# HL7 Gateway Programming Exercise

This exercise is designed to let the candidate showcase their skills with both JavaScript and Mirth development, as well as knowledge of HL7.

The exercise involves developing two distinct components that collectively form a rudimentary HL7 gateway solution.

Each component should be designed as two channels within Mirth.

* Component #1 is an HL7 producer Mirth channel.
* Component #2 is an HL7 consumer Mirth channel.

Details for both components follow below.

Several criteria are required for a successful submission of these components:

* The candidate must make both Mirth channels available for review via a publicly accessible repository (e.g., GitHub).

[RAY] The Mirth channels have been posted to GitHub and are available in the following GitHub directory:

* <https://github.com/Ray-Walker/Availity>
* The candidate must be prepared to demonstrate and discuss the two components working together in a live demonstration via screenshare.
* Each component must be the candidate’s own work.  It is acceptable to use open-source libraries and frameworks but the principal functionality of each component must be original work.
* The candidate should be mindful that the goal of this exercise is to demonstrate not only technical fluency, but also their level of professionalism as a software engineer.  The candidate should expect that their work will be assessed with the same rigor as a pull request.

## Component #1 – HL7 Message Producer

This component should ingest a delimited file and create HL7 messages from each row of data. A minimum of 5 HL7 messages should be produced. The messages can be rudimentary in terms of HL7 structure and fictionalized data: it must have a message header, patient demographics, and visit information. It may include any additional segments that the candidate chooses to use to highlight the functionality described below for Component #2.

Component #1 should then send these HL7 messages along with the channel name to Component #2. The channel name of Component #1 should be available to retrieve in Component #2. The method at which to send them will be up to the candidate. As always, creativity is a plus!

The messages that are produced should have sufficient variability to allow Mirth channel features to be employed including filtering, transforming, and dealing with successful and errored message sources.

## Component #2 – HL7 Consumer

This component consumes the HL7 messages written by Component #1 and must be implemented as a Mirth channel.  (The candidate must download and install the freely available NextGen Connect Integration Engine to complete this task.)

The channel must demonstrate that it uses the correct mechanism as the message source, with message filtering and transformation applied to that source.  The channel must have at least one destination as well.  The details of the destination(s) are up to the candidate, but one suggestion would be to write to a custom file.  More interesting destinations are always appreciated, creativity is a plus! The only requirement is to display the channel name of Component #1 is some aspect when sending the files through the destination.

[RAY]  The ‘Component2 - HL7 Consumer’ Channel includes 3 outbound Destination Connectors, as follows:

* File Writer - HL7 to SFTP Directory
  + Sends HL7 file to Cloud SFTP Directory
  + Enabled by default
* File Writer - HL7 to Windows Directory (Disabled by default)
  + Sends HL7 file to Windows directory on Local Machine hosting Mirth Connect Server
  + Disabled by default. Enable if Mirth Connect Server hosted by Windows
* File Writer - HL7 to Mirth Appliance Directory (Disabled by default)
  + Sends HL7 file to Mirth Appliance Directory.
  + Disabled by default. Enable if Mirth Connect Server hosted by a Mirth Appliance

To obtain outbound HL7 files from the cloud SFTP Directory, perform the following:

1. Login to cloud SFTP Directory

URL : <https://app.sftpcloud.io/login>

1. Enter the following username and password

USER : [raymondhwalker@outlook.com](mailto:raymondhwalker@outlook.com)

PASS : Surfsup01

1. Select availity-instance from the SFTP instances page that appears
2. Select the File manager tab from the page that appears. (See image below)
3. Select a file to download and review

A screenshot of a computer

Description automatically generated

## HL7 Questionnaire

1. What is an ADT message?

[RAY] The ADT Message stands for Admission, Discharge and Transfer and is the Patient Administration transaction set that provides for the transmission of new or updated demographic and visit information about patients. ADT messages are a type of electronic notification that communicate patient information across the healthcare community. An ADT message is written in HL7 ASCII or UTF-8 character sets depending on the HL7 version and/or preference.

2. What are HL7 Separator characters?

[RAY] The only true HL7 Separator is the pipe character,(|). However, there also HL7 Encoding characters that are sometimes referred to as HL7 Separator characters as well. Those are:

* ^ - Component Separator: Usually represented by the hat character (^)
* ~ - Field Repeat Separator: Usually represented by the tilde character (~)
* \ - Escape Character: Usually represented by the backslash character (\)
* & - Subcomponent Separator: Usually represented by the ampersand character (&)

3. What function would you use to update the date and time to current in MSH segment in Mirth?

[RAY] Mirth Connect includes a DateUtil class (function) with a getCurrentDate method. Here is one way to update MSH.7.1 (Date/Time of Message):

// Logger Debug Switch

var debug = true;

// Current DateTime

var dateTime = DateUtil.getCurrentDate('YYYYMMddHHmmsssss');

if (debug === true) { logger.info(channelName + ' - dateTime: ' + dateTime) };

channelMap.put('dateTime', dateTime);

4. A sending application sends all ADT types. The client only accepts ADT-A01. How would you handle that in Mirth?

[RAY] Mirth Connect includes a Source Filter which determines whether a message proceeds through the channel depending on the filter criteria. Source filtering is the 3rd step in the message workflow. The steps are:

* Batch processing, if enabled
* Preprocessing
* Source Filter
* Source Transformer
* Destination Filter
* PostProcessing

Exercise Note: In the CSV Source Batch file, the ADT Trigger event is not passed, (as most CSV’s would not know about Trigger Events). Instead, the first column passed is ‘EVENT’ which describes what type of ADT message to use. The event value is transformed to the Trigger Event in MSH.9.2 (Event).

Here is an example using the Source Filter to accept ADT-01 message types, rejecting all other message types.

if (msg['row']['column1'].toString() != 'admit') {

return true;

};

return false;

5. Where in Mirth would you write code that is used in multiple channels?

[RAY] Code Templates. (Mirth Connect \ Channel Tasks \ Edit Code Templates)

6. Where would you set a certain type of data for a channel to receive and send out?

[RAY] Mirth Connect \ Channel \ Summary Tab \ Data Types \ Click on ‘Set Data Types’

7. How would you allow the code, written in question 5, to be used by a channel?

[RAY] A Code Template (CT) is used in a channel by calling the CT as a function(). The function then returns the transformed value of the function. For example:

* var CDateTime = cDateTime(dateTimeOfMessage);

8. Inbound and outbound message templates are used for what?

[RAY] Message templates are used to interpret Inbound and outbound messages based on their structure and data type and act as a guide for the connector to process the incoming data.

For example, if the incoming message is an HL7 document, the inbound template will define the structure of the HL7 segments. Message Templates also help you create message mappings and transforms "by example".

The inbound template represents the incoming message. When you paste in a sample message, and drag the segments over from the tree to create mapping steps, you're essentially telling Mirth Connect that you are expecting to receive a message that looks like the inbound template (that is, has the same segments). Since the inbound template is just an "example", none of the data from the template is used, just the mappings that were created. You would use an inbound template when you want to extract segments or modify an incoming message.

The outbound template is used to specify the outgoing message format. By providing an outbound template, you are telling Mirth Connect that you want your outgoing message to look different from your incoming message. In this case, the content of the outbound template is exactly what is going to be sent, so you'll want to make sure that you clear out any data that you wouldn't want to be sent with every message. Once you've pasted in an outbound template, you can use the Message Builder step type to modify the message however you want. The most common way to do this is to paste in both an inbound and an outbound template, and then drag-and-drop from a segment in the inbound to a segment in the outbound. This will create a new step that takes the data from the incoming message and inserts it into the outgoing message.

Below is a visual example on the use of both Inbound and Outbound messages for this exercise:

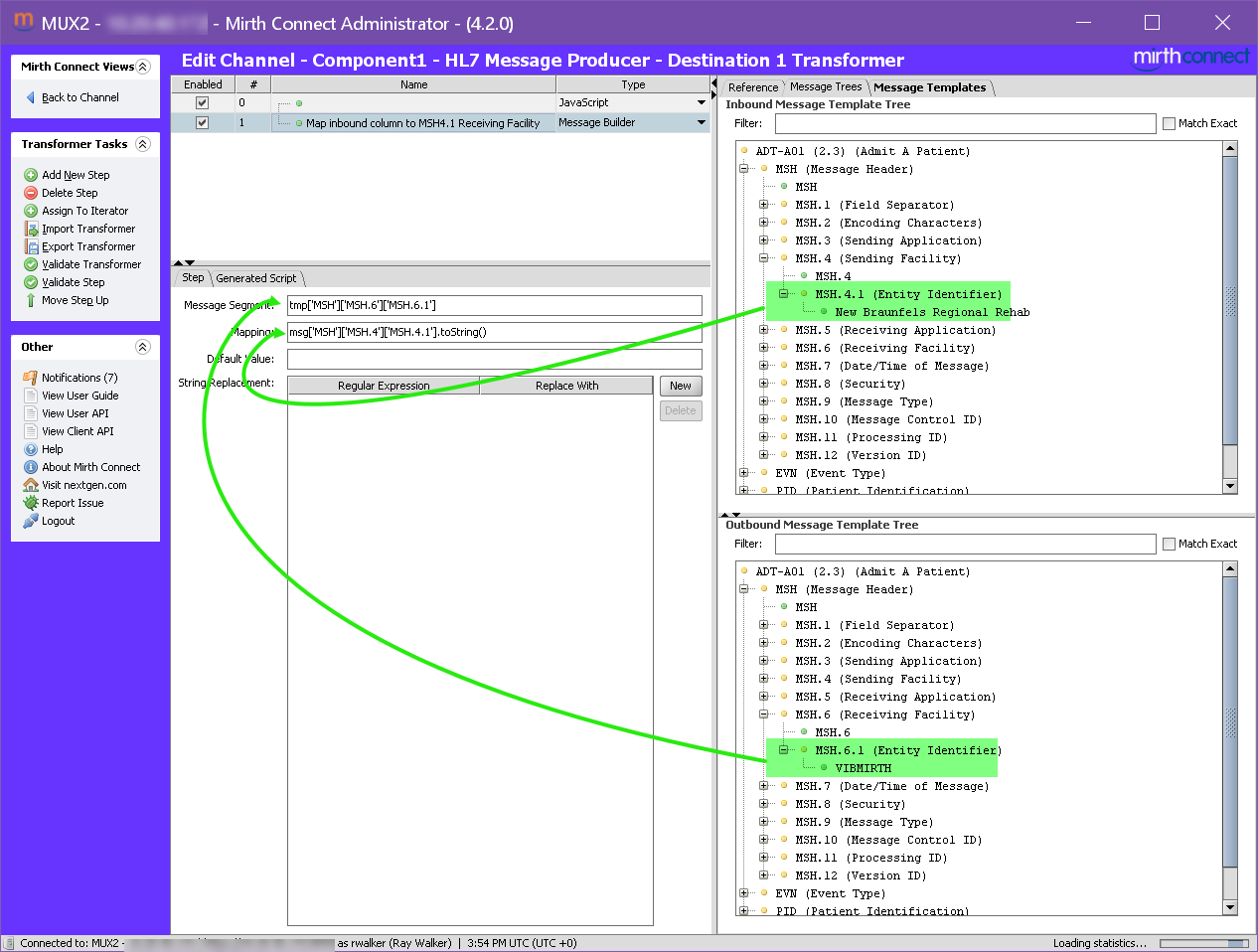
* Each column in the Source CSV Batch file is mapped to a variable
* Each HL7 Segment SubComponent is then mapped from the variable

A screenshot of a computer

Description automatically generated

9. How would you map a variable without the use of JavaScript?

[RAY] A great lower TCO and ‘getting started’ feature of Mirth Connect is the ability to process messages without necessarily having to understand JavaScript and other languages. Using the ‘Message Builder’ step type in a source and/or destination transformer, an individual can simply ‘drag and drop’ object representations of the inbound Source msg template (msg) to the Outbound Destination msg template (tmp). The data can also be manipulated using regEx and a Default Value can be defined with the ‘Message Builder’ GUI.



10. Using JavaScript, loop through any 3 element array and log/write out the individual elements.

[RAY] I’ve created the code below and can be found in the Source Transformer Step2 of the Component1 - HL7 Producer channel.

A screenshot of a computer

Description automatically generated

Mirth Connect Dashboard Log showing the individual array elements being logged

A screenshot of a computer

Description automatically generated

Mirth Connect Channel Dashboard showing the individual array elements being mapped

A screenshot of a computer

Description automatically generated

11. What is the Velocity Template Language? Can you provide an example?