# 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).
* 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.

## HL7 Questionnaire

1. What is an ADT message?

2. What are HL7 Separator characters?

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

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

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

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

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

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

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

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

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