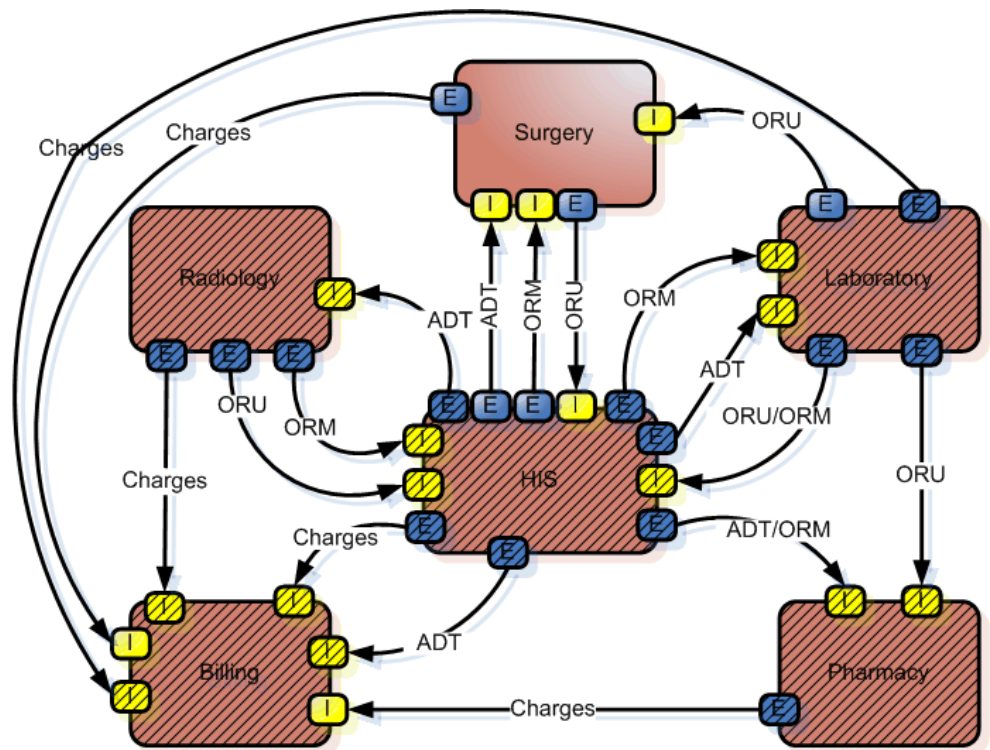


Figure 5: Adding Application in a Peer to Peer Model Requires 10 interfaces at approximate cost of \$100K

Cost of adding new application

The costs to add the surgery application in the previous point-to-point example are:

- 10 endpoints (not shaded) at 10K each = \$100,000

- Test and workflow issues - \$25K
- Effort: Time waiting for four application vendors to create and update endpoints to communicate with the new application at three months each – 15 to 30 months counting for overlap and testing once all endpoints are created

Monitoring Interfaces in Point-to-Point Models

Overview

Point-to-point communication models do not have any centralized monitoring.

The lack of centralized monitoring results in:

- More time and money required to monitor the interfaces
- Lack of meaningful, system-wide information available in a timely manner

Lack of knowledge

Lack of centralized monitoring typically results in no one knowing that a problem occurred until it has reached crisis mode. There are some utilities that can be used to monitor connections but are difficult to use and frequently do not provide the complete picture of the effectiveness of the connection.

For example, if the lab interface goes down, there is often no way to know until a doctor calls looking for lab results. Unfortunately, this typically means that several messages have not been delivered where they need to be and systems are backed up throughout the facility.

Finding communication problems

The lack of centralized monitoring also makes troubleshooting very difficult, if not impossible. When the results are not received, there is no easy way to tell where the problem exists.

In the previous example, it takes a long time to determine where the breakdown in communication occurred. Several different systems must be examined to answer questions such as:

- Did the HIS send the order?
- When did the HIS send the order?
- Did the lab get the order?
- When did the lab get the order?
- Did the lab send the results?
- When did the lab send the results?

Once the problem is located, then it takes additional work with various groups to correct the problem.

Estimating load on system

Without centralized monitoring there is no way to figure out the overall stress on the system or the number of messages going through the system at any one day or time of day. This situation makes accurately estimating the need for server size, network communications, and staff needed to support the connections very difficult, if not impossible.

Communications with an Interface Engine

Overview

An interface engine is designed to simplify connecting, maintaining, monitoring, and sharing data between interfaces. An interface engine can take data from a sending application and filter it or change the format of the data to match each individual application's needs.

This feature greatly reduces the number of individual endpoints required to communicate between applications which in turn, saves on the price of implementing of an integrated system.

Example of engine filtering and mapping

The following figure illustrates how an engine can filter and map data to match each individual application needs.

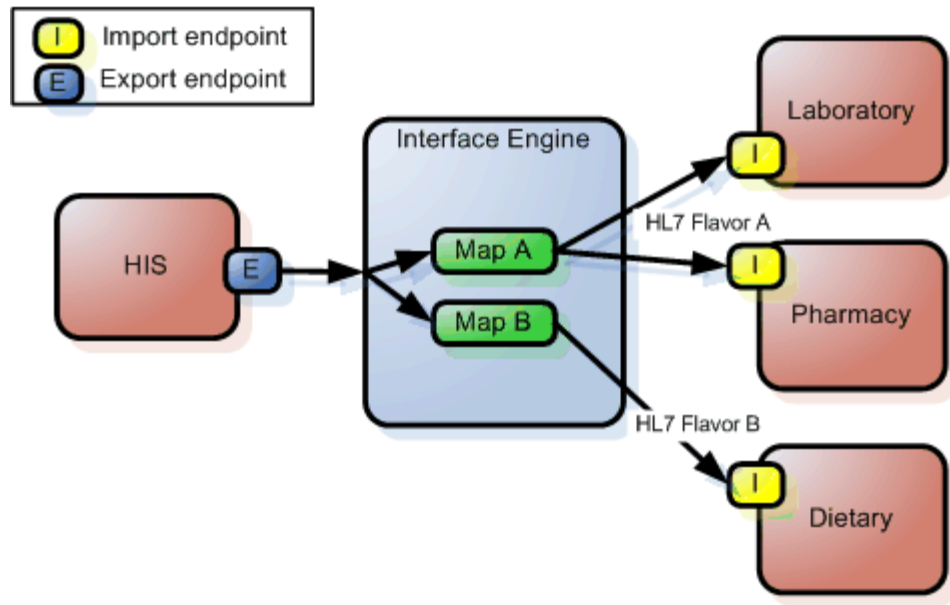


Figure 6: Interface Engine Leveraging Data

In the above figure the interface engine leveraged the data in the following manner:

- One message from the HIS
- Sent to three different applications without the need for three export endpoints on the HIS
- The engine filtered and mapped the data to meet the needs of the lab, pharmacy, and dietary applications

Explanation of filtering and mapping

In Figure 6, the HIS exports data into the interface engine. The interface engine takes the data and filters it for specific applications. Next, the interface engine puts the data into different formats by using a map to translate the received format into the format needed for the receiving application.

The laboratory and pharmacy both use the same data format or HL7 flavor so the data produced by Map A can be sent to both applications. The dietary application uses a different data format, HL7 Flavor B, so Map B is used to get data ready to send to Dietary.

Example interface engine illustration

The following illustration represents the same five interfaces shown in Figure 3 communicating with the benefit of an interface engine. Notice that the interface engine model requires considerably fewer endpoints (shown by yellow and blue boxes) than the point-to-point model.

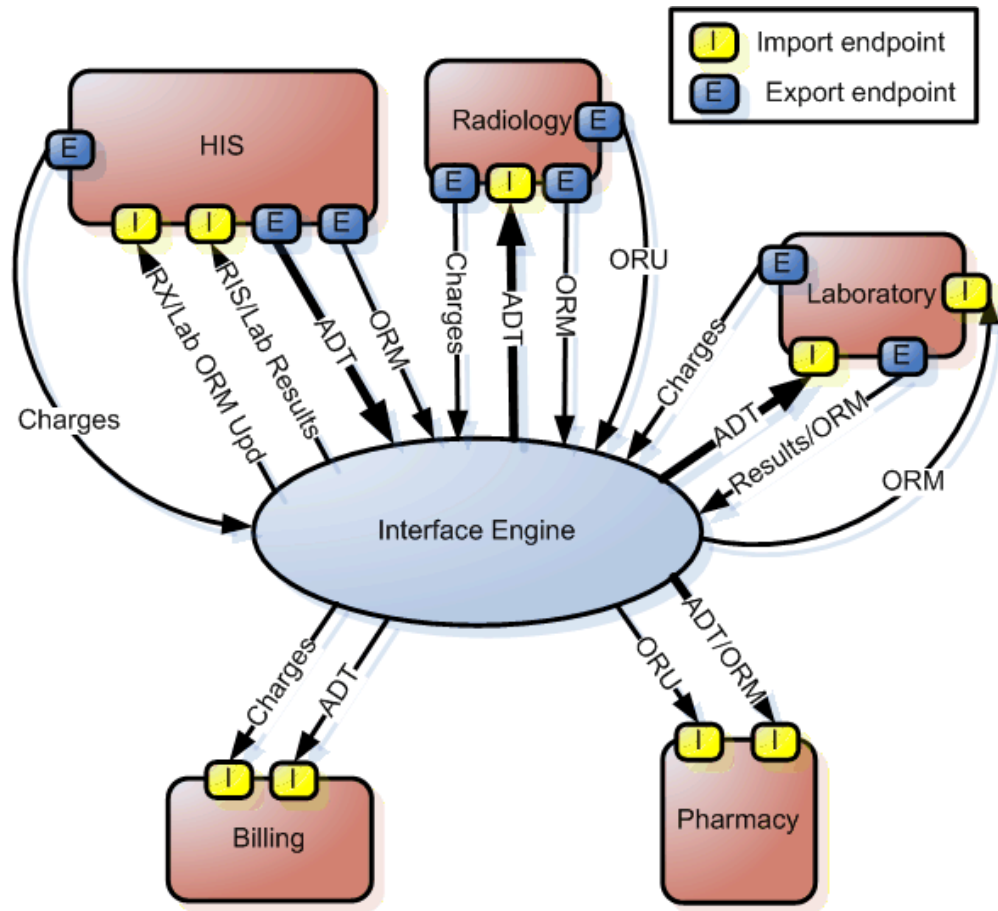


Figure 7: Interface Engine Model Requires 17 interfaces at approximate cost of \$170K (a reduction of \$90K over the point-to-point example provided in Figure 3)

Cost of implementing interface engine model

The costs to implement the previous interface engine example are:

- 17 endpoints at \$10K each = \$170K
- Time waiting for application vendors is greatly reduced due to leveraging of messages allowable by an interface engine

- Effort: If each endpoint takes three calendar months to complete, worst case is 50 calendar months. However, if all endpoints were built at once, there would be significant schedule overlap. A reasonable estimate is a total of six to nine calendar months.

Savings incurred

In a point-to-point approach as shown on page 7, this same model costs approximately \$260K and 24 calendar months to implement.

Using an interface engine saves \$90K and 15 to 18 months.

Replacing and Adding New Interfaces in an Engine Model

Overview

In an engine model, all messages flow through the engine so the process of adding or changing an interface is much simpler than in a point-to-point model.

Process to replace applications

The only endpoints that need to be created for the replaced application are the import and export endpoints to and from the new application. The interface engine insulates the other applications from many of the changes.

Example of replacing an application

The following figure illustrates replacing the laboratory application in the previous example along with the required endpoints.

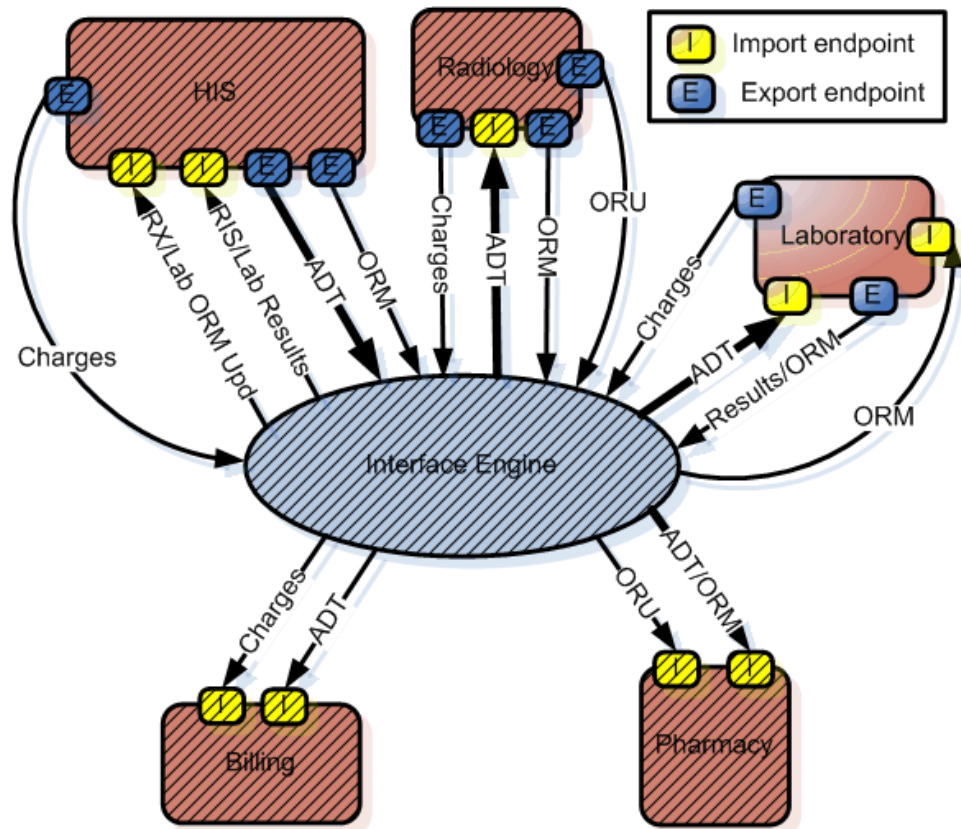


Figure 8: Replacing Interface Engine Model Requires 4 interfaces at approximate cost of \$40K

Costs of replacing application example

The costs to replace the application in the previous engine example are:

- 4 endpoints (not shaded) at \$10K each = \$40K
- Effort: Only the application vendor of the replaced application needs to create four endpoints. Approximately 4 months.

Savings incurred

In a point-to-point approach as shown on page 10, to replace this same interface would cost approximately \$125K and 15 calendar months to implement.

Using an interface engine saves \$85K and 11 months.

Process to add applications

With an interface engine, adding an application does not require as many endpoints to be created by the new application vendor and the existing application vendors. The interface engine can take the same data that it is already receiving and send it on to the new application.

Example of adding a new application

The following figure illustrates adding a new surgery application to the interface engine example which needs bidirectional communications with the HIS and one-way communication with the billing system and laboratory.

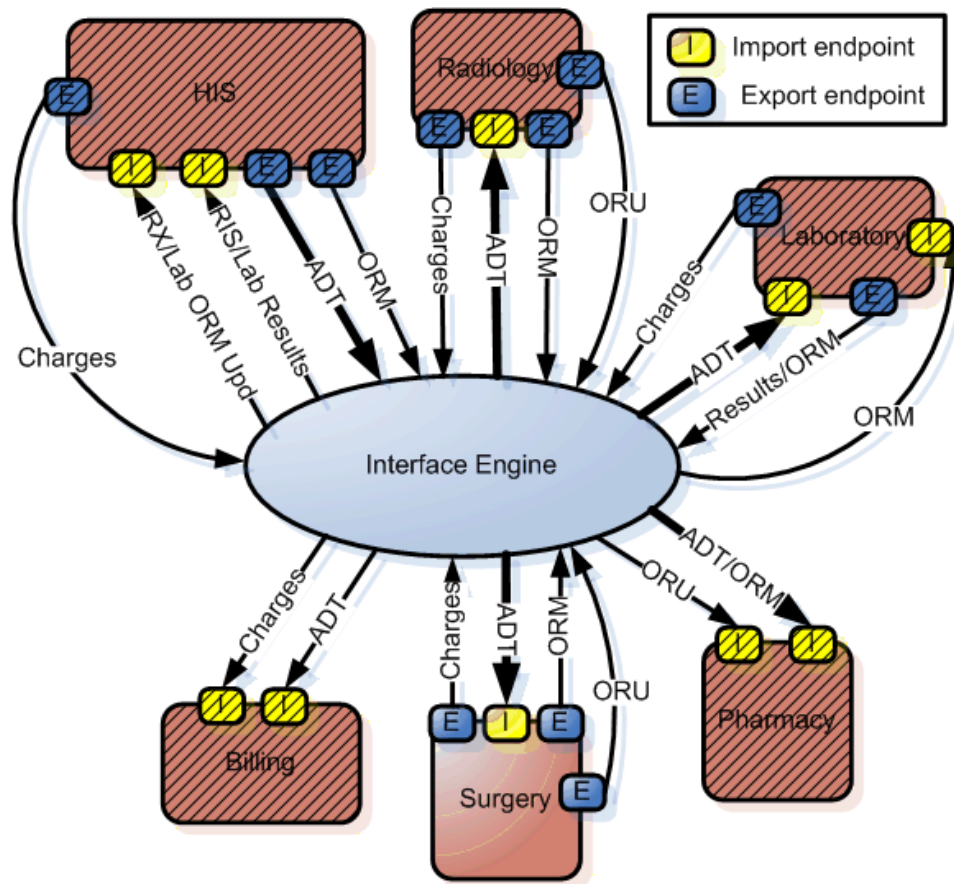


Figure 9: Adding Application in engine model Requires 4 interfaces at an approximate cost of \$40K

Cost of adding new application

The costs to add the surgery application in the previous interface engine example are:

- 4 endpoints (not shaded) at \$10K each = \$40K
- Effort: Only the application vendor of the new application needs to create four endpoints. Approximately 4 months.

Savings incurred

In a point-to-point approach as shown on page 11, adding this same application costs approximately \$125K and 15 calendar months to implement.

Using an interface engine saves \$85K and 11 months.

Monitoring Interfaces with an Interface Engine

Overview

An interface engine provides monitoring of all connections from a centralized location.

The monitoring capability provides the ability to:

- Quickly view the state of the connections
- Review details about the connections
- Create alerts when a problem arises

Quick responses

With an interface engine, when a connection stops it is immediately apparent. Being able to quickly view information about connections allows the interface team to immediately respond in order to resolve the problem. This approach ensures each doctor receives results in a timely manner.

Connection information

An interface engine also provides detailed information about each connection, such as the number of messages processed and errored, how long the connection has been idle, and how many messages are waiting processing for a connection.

This information is extremely valuable when looking at trends of a connection in order to:

- Catch problems quickly
- Estimate hardware and staffing needs to support a particular connection

Proactive alerting

In addition to being able to monitor all the connections, some interface engines provide the ability to create alerts when an issue occurs on a connection that may negatively impact the system.

In addition to being visible on the monitoring window, these alerts typically can be configured to send an e-mail to the appropriate party or parties to let them know of the situation even if they are not actively monitoring the system.

Configurable alerts

An interface engine should allow for the configuration of alerts that should be alerted on and not for events that do not need alerts.

This configuration should be dynamic by connection, day, and time of day because message activity varies due to each of these factors.

Example: The following table shows the message activity for a lab during a weekday. The lab is closed after 6:00 PM and on the weekends.

Time	Approximate Number of Messages
7:30 AM – 10:00 AM	500
10:00 AM – 12:00 PM	100
12:00 PM – 2:00 PM	300

2:00 PM – 4:00 PM	100
4:00 PM – 6:00 PM	300

In this example, if the lab has not received a message for 15 minutes between 7:30 AM and 6:00 PM, an idle alert should be created. The fact that the system has been idle for this long during this period of time could indicate a problem with an external healthcare provider's system or an internal application.

However, if the connection was idle for 15 minutes between 6:00 PM and 7:30 AM, no alert should be created because the lab is not open and would not be expecting a large volume of messages. Additionally, if the system is idle for 15 minutes on the weekend, no alerts should be created since the lab is not open on the weekends.

Logging with an Interface Engine

Overview

An interface engine logs connection activity including all the sent and received messages, connection states, errors, and alerts.

Without logging or archiving of messages, there is no way to tell what happened to a message once it was sent from an application. Unfortunately, in an interfaced environment, communication problems occur from time to time and logging is the only way to determine the root of the problem and correct it.

Benefit of logging

The logging of an interface engine provides the ability to:

- Track the flow of a message throughout the system
- Know exactly when a message was sent, received and acknowledged, and what information the message contained
- Easily troubleshoot communication problems
- Resend messages if they are lost

About NeoTool

NeoTool is a leading provider of healthcare integration solutions that empower organizations to develop, test, deploy, and manage data exchanges between healthcare applications and providers. Through software, HL7 training, and consulting, NeoTool is dedicated solely to healthcare application interfacing. NeoTool customers include healthcare providers (e.g., hospitals, imaging centers, labs, and clinics), healthcare software application providers, and medical device manufacturers. www.neotool.com

NeoTool
6509 Windcrest Drive
Suite 160B
Plano, Texas 75024
469-229-5000
Sales@neotool.com

