HL7 Gateway Programming Exercise

The files for this project reside online at the following link:

> https://www.mediafire.com/folder/27naidd77scsq/Availity

In that link is the Availity.zip file. You will find the following:

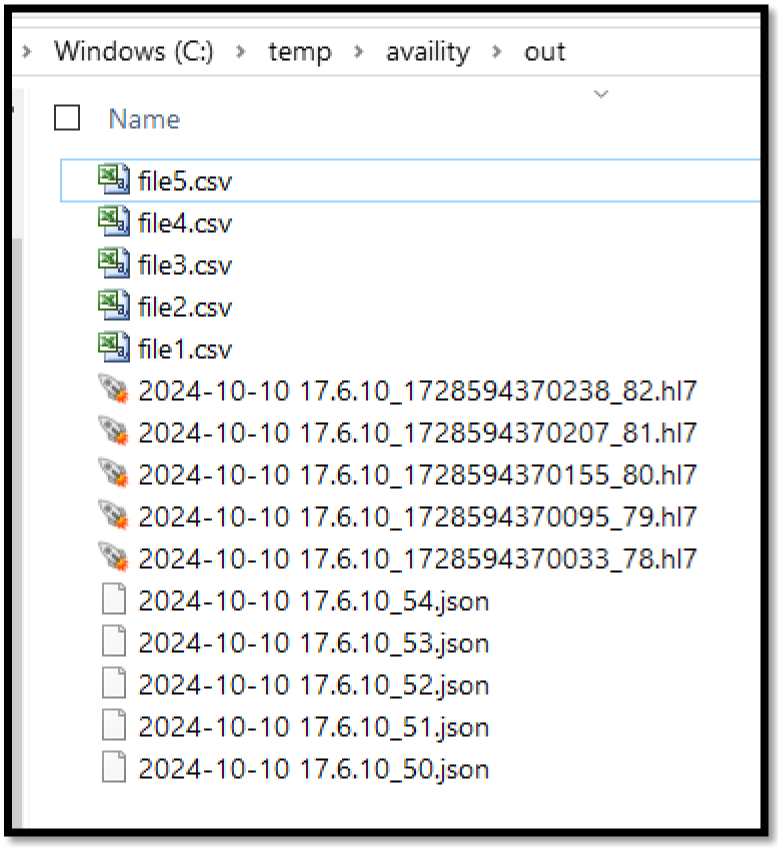
- Availity Homework – Mirth HL7.pdf
  - Answers to questions and project description
- 5csv.zip
  - the csv files that Component #1 will ingest.
- HL7 Consumer Mirth v4_5_1.xml
- HL7 Producer Mirth v4_5_1.xml

The project consists of 2 Channels:

> **Channel 1: HL7 Producer** - Component #1. This channel will ingest 5 delimited files from the "in" folder, and create 1 HL7 message from each row of data. The result will be 5 HL7 files written to the "out" folder. Messages will have a message header (MSH), patient demographics (PID), and visit information (PV1). It will also include an additional NK1 segment. This channel will then send both the HL7 and the channel name to Channel #2 named "HL7 Consumer".
>
> **Channel 2: HL7 Consumer** - Component #2. This channel receives the HL7 messages from Component #1. It ensures only messages from the Component #1 (HL7 Producer) are processed via Mirth filter. It then retrieves the Component #1 channel name and patient demographics and creates a json file - one file per patient.

The result of this project will be found in the *out* folder:

- 5 csv files - original 5 files
- 5 hl7 files - generated by Component #1
- 5 json files - generated by Component #2.



Windows (C:) › temp › availity › out

Name

```
file5.csv
file4.csv
file3.csv
file2.csv
file1.csv
2024-10-10 17.6.10_1728594370238_82.hl7
2024-10-10 17.6.10_1728594370207_81.hl7
2024-10-10 17.6.10_1728594370155_80.hl7
2024-10-10 17.6.10_1728594370095_79.hl7
2024-10-10 17.6.10_1728594370033_78.hl7
2024-10-10 17.6.10_54.json
2024-10-10 17.6.10_53.json
2024-10-10 17.6.10_52.json
2024-10-10 17.6.10_51.json
2024-10-10 17.6.10_50.json
```

HL7 Questionnaire

1. What is an ADT message?

An ADT Message is an Admission, Discharge, and Transfer HL7v2 message for electronic communications in a healthcare setting. It consists of information about a patient's hospital admission, discharge, and transfer.

For me, I can easily identify an ADT message by the MSH-9 field. For example, below I can see the message is an ADT A04 message, which is a Patient Registration message.

```
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|||ADT^A04|103102|P|2.4||
```

2. What are HL7 Separator characters?

Separators are special characters that help ensure that the data in the message is organized and can be processed correctly by the receiving system.

The separators below are defined in MSH-1 (field separator) and MSH-2 (encoding characters):

```
MSH|^~\&
```

- **Field Separator (|):** This character separates individual fields within an HL7v2 message segment. See highlighted field separators below:

```
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|||ADT^A04|103102|P|2.4
```

- **Component Separator (^):** This character separates components within a field. See highlighted component separator below:

```
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|||ADT^A04|103102|P|2.4||
```

- **Subcomponent Separator (&)**: This character separates subcomponents within a component. This is used for more granular data within a component. See PV1-7.9 (assigning authority):

```
PV1|1|R|||||Manning^Manning^Terry^^^^^^7654321&UPIN||||||||N||A
```

- **Repeating Field Separator (~):** This character indicates that a field can repeat.

```
PID|1||PATID1234^5^M11^ADT1^MR^GOOD HEALTH
HOSPITAL~123456789^^^USSSA^SS||
```

- **Escape Character (\):** This character is used to escape the separator characters if they need to be included in the data. In the example below, if a value is AT&T, the value escaped would be AT\T\T.

```
PID|1||PATID1234^5^M11^ADT1^MR^AT&T HOSPITAL~123456789^^^USSSA^SS||
```

PID|1||PATID1234^5^M11^ADT1^MR^AT\T\T HOSPITAL~123456789^^^USSSA^SS||

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

The Message date and time is in the MSH header and resides in the MSH-7 field.

MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|20241008000000||ADT^A04 |103102|P|2.4||

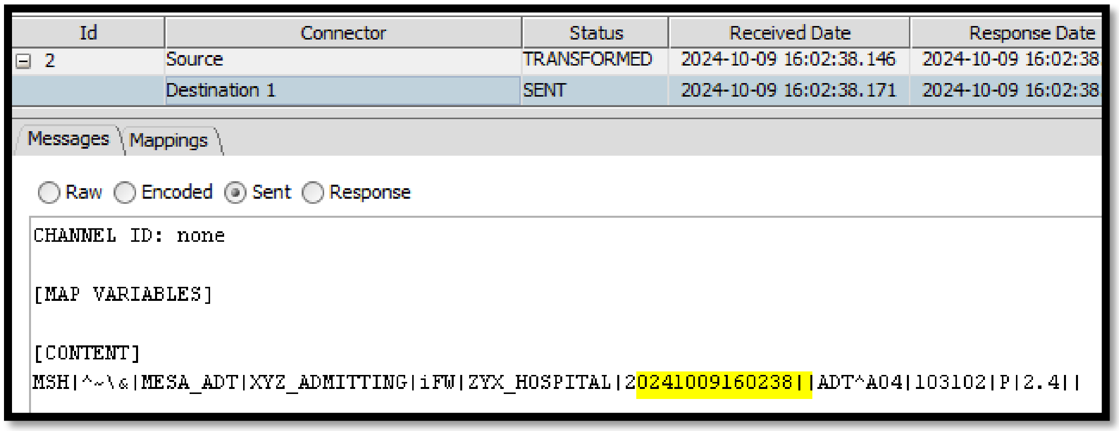
In the MSH-7 field above, the date is October 8, 2024 with 00 hours, 00 mins, and 00 seconds.

To update MSH-7 to current date and time to today's date/time (10/9/24), I would use:

msg['MSH']['MSH.7'] = DateUtil.getCurrentDate("yyyyMMddHHmmss");

Then Mirth would change it to 10/9/24 @ 4:02pm with 38 seconds:

MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|20241009160238||ADT^A04 |103102|P|2.4||



Id	Connector	Status	Received Date	Response Date
2	Source	TRANSFORMED	2024-10-09 16:02:38.146	2024-10-09 16:02:38
	Destination 1	SENT	2024-10-09 16:02:38.171	2024-10-09 16:02:38

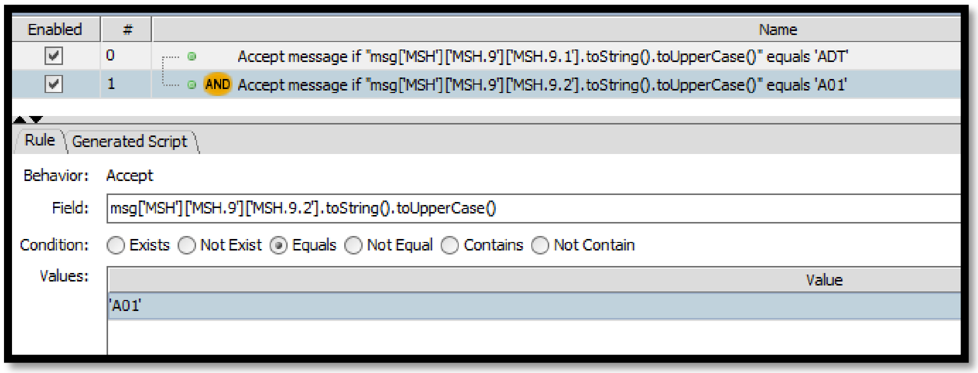
```
CHANNEL ID: none

[MAP VARIABLES]

[CONTENT]
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|20241009160238||ADT^A04|103102|P|2.4||
```

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

I would use a SOURCE FILTER and ensure the MSH-9.1 is "ADT" and MSH-9.2 is "A01":



Enabled	#	Name
✓	0	Accept message if "msg['MSH']['MSH.9']['MSH.9.1'].toString().toUpperCase()" equals 'ADT'
✓	1	AND Accept message if "msg['MSH']['MSH.9']['MSH.9.2'].toString().toUpperCase()" equals 'A01'

Rule | Generated Script

Behavior: Accept

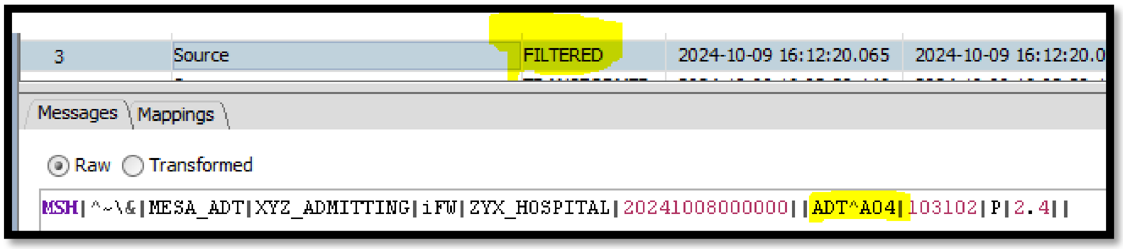
Field: msg['MSH']['MSH.9']['MSH.9.2'].toString().toUpperCase()

Condition: Exists Not Exist [✓ Equals] Not Equal Contains Not Contain

Values:

Value
'A01'

The result when I send an ADT-A04 = Mirth filters message:



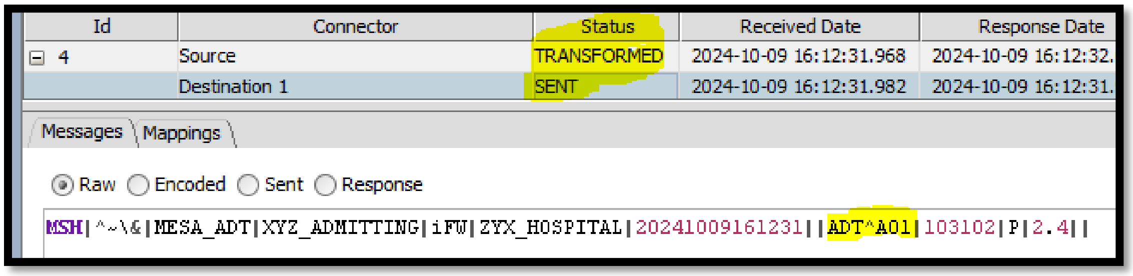
3	Source	FILTERED	2024-10-09 16:12:20.065	2024-10-09 16:12:20.0

Messages | Mappings

[✓ Raw] Transformed

```
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|20241008000000||ADT^A04|103102|P|2.4||
```

The result when I send an ADT-A01 = Mirth accepts message:



Id	Connector	Status	Received Date	Response Date
4	Source	TRANSFORMED	2024-10-09 16:12:31.968	2024-10-09 16:12:32.
	Destination 1	SENT	2024-10-09 16:12:31.982	2024-10-09 16:12:31.

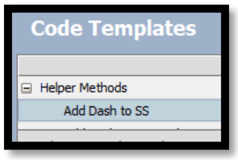
Messages | Mappings

[✓ Raw] Encoded Sent Response

```
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|20241009161231||ADT^A01|103102|P|2.4||
```

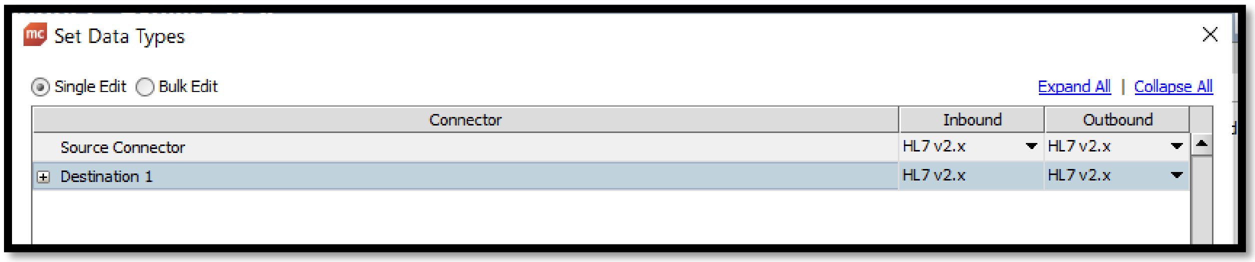
5. Where in Mirth would you write code that is used in multiple channels?

I would write it in a code template, which can be found in the CHANNELS > EDIT CODE TEMPLATES section.



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

In the channel's summary tab under SET DATA TYPES:

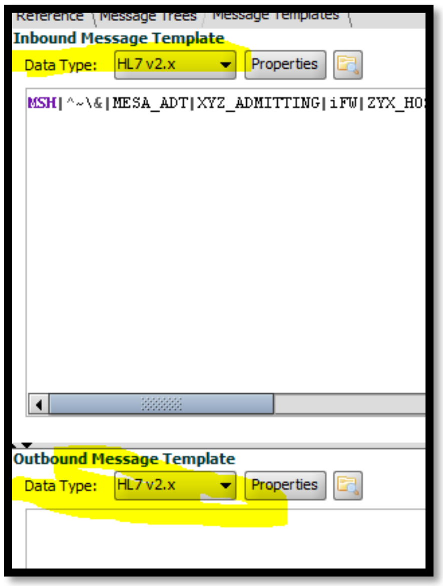

Set Data Types

Single Edit [✓] Bulk Edit

Connector	Inbound	Outbound
Source Connector	HL7 v2.x	HL7 v2.x
Destination 1	HL7 v2.x	HL7 v2.x

In the example above, the Mirth channel will only receive HL7v2 data type and send out HL7v2 data type.

You can also change it in the Message Template sections:


Inbound Message Template

Data Type: HL7 v2.x

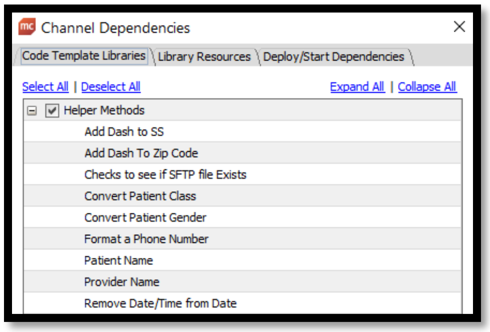
```
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HO
```

Outbound Message Template

Data Type: HL7 v2.x

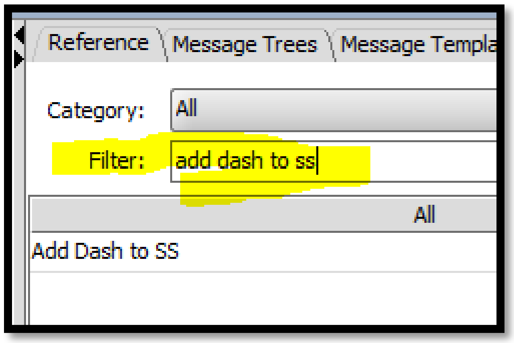
7. How would you allow the code, written in question 5, to be used by a channel?

First, we must set the dependency. You can find that in the SUMMARY tab, click the SET DEPENDENCIES button. From there, you check the box of the code template library you would like to use. In the image below, if I want to use "Add Dash to SS", I have to click HELPER METHODS check box.


Channel Dependencies
Code Template Libraries
Library Resources
Deploy/Start Dependencies
Select All | Deselect All
Expand All | Collapse All

[✓] Helper Methods
- Add Dash to SS
- Add Dash To Zip Code
- Checks to see if SFTP file Exists
- Convert Patient Class
- Convert Patient Gender
- Format a Phone Number
- Patient Name
- Provider Name
- Remove Date/Time from Date

Then in the transformer section, I can search (or filter) that function...



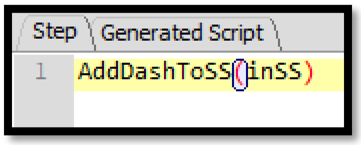
Category: All

Filter: add dash to ss

All

Add Dash to SS

and use it in my channel's transformer, by dragging and dropping:



```
AddDashToSS(inSS)
```

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

Inbound Message Templates = **msg** variable

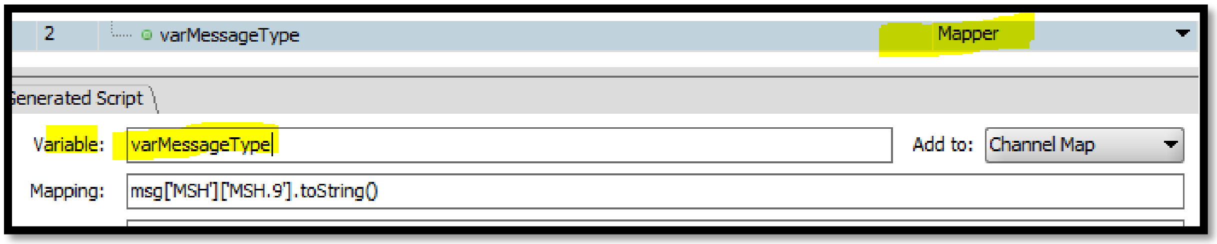
Purpose: These templates define how incoming messages are processed and interpreted.

Outbound Message Templates = **tmp** variable

Purpose: These templates define how messages are formatted and sent out after processing.

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

By using Mapper. In the image below, I created a variable named "varMessageType" for use in the Channel Map. This variable is then accessible in the Source and Destination transformers.



```
2    varMessageType                                   Mapper
Generated Script
Variable: varMessageType                      Add to: Channel Map
Mapping:  msg['MSH']['MSH.9'].toString()
```

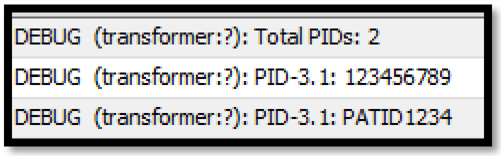
10. Using JavaScript, loop through any 3 element array and log/write out the individual elements.

Here is the PID Segment to loop through:

PID|1||PATID1234^5^M11^ADT1^MR^GOOD HEALTH HOSPITAL~123456789^^^USSSA^SS||EVERYMAN^ADAM^A^III||19610615|M||C|2222 HOME STREET^^GREENSBORO^NC^27401-1020|GL|(555) 555-2004|(555)555-2004||S||PATID12345001^2^M10^ADT1^AN^A|444333333|987654^NC|

```
var counter = 0;
//loop through PIDs
for each (pid in msg['PID']['PID.3'])
{
logger.debug("PID-3.1: " + pid['PID.3.1']);
counter++;
}
logger.debug("Total PIDs: " + counter);
```

Output:



```
DEBUG  (transformer:?): Total PIDs: 2
DEBUG  (transformer:?): PID-3.1: 123456789
DEBUG  (transformer:?): PID-3.1: PATID1234
```

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

I have never used nor heard of VTL, but I looked it up. After reading Apache websites below, I instantly saw that this is similar to the Apache Struts Framework I used back in 2003 with Java system back when I was a Jr. Programmer for Anteon, which is now General Dyanmics. We were web-enabling a Departmenet of Defense meteorological system written in Java OOP.

https://velocity.apache.org/engine/1.7/user-guide.html

https://velocity.apache.org/engine/1.7/vtl-reference.html

VTL is a scripting languge that is used for creating content for web applications. It’s part of the Apache VTL project often ussed to generate HTML, XML, or other text formats by merging templates with data provided by a backend system. It’s used with Java-based web framworks, making it a popular choice for Java developers who want to implement templating solutions.

Here’s a simple example of a VTL template for generating a greeting message:

```
#set($userName = "Alice")
#if($userName)
  Hello, $userName! Welcome to our site.
#else
  Hello, Guest! Please log in.
#end
```

Output:

```
Hello, Alice! Welcome to our site.
```