Module 5: Exercise 1

Setting up your environment

Objective: This exercise aims to set up your environment for future exercises.

Note: You may need to refer back to these instructions as you start other exercises or if you stop partway through an exercise.

To start, please make sure you have the following extensions installed in Visual Studio Code: * InterSystems ObjectScript * InterSystems Server Manager * InterSystems Language Server * InterSystems ObjectScript Extension Pack

Download the latest version of this repository (or unzip onto your local machine if provided a zip file). Open the unzipped folder from Visual Studio Code.

Next you'll need to build and start your docker container. Make sure your Docker desktop application is running. Once that is confirmed, ensure you don't have any older versions of this code running in Docker. If so, stop those containers and delete them if not needed.

Open a terminal prompt from the root folder of the codebase and navigate to the **\iris-container** folder and run the following command.

```
docker-compose up --build -d
```

This command will take a few moments to run.

Once your container is running you can now access the IRIS portal at the following url:

```
http://localhost:32783/csp/sys/%25CSP.Portal.Home.zen?$NAMESPACE=%25SYS
```

Starting the Production

Now we just need to start the production before we can move forward. Click on **Interoperability** and navigate to the **FHIRDEMO** namespace if not already selected. Click **Configure** then **Production** and finally **Go**.

Inside the Production Configuration screen, you just need to click **START** (you'll be adding production components later).

Opening a Terminal in VSCode

1. Open the InterSystems Terminal from VS Code. First make sure the Docker container is running. Open a Terminal by going to **View Menu -> Terminal**

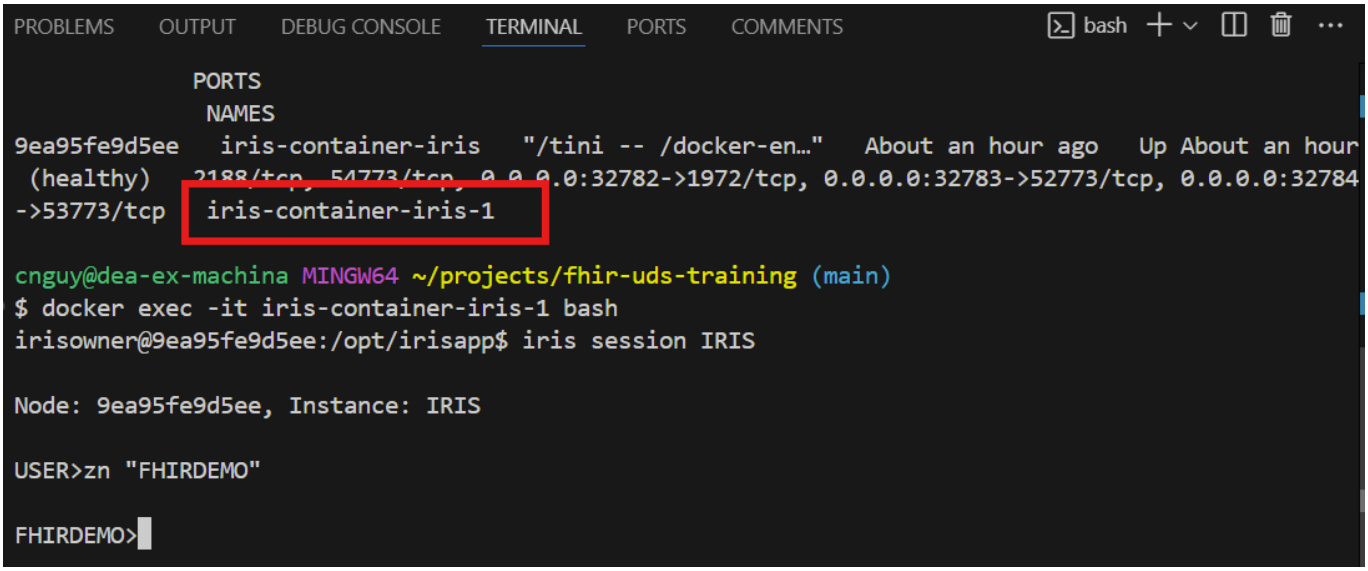
2. This opens the Terminal window at the bottom of the project. (It may already be open).

- Find the `+` sign with the pull down.
- Select `Git Bash` as the terminal type.
- Type `docker ps` to figure out the name of the container. In this case `iris-container-1`
- Once you have the name, type this command to start the shell. You will be looking at the internal Docker file system:

```
docker exec -it **iris container name** bash
```

- From the new command prompt, open IRIS terminal by typing: `iris session IRIS`
- Log in with the `_system/SYS` user/password

Here is a screenshot showing the commands:



Save the above command somewhere handy for later reference.

Module 5: Exercise 2

Loading and Navigating SDA

To start this exercise you must first complete Module 5 Exercise 1.

Objective: The goal of this exercise is to learn about InterSystems SDA.

First, navigate to the XML Schemas screen of your IRIS instance by clicking on Interoperability -> Interoperate -> XML -> XML Schema Structures. You can also get there by clicking the following link:

```
http://localhost:32783/csp/healthshare/fhirdemo/EnsPortal.EDI.XML.SchemaMain.zen?
$NAMESPACE=FHIRDEMO&$NAMESPACE=FHIRDEMO&
```

Now, click on the Import button and select the HS.SDA3.xsd file located under the \irisdata\ folder. If you don't see the file, ensure that your "File of Type" dropdown is set to see XSD files. Complete the import.

Now you should see all of the sections/structures contained within the SDA XML format. Any documentation you may run across that relates to "Viewer cache or VIEWERLIB" can generally be ignored as this relates to Clinical Viewer in a HealthShare UCR environment only.

Tasks:

1. Search for and open the link for Container. How is this similar and different from a FHIR bundle?
2. Click on the Procedures() on Row 15. Review the details for ProcedureTime and indicate below what HL7 field this data comes from.
3. Scroll to the EncounterNumber field on Row 4. What happens when PV1-19.1 is null?
4. What is the EncounterNumber when PV1-19.1 is populated and FromTime is also populated?
5. Head back to the main page for Container and indicate which section of SDA will contain the data coming in from DG1?
6. Go another level up to the main SDA schema page and dig into three more SDA structures. Scan the documentation to get a feel for some of the constraints, data types and other details that involve mapping data into SDA3.

Module 5: Exercise 3

Navigating the FHIR documentation

To start this exercise you must first complete Module 5 Exercise 1.

Objective: The goal of this exercise is to learn how to access InterSystems FHIR Annotations.

First, navigate to the Home screen of your IRIS instance. You can do this by clicking the Home button on the top bar or just click on the following link:

```
http://localhost:32783/csp/sys/%25CSP.Portal.Home.zen?$NAMESPACE=%25SYS
```

Now, navigate to the **Health** tab from the Home screen.

To the right of the word **Foundation,** click on the FHIRDEMO namespace.

On the left navigation bar, click **Schema Documentation** and then **FHIR Annotations.**

On this screen, you can access all of the InterSystems FHIR documentation. Note the three different drop-downs you have access to. As you use these you will notice that only one drop down can be active at one time. However, the first two dropdowns work together and are specific to FHIR to SDA3 mapping information while the third dropdown is for SDA3 to FHIR mapping information. You'll also notice that the each dropdown may include FHIR resource types, data types AND other structural element types. Keep that in mind as you use this page.

Tasks:

1. Find Medication using the **FHIR4** dropdown.
2. Find Patient using the **Category** dropdown.
3. Find ObservationGroup using the **HS.SDA3** dropdown. Note the differences and similarities in the three drop downs.
4. Find the mapping from the Medication section of SDA3 to FHIR. Use whatever dropdown you deem appropriate.
5. Find the Quantity.system **SDA3 Target** field using whatever dropdown you deem appropriate.
6. Find SearchParameter:multipleAnd **FHIR4 Data Type** using whatever dropdown you deem appropriate.

Module 6: Exercise 1

Creating an HL7 to FHIR Integration

Learning Objectives:

- Build an end-to-end HL7 to FHIR pipeline
- Learn about standard IRIS FHIR processes, operations, and transformations
- Configure the IRIS FHIR Server
- Read a FHIR message trace
- Validate a FHIR Bundle using FHIR Vaildator

Start the Docker Container

1. Start the docker instance by right clicking on the `docker-compose.yml` file located in the `iris-container` folder. Click on the file name and select `Compose Restart` to stop and start the Docker container.

This can also be done from the terminal with the following command:

```
docker-compose up --build -d
```

This command will take a few moments to run.

2. Once your container is running you can now access the IRIS portal at the following url:

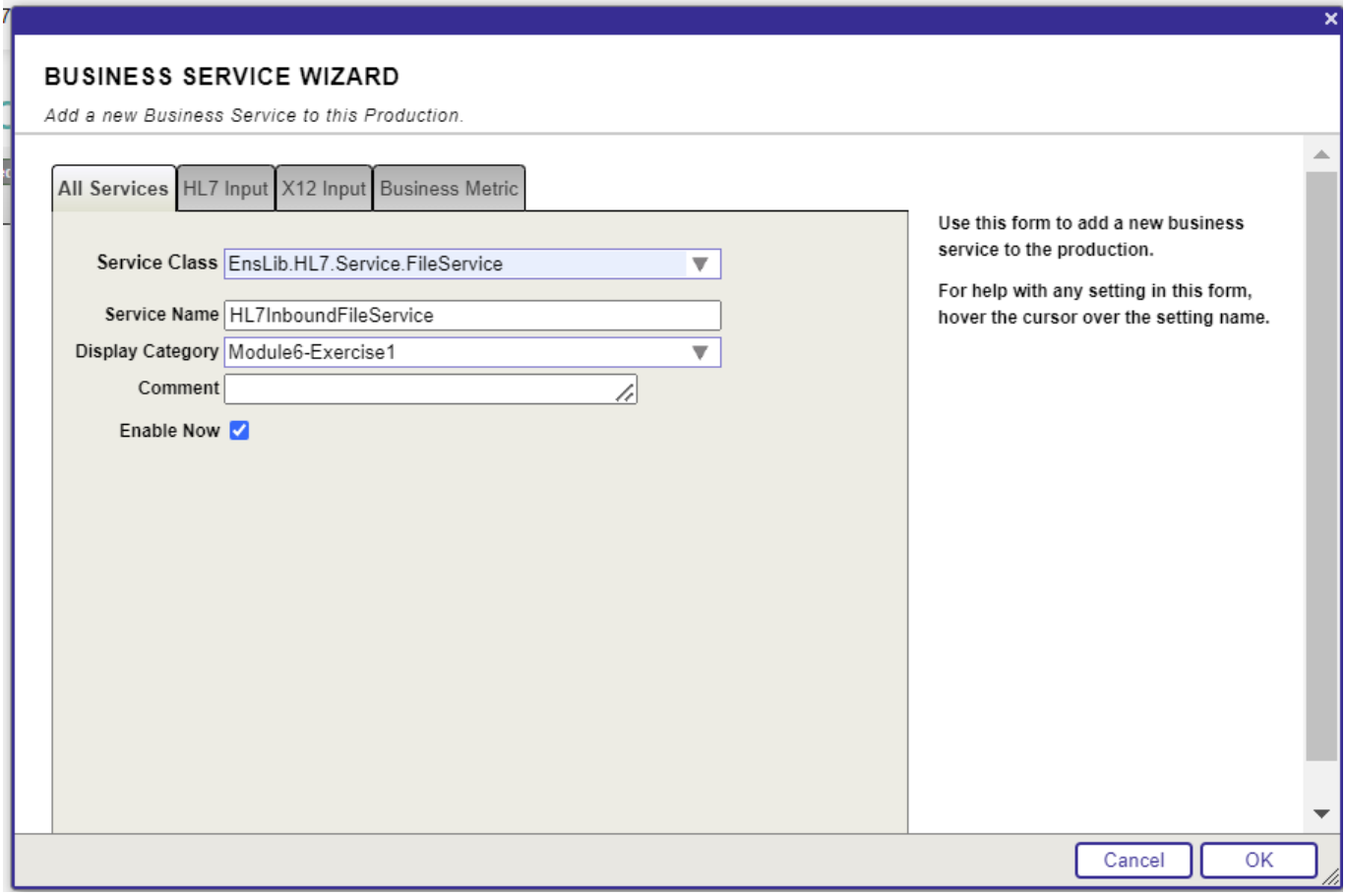
   http://localhost:32783/csp/sys/UtilHome.csp

3. **Open the FHIRDEMO Production:** Navigate to the production by selecting **HOME->Interoperability**. You will see **FHIRDEMO** now. Select **FHIRDEMO** and then in the next screen, click the **Go** button to view the production.

4. **Add a Business Service:** Click on the plus `+` next to the **Services** header.

Configure these **Business Service** settings in the wizard:

Configuration Name	Value
Service Class	EnsLib.HL7.Service.FileService
Service Name	HL7InboundFileService1
Display Category	Module6-Exercise1
Enable Now	Selected



The service should be added to the production now.

6. **Add a Business Process:** Click on the + symbol next to the **Processes** header.

Configure these **Business Process** settings in the wizard:

Configuration Name	Value
Business Process Class	HS.Local.FHIR.HL7toSDAProcess
Business Process Name	FHIR.HL7toSDA1
Display Category	Module6-Exercise1
Enable Now	Selected

7. **Add the Standard FHIR Business Process:** Click on the + symbol next to the **Processes** header. (We will have two business prcesses).

Configure these **Business Process** settings in the wizard:

Configuration Name	Value
Business Process Class	HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process
Business Process Name	HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process1
Display Category	Module6-Exercise1
Enable Now	Selected

8. **Add a Standard FHIR Server Operation:** Click on the + symbol next to the **Operations** header.

Configure these **Business Operation** settings in the wizard:

Configuration Name	Value
Operation Class	HS.FHIRServer.Interop.Operation
Operation Name	HS.FHIRServer.Interop.Operation
Display Category	Module6-Exercise1
Enable Now	Selected

9. **Build End-to-End**: Now you have four business components. In order to hook them together to run and end-to-end, we'll cover the **Properties** for each of the components.

9-1. Start on the left by clicking on the icon/name for the **HL7InboundFileService1** service:

Then click on the **Settings** tab on the right panel to configure the service properties:

Property Name	Value
File Path	/irisdata/module6-exercise1-inbound/
TargetConfigNames	FHIR.HL7toSDA1

Make sure to click **Apply** to save your Settings.

9-2. Click on the icon/name for the **FHIR.HL7toSDA1** Process.

This is a custom process that is identifying what HL7 field will be set as the `PatientResourceId`, which is required in the Patient Resource. Configure the following Settings.

Property Name	Value
PatientIdLocation	PID:3.1
TargetConfigNames	HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process1

Make sure to click **Apply** to save your Settings.

9-3. Click on the icon/name for the **HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process1** Process.

Configure the following Settings.

Property Name	Value
TargetConfigNames	HS.FHIRServer.Interop.Operation
TransmissionMode	transaction
FHIRMetadataSet	HL7v40 / FHIR R4 Core Specification
FHIREndpoint*	/csp/healthshare/fhirdemo/fhir/r4

Property Name	Value
LogTraceEvents	*checked*
TraceOperations	`*FULL*`

- The **FHIREndpoint** path doesn't yet exist, but you will configure this in a few steps.

Make sure to click **Apply** to save your Settings.

9-4. Click on the icon/name for the **HS.FHIRServer.Interop.Operation** Operation.

Configure the following Settings.

Property Name	Value
LogTraceEvents	*checked*
TraceOperations	`*FULL*`

Make sure to click **Apply** to save your Settings.

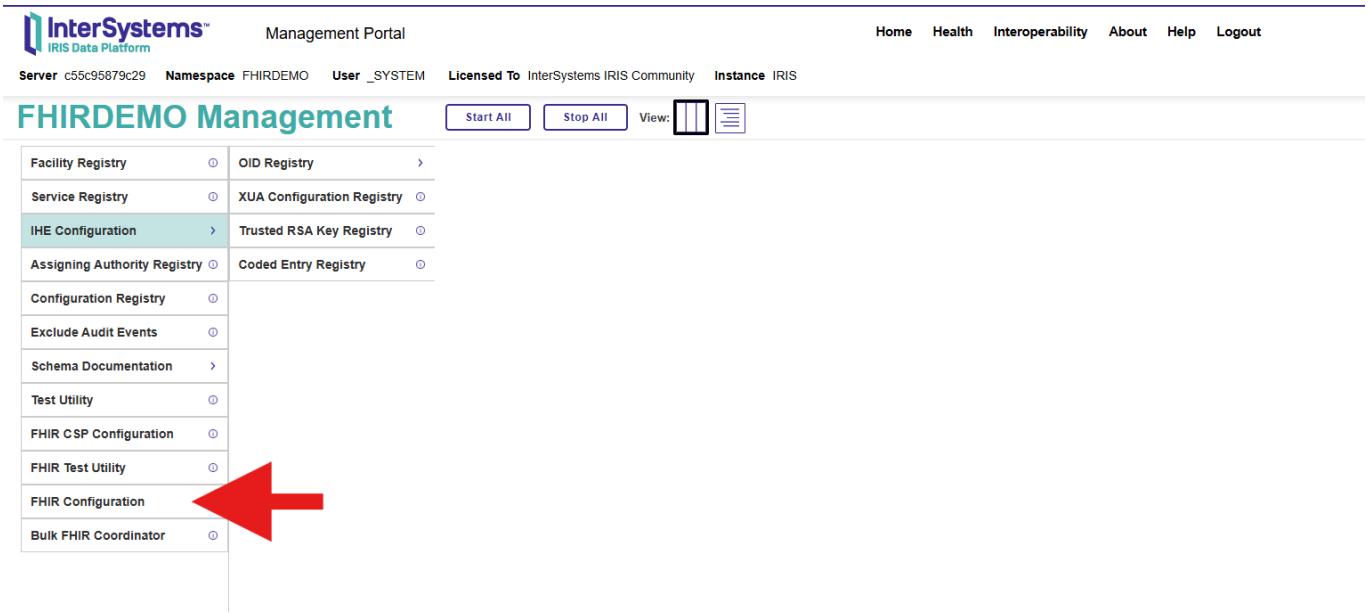
10. **Add the Trace Logging Operation**

Click on the `+` on the Operations header to the right and add `HS.Util.Trace.Operations` as the Class name and Operation name. Select `Enable` and **Apply**.

11. **Configure the FHIR Server**

With the **FHIRDEMO** namespace selected, click on **Home** and then select **HEALTH** either on the panel on the left or at the top of the System Management portal.

Select the **FHIR Configuration** section:



11-1. Log in with the same username and password for IRIS. User:`_system` Password: `SYS`

11-2. Select **Server Configuration** and then click the **Add Endpoint** button.

Enter these configurations: Configure the following Settings.

Property Name	Value
CORE FHIR package	hl7.fhir.r4.core@4.0.1
URL	/csp/healthshare/fhirdemo/fhir/r4
Additional packages	hl7.fhir.us.core@3.1.0
Interactions Strategy Class	HS.FHIRServer.Storage.Json.Interactions.Strategy
Storage	*Keep all default values*

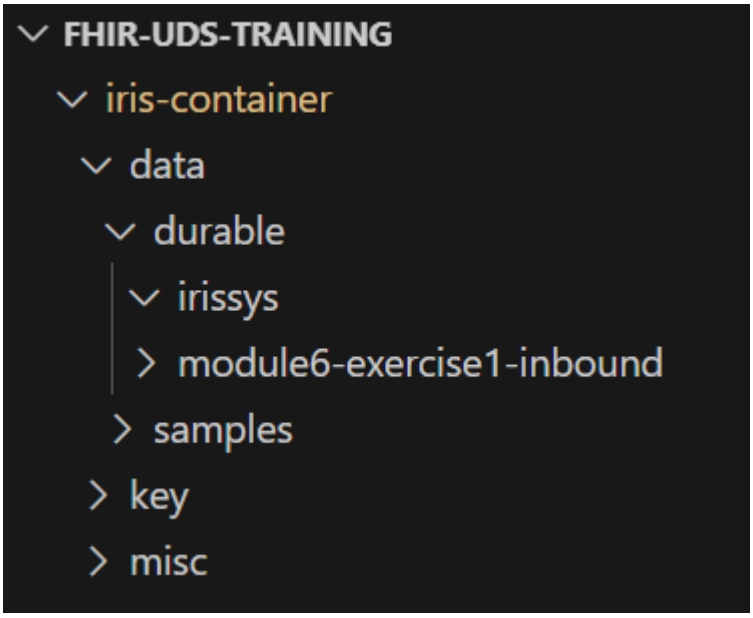
Click **Add**. It will take a few minutes to build the endpoint. You can leave this screen and return to VSCode while the endpoint builds.

12. **Create the data iput directory in VS Code**

Typically, the file system in the Docker container is completely separate from the folders and files you have in the project folder. However, the container has been set up with a bind mount that connects the `iris-container/data/durable` folder in the FHIR-UDS-TRAINING project with the `/irisdata` folder in the IRIS instance.

12-1. In VSCode, right click on the `iris-continer/data/durable` folder and select `Add Folder`. Name the folder `module6-exercise1-inbound` in order to match what you configured in the Business Service Settings in the IRIS production.

Your folders will look like this (Don't worry if the top-level name is more detailed than `FHIR-UDS-TRAINING`):



12-2. Open the `samples` folder. Right click on the `ADT_A01JohnDoe.hl7` file and select `Copy`.

12-3. Click on the `module6-exercise1-inbound` folder. Right click and select `Paste`.

You will see the file copied into the directory and then it may disappear. This is good news. The inbound file service running in IRIS has picked it up and attempted to process it.

13. **Check the Production:** Return to your System Management Portal. If you are looking at the "FHIR Server" screen, you can click on the profile for the `_system` user in the right corner. Once you click on the icon, select **Management Portal** to retrn **Home**.

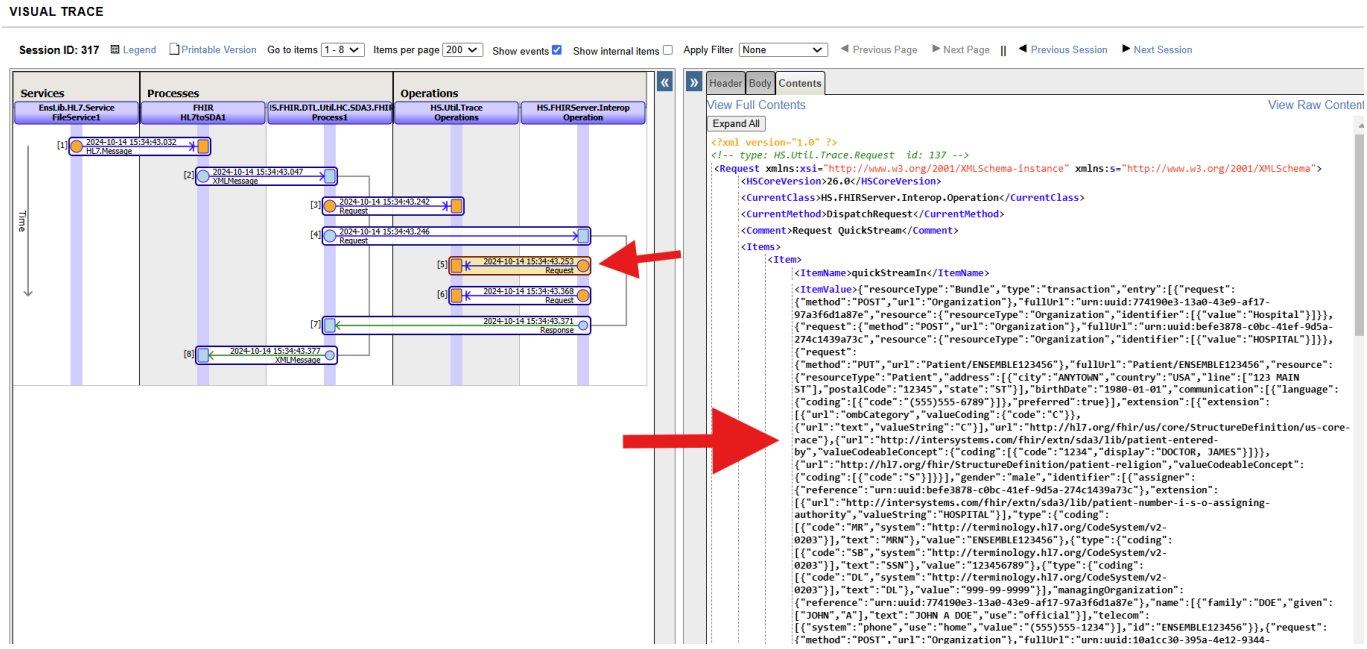
Go to **Home -> Interoperability -> Select FHIRDEMO -> Configuration -> Production**.

Click on the **Messages** tab. You should see the available message traces. Click on the link under **Header** to trace the activity.

If you see errors, read the error messages, doouble check settings, and try to fix things so you get a complete message trace (see below). If you ever need to re-run the message, you can re-send from the **Message Viewer** or drop the file again like you did in **Step 14**.

14. **Review the Message Trace:**

If everything has gone well, Step 5 of the trace will show the actual FHIR bundle that was sent to the local FHIR Server.



15. Review the steps in the message trace. Can you confirm whether or not the FHIR bundle posted successfully? What was the response status code from the FHIR Server?

16. You turned on `*FULL*` Trace so you could see detailed information like the actual FHIR bundle.

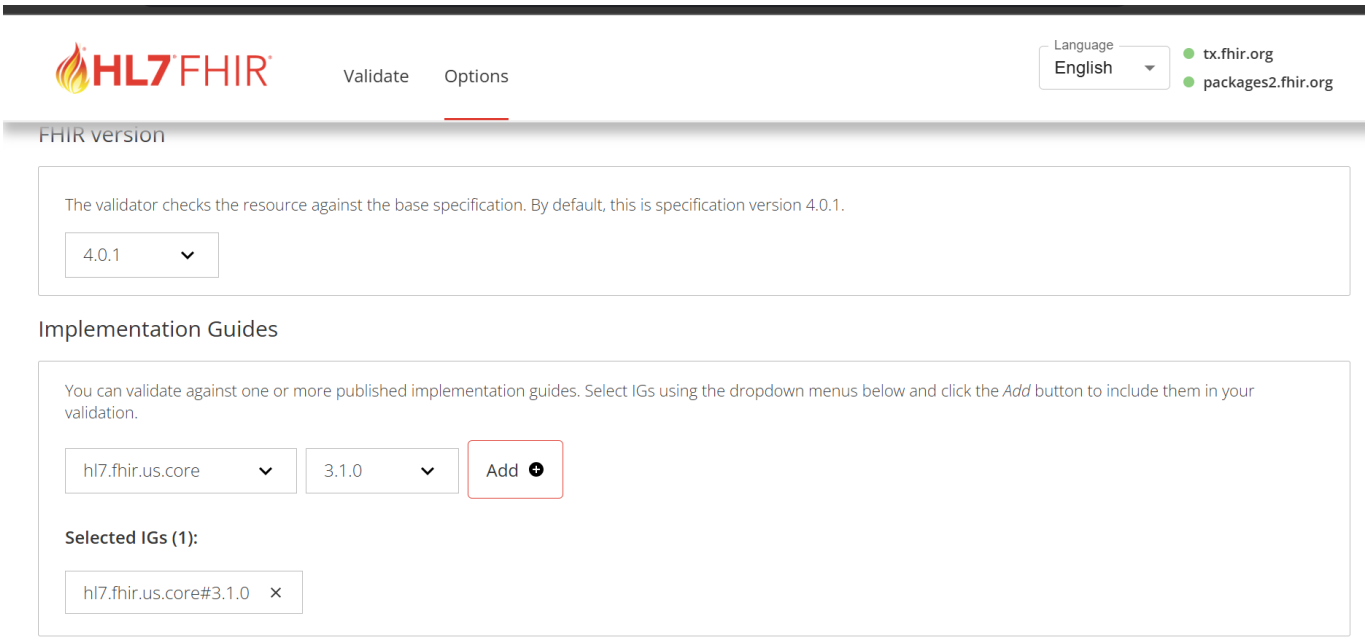
> This is a good technique for testing, but should be turned off once the interface is in production as FHIR bundles can get very big.

Click on Step 5 and on `View Raw =Contents`. Copy the JSON string in the tags, but do not include the tag.

17. Go to the FHIR validator at FHIR Validator.

17-1. Select the **Options** menu at the top. Then select the **FHIR Version (4.0.1)** and **Implementation Guides (hl7.fhir.us.core version 3.1.0)** and select **Add**.

This is the version of the FHIR Server we are checking against (remember when we configured those packages on the endpoint?)



17-2. Click back on **Validate** and paste the JSON into the **Enter Resource** window.

Select the **Validate** button at the bottom of the screen and review the errors and warnings.

18. Even though the FHIR bundle was accepted by the IRIS FHIR Server, there are many errors still when the message is validated against the official specifications. FHIR Servers will vary in how strict their validations are and whether they are configured to reject messages or accept them when there are non-conformance issues that are not deemed fatal.

19. Now that you have completed an end-to-end, go back to the **FHIRDEMO** and try to figure out how to do each of these:

19-1. What happens when you send another HL7 message through? What about an ORU?

19-2. How can you configure the components to send indivudal resources rather than an entire bundle at once?

19-3. How can you configure the feed to use the **Message Control ID (MSH:10)** as the `PatientResourceId`? (You wouldn't necessarily want to do this, but it is good to see how the message changes when that's done)

19-4. **Bonus!!!** Can you figure out how to use Postman to query back the message you just sent in?

> DO NOT do a ***COMPOSE RESTART*** on the container. This will rebuild the container and you will lose all changes unless you have saved them first.

To Save the Production:

You can copy the contents of the **FHIRDEMO.FoundationProduction** class to the **FHIR-UDS-TRAINING/src/FHIRDEMOPKG** folder via Cache Studio or Visual Studio Code.

> Tip: You can keep the Docker service > running in the background while you work. If you want to shut it down, select **Compose - Down**. When you want to restart it, select **Compose - Up**. It will start up much faster than when you select **Compose - Restart** however all your coding and configuration changes will be reset.

> Solution: There is a completed production class saved in the Module 6 Solutions Folder. This contains the production configuration, but does not contain the FHIR Server configuration.

Module 6 Exercise 2 - Creating a Custom DTL

Learning Objective:

- Identify the location of standard DTLs
- Create custom DTLs for mapping FHIR resources
- Create sub-transform DTL for mapping UDS age extension
- Validate FHIR bundle against UDS definitions

Reference: SDA to FHIR Transformation Product Documentation

Task: The UDS Profile for the Patient Resource contains extensions that are unique to UDS. In this exercise, you will be customizing the Patient Resource transformation in order to include the UDS Plus Age Extension. The output format should look like the following sample:

```
{
      "url" : "http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-age-extension",
      "valueQuantity" : {
        "value" : 70,
        "unit" : "yr",
        "system" : "http://unitsofmeasure.org",
        "code" : "a"
}
```

Requirements: Calculate the age *value* by taking the current year and subtracting the year sent in the **source.birthTime** field in the SDA. You will hardcode the other values in the extension.

Setting up Custom Transformation

First set the location of the custom DTL library

1. Open the InterSystems Terminal from VS Code. First make sure the Docker container is running. Open a Terminal by going to **View Menu -> Terminal**

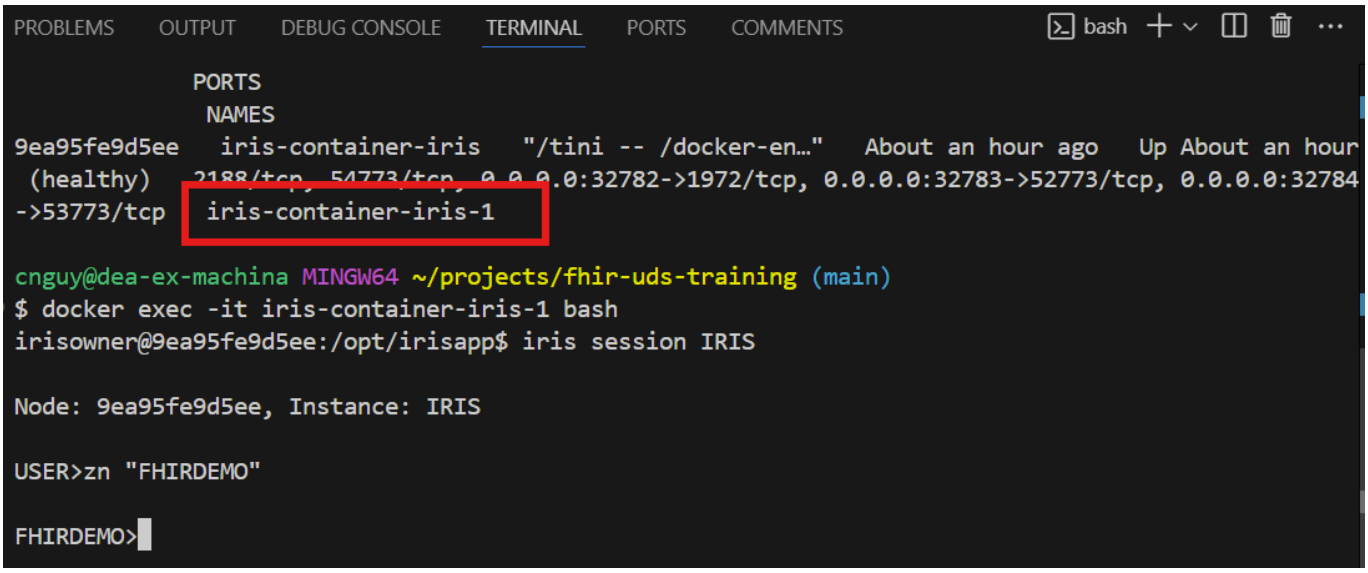
2. This opens the Terminal window at the bottom of the project. (It may already be open).

- Find the `+` sign with the pull down.
- Select `Git Bash` as the terminal type.
- Type `docker ps` to figure out the name of the container. In this case `iris-container-1`
- Once you have the name, type this command to start the shell. You will be looking at the internal Docker file system:

```
docker exec -it **iris container name** bash
```

- From the new command prompt, open IRIS terminal by typing: `iris session IRIS`
- Log in with the `_system/SYS` user/password

Here is a screenshot showing the commands:



3. Change to **FHIRDEMO** namespace:

```
set $namespace = "FHIRDEMO"
```

4. To check if a custom DTL package already exists, enter:

```
Write ##class(HS.FHIR.DTL.Util.API.ExecDefinition).GetCustomDTLPackage()
```

5. If the custom DTL package does not already exist, enter the following command which designates **HS.Local.FHIR.DTL** as the name of your custom DTL package:

```
set status =
##class(HS.FHIR.DTL.Util.API.ExecDefinition).SetCustomDTLPackage("HS.Local.FHIR.DTL")
```

6. To check that the package was defined successfully, enter:

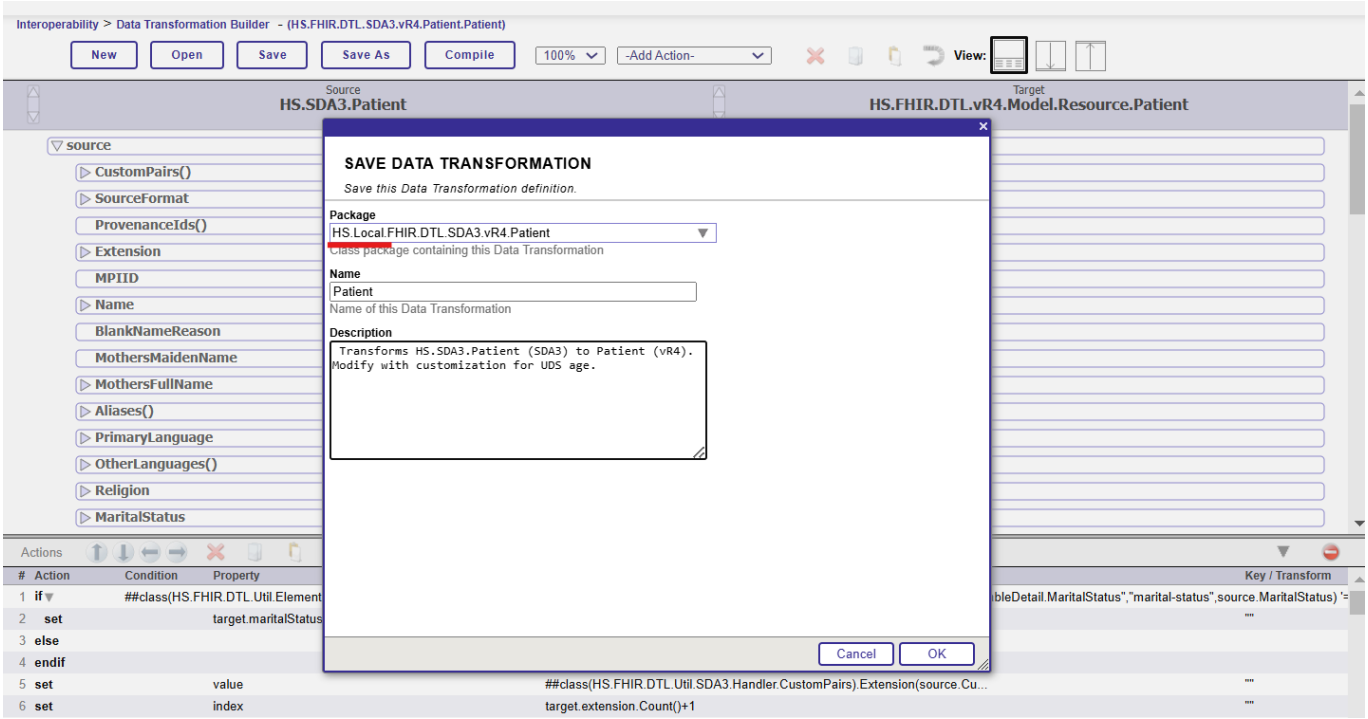
```
write status
```

The response should be: 1 which means the process was successful. You have set your custom DTL package. The FHIR processes will automatically give precedence to any versions of the DTL transforms located **HS.Local** in the **FHIRDEMO** namespace.

Modifying the DTL Transformation Code

1. Open up the Patient Resource DTL: **HS.FHIR.DTL.SDA3.vR4.Patient.Patient**

2. Click on **Save As** to copy the **HS.FHIR.DTL.SDA3.vR4.Patient.Patient** class to a new class called: **HS.Local.FHIR.DTL.SDA3.vR4.Patient.Patient**. The naming is important here as the FHIR base code will give this custom class precedence over the out-of-the-box transform.



3. Create a Sub-transform to do the work of mapping the extension.

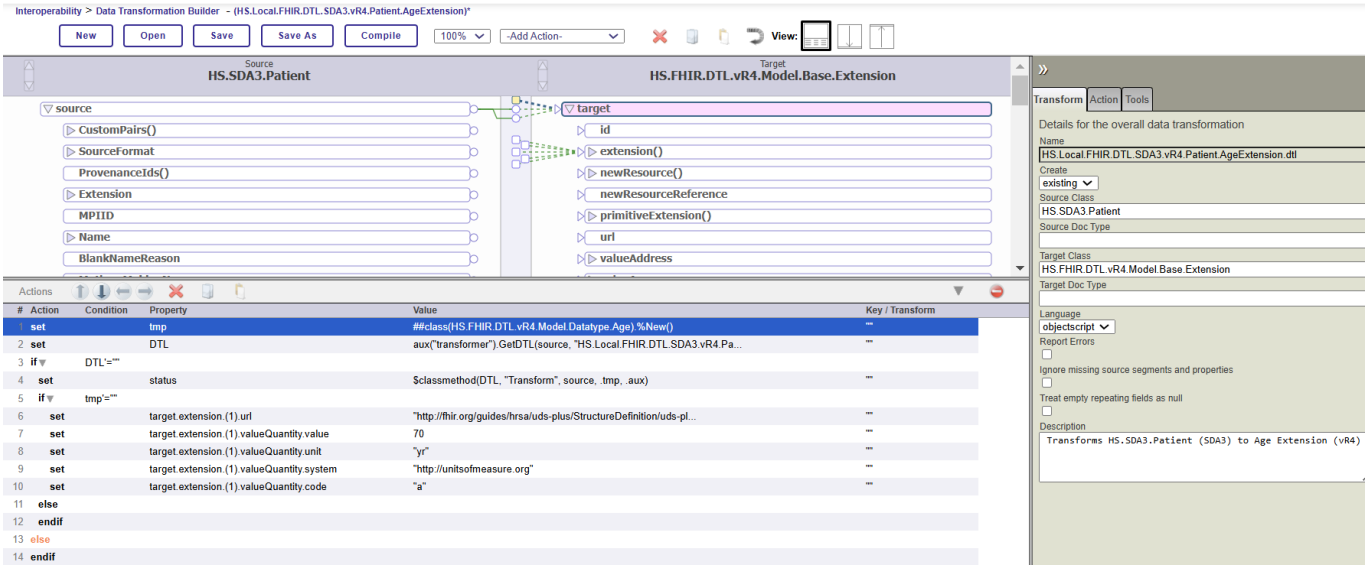
From the Data Transformation Builder, open **HS.FHIR.DTL.SDA3.vR4.Address.Extension** as a model of an extension mapping.

3-1. Save As: **HS.Local.FHIR.DTL.SDA3.vR4.Patient.AgeExtension**

The location of this is not as sensitive as the code you write will reference this class directly.

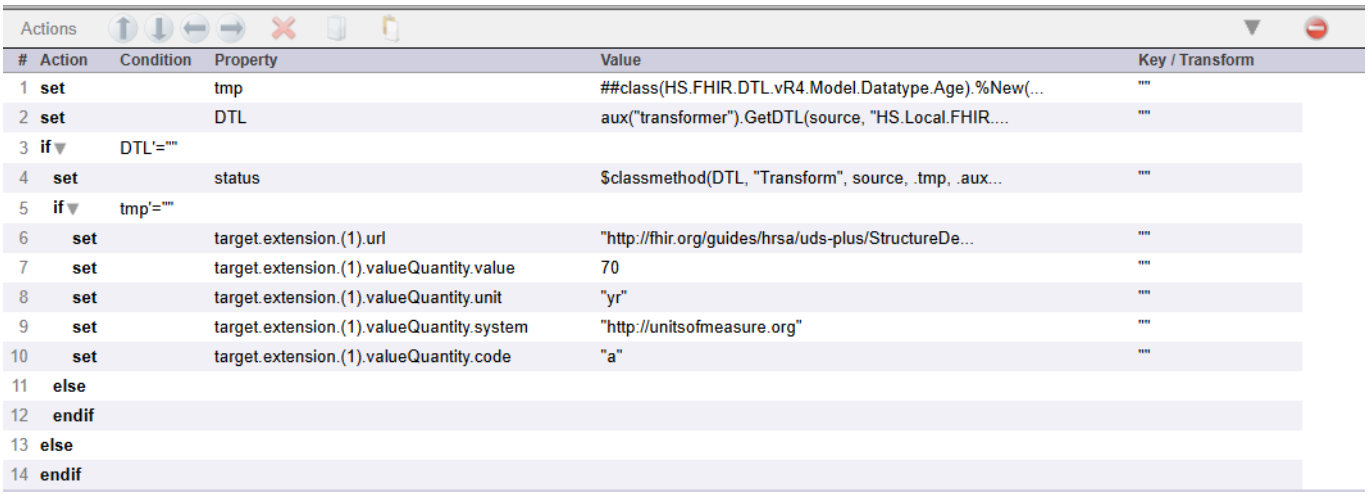
3-2. Click on the **Transform** tab and update the following settings:

Setting Name	Value
Source Class	HS.SDA3.Patient
Target Class	HS.FHIR.DTL.vR4.Model.Base.Extension



4. Following the existing code in the DTL as a model, modify the values to follow the same pattern, but map out the **UDS Plus Age Extension** as detailed in the **Task** section above.

Example: This image shows an example with the age (target.extension(1).valueQuanity.value) hard-coded rather than calculated. You can do that as a first step.



Actions

#	Action	Condition	Property	Value	Key / Transform
1	set		tmp	##class(HS.FHIR.DTL.vR4.Model.Datatype.Age).%New(...	""
2	set		DTL	aux("transformer").GetDTL(source, "HS.Local.FHIR....	""
3	if	DTL'=""			
4	set		status	$classmethod(DTL, "Transform", source, .tmp, .aux...	""
5	if	tmp'=""			
6	set		target.extension.(1).url	"http://fhir.org/guides/hrsa/uds-plus/StructureDe...	""
7	set		target.extension.(1).valueQuantity.value	70	""
8	set		target.extension.(1).valueQuantity.unit	"yr"	""
9	set		target.extension.(1).valueQuantity.system	"http://unitsofmeasure.org"	""
10	set		target.extension.(1).valueQuantity.code	"a"	""
11	else				
12	endif				
13	else				
14	endif				

Save and Compile when done.

5. Open the Transformation for **HS.Local.FHIR.DTL.SDA3.vR4.Patient.Patient**

> Be very careful when editing this code. You are keeping the Patient Resource mapping intact while adding in a call-out to the sub-transform for the Age Extension you just created.

Follow the model in the existing class to call the sub-transform. In the example, the code block was added rigt before the **birthTime** mapping since the **source.birthTime** value is used to caculate the age.

```
comment:  CNR: Hardcoding age extension to try things out
set       index          target.extension.Count()+1                                  ""
set       tmp            ##class(HS.FHIR.DTL.vR4.Model.Base.Extension).%New()        ""
set       DTL            aux("transformer").GetDTL(source, "HS.Local.FHIR.DTL.SDA3.vR4.Patient....  ""
if        DTL'=""
  set     status         $classmethod(DTL,"Transform", source, .tmp, .aux)           ""
  if      tmp'=""
    set   tmp.url        "http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-age...  ""
    set   target.extension  tmp                                                      index
  else
  endif
else
endif
comment:  End of age extension code
```

Save and Compile When done.

Test The Changes

No changes to the existing Production are needed for the local DTL code changes to take effect

1. Repeat the steps you did in **Module 6 - Exercise 1** to drop an HL7 message in **iris-container/data/durable/module6-exercise1-inbound**.
2. Check in the **Message Trace** to ensure there were no errors. Review the output FHIR bundle or transaction to confirm that the update took effect.
3. Troubleshoot the mapping as needed.

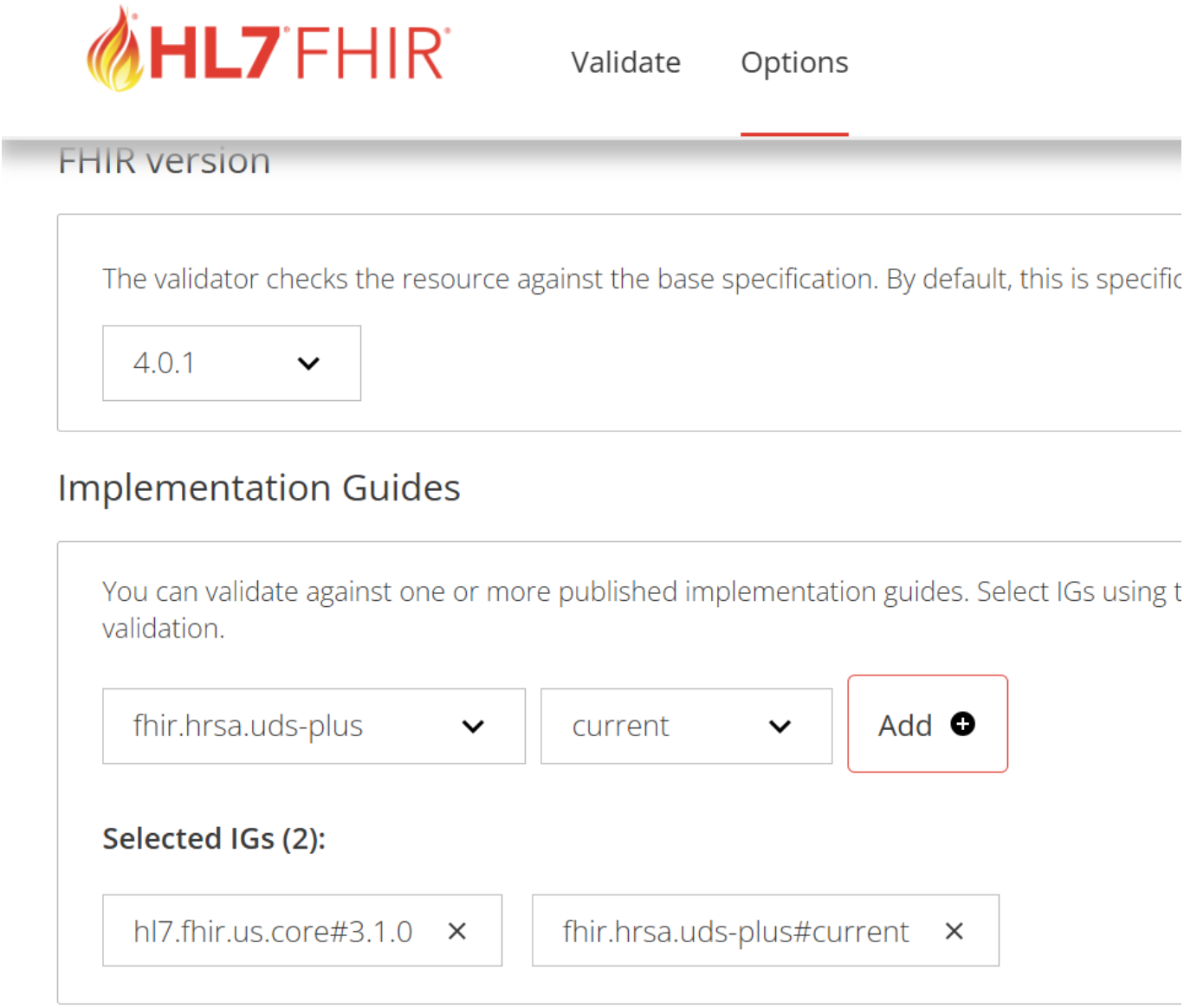
FHIR Validation

1. Go to the FHIR validator at FHIR Validator.

1-1. Select the **Options** menu at the top. Then select the **FHIR Version (4.0.1)** and **Implementation Guides (hl7.fhir.us.core version 3.1.0)** and select **Add**.

1-2. Search for `uds` in the Implementation Guides. Select **fhir.hrsa.uds-plus** and select the **current** version. Click **Add**.

Now you are validating against the UDS+ IG as well! Isn't FHIR fun?



2. Click back on **Validate** and paste the updated JSON into the **Enter Resource** window.

3. Select the **Validate** button at the bottom of the screen and review the errors and warnings.

4. Search through the errors. You should find a few that are specific to the `uds-plus-age-extension`.

Warning Line: 1, Col:852	A code with no system has no defined meaning, and it cannot be validated. A system should be provided

Error Line: 1, Col:1240	The Extension 'http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-age-extension' definition is for a simple extension, so it must contain a value, not extensions

Error Line: 1, Col:1240	Extension.value[x]: minimum required = 1, but only found 0 (from http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-age-extension|1.1.0)

5. To experiment, you can update the uds age extension directly in the window and validate again. This is an easy way to see what the correct mapping is supposed to be.

Summary: We muscled in a change just to see it take effect and start the testing and validation cycle. There's work ahead to complete the full mapping and make sure it conforms, but hopefully this exercise has given you a good idea of how to accomplish this task using the tools available.

> DO NOT do a ***COMPOSE RESTART*** on the container. This will rebuild the container and you will lose all changes unless you have saved them first.

To Save the Production:

You can copy the contents of the **HS.Local.FHIR.DTL.SDA3.vR4.Patient.Patient** and the **HS.Local.FHIR.DTL.SDA3.vR4.Patient.AgeExtension** class to the **FHIR-UDS-TRAINING/src/FHIRDEMO** folder via Cache Studio or Visual Studio Code.

> Tip: You can keep the Docker service > running in the background while you work. If you want to shut it down, select **Compose - Down**. When you want to restart it, select **Compose - Up**. It will start up much faster than when you select **Compose - Restart** however all your coding and configuration changes will be reset.

> Solution: There is a completed production class saved in the Module 6 Solutions Folder. This contains the production configuration, but does not contain the FHIR Server configuration.

Module 7 Exercise 2 - Creating SDA Extensions

Learning Objectives:

- Extending the `HS.SDA3.Patient` class
- How to load and activate custom SDA extensions
- Updating existing Transformation DTLs with new extensions added

Reference:

This information can also be found in the InterSystems documentation: Customizing the SDA

Task: In this exercise, you will build on the previous exercise by extending the Patient class in the SDA to include custom properties

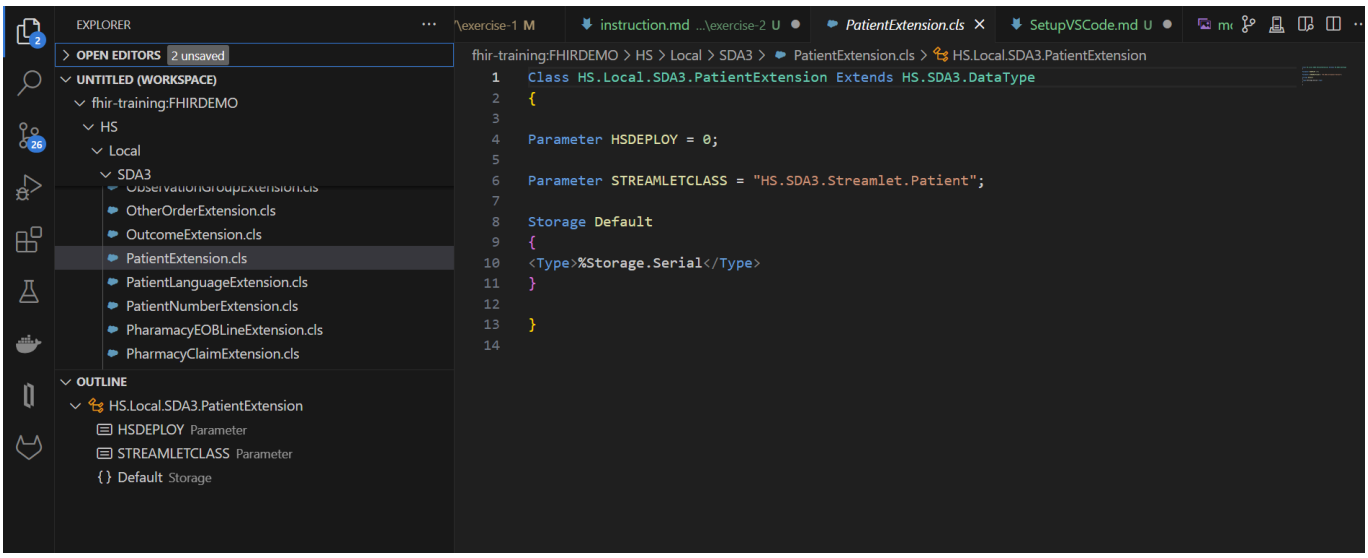
Instructions:

Before you begin: For this exercise, you will be accessing the IRIS classes using the **InterSystems Language Server - VSCode Extension**.

> If you do not have the Extension installed, refer to the instructions in **Module 5**.

Task 1: Modifying the Local Version of the SDA Class

1. In your code editor, open the **HS.Local.SDA3.PatientExtension** class. This is located in the **HSCUSTOM** namespace but should be package mapped to all Foundation namespaces and available from **FHIRDEMO**.



2. In order to add extensions, add properties to the class. Here's an example adding the `AgeExtension` and `AgeExtensionURL` properties:

```
fhir-training:FHIRDEMO > HS > Local > SDA3 > PatientExtension.cls > HS.Local.SDA3.PatientExtension
Class HS.Local.SDA3.PatientExtension Extends HS.SDA3.DataType
{

Parameter HSDEPLOY = 0;

Parameter STREAMLETCLASS = "HS.SDA3.Streamlet.Patient";

Property AgeExtension As %String;

Property AgeExtensionURL As %String(MAXLEN = 256) [ InitialExpression = "http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-age-exten

Storage Default
{
<Type>%Storage.Serial</Type>
}

}
```

Follow the model and add these properties:

Property Name	DataType	MAXLEN	Initial Expression
AgeExtension	%String		
AgeExtensionURL	%String	MAXLEN = 256	"http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-age-extension"
AgricultureWorkerStatus	%String		
AgricultureWorkerStatusURL	%String	MAXLEN = 256	"http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/
HousingStatus	%String		
HousingStatusURL	%String	MAXLEN = 256	"http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-housing-status-extension"
VeteranStatus	%String		
VeteranStatusURL	%String	MAXLEN = 256	"http://fhir.org/guides/hrsa/uds-plus/StructureDefinition/uds-plus-veteran-status-extension"

3. Recompile the `HS.SDA3` classes.

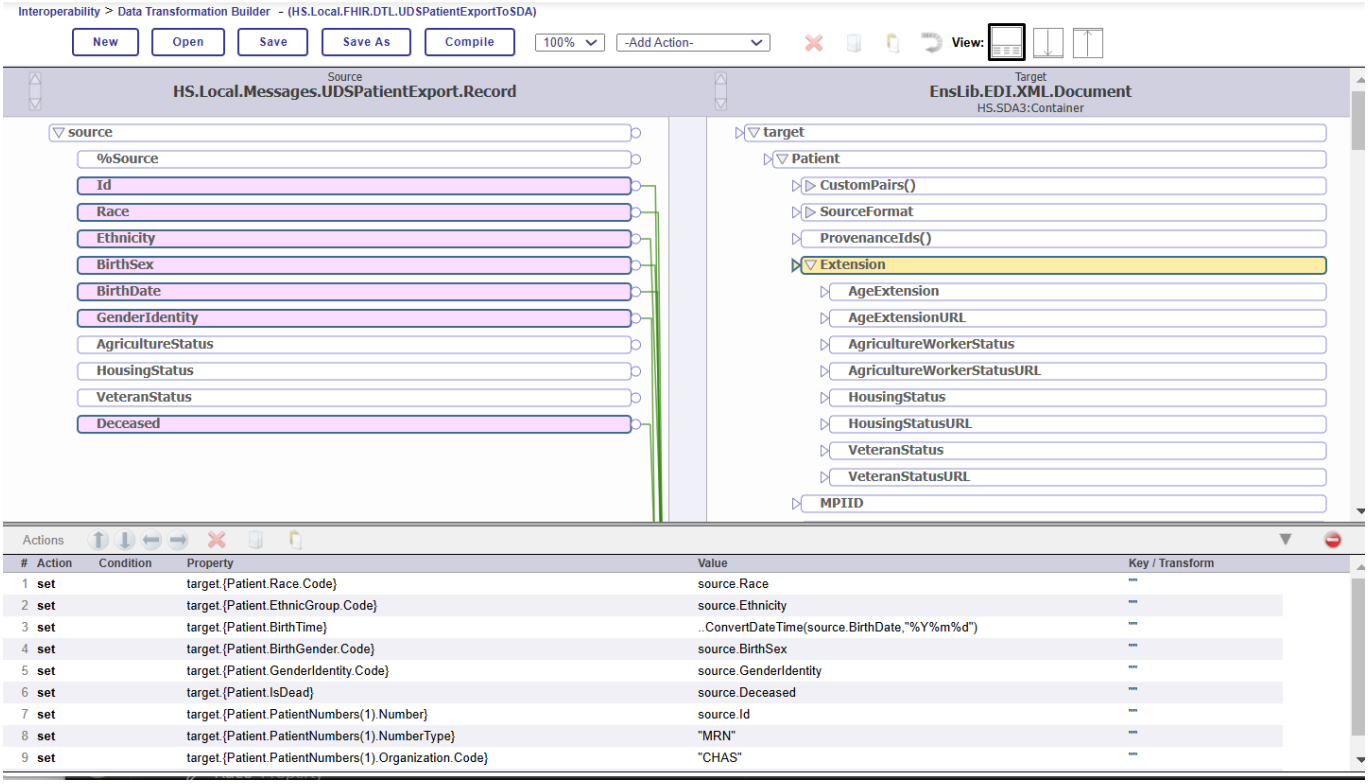
- In VSCode, navigate to the `HS.SDA3` package in **FHIRDEMO**. These packages are mapped from **HSLIB** and should be available from any Foundation namespace.
- Right click and select **Compile**. This will compile the new version of the `HS.Local.SDA3.PatientExtension` into the package.

4. Import the HS.SDA3 Schema into the namespace.

- In Management Portal: Navigate to **Health > HSDEMO > Schema Documentation**
- Click the **Import SDA3 Schema** button.

5. If you re-open the DTL you created in the previous exercise, the **Patient** class inside the SDA Container should have the new extensions you added. You may need to re-open the DTL if you had it open all this

time. If you don't see the extensions still, repeat the compilation and import steps.



Task 2 - Map Source Data into the New Extensions

1. Modify your existing DTL by mapping the source fields into the extensions.
2. You do not need to map anything into the "URL" extensions. Remember the `InitialExpression` values you set? These properties are defaulted to those values if left empty.

Task 3 - Test the new Extension mapping

1. Run a test by resending a previous message or by copying the `UDS_patient_export_sample.txt` from the **samples** directory into the **iris-container/data/durable/module7-exercise1-inbound** folder.

2. Confirm in the Message Viewer that the SDA extension mapping is carrying through in the SDA that is sent to the FHIR Process.

3. Those new extensions are not yet mapped to the FHIR Patient resource. The next step would be to add those extension mappings to the customized **HS.Local.FHIR.DTL.SDA3.vR4.Patient.Patient** that you created in **Module 6 - Exercise 2**.

> Solution: The completed extension class is located in the Module 7 Solutions Folder. The extension is included, but not the final DTL Transformation.

Module 7 Exercise 1 - Mapping Custom File Data to SDA

Learning Objectives:

- Use the Record Mapper Utility for Delimited Data
- Create a Transformation DTL for XML Data
- Map a proprietary, delimited message to HS.SDA3.Container
- Validate a FHIR Resource using the FHIR Vaildator

Task: In this exercise, you will take a sample batch file in pipe-delimited format and transform it directly to SDA format which will then be transformed to a FHIR Patient Resource.

Instructions:

Task 1: Build the Record Map

The sample file is pipe-delimited (|) and contains a batch of patient records.

1. Create a new folder in the project at `iris-container/data/durable/` named `module7-exercise1-inbound`.

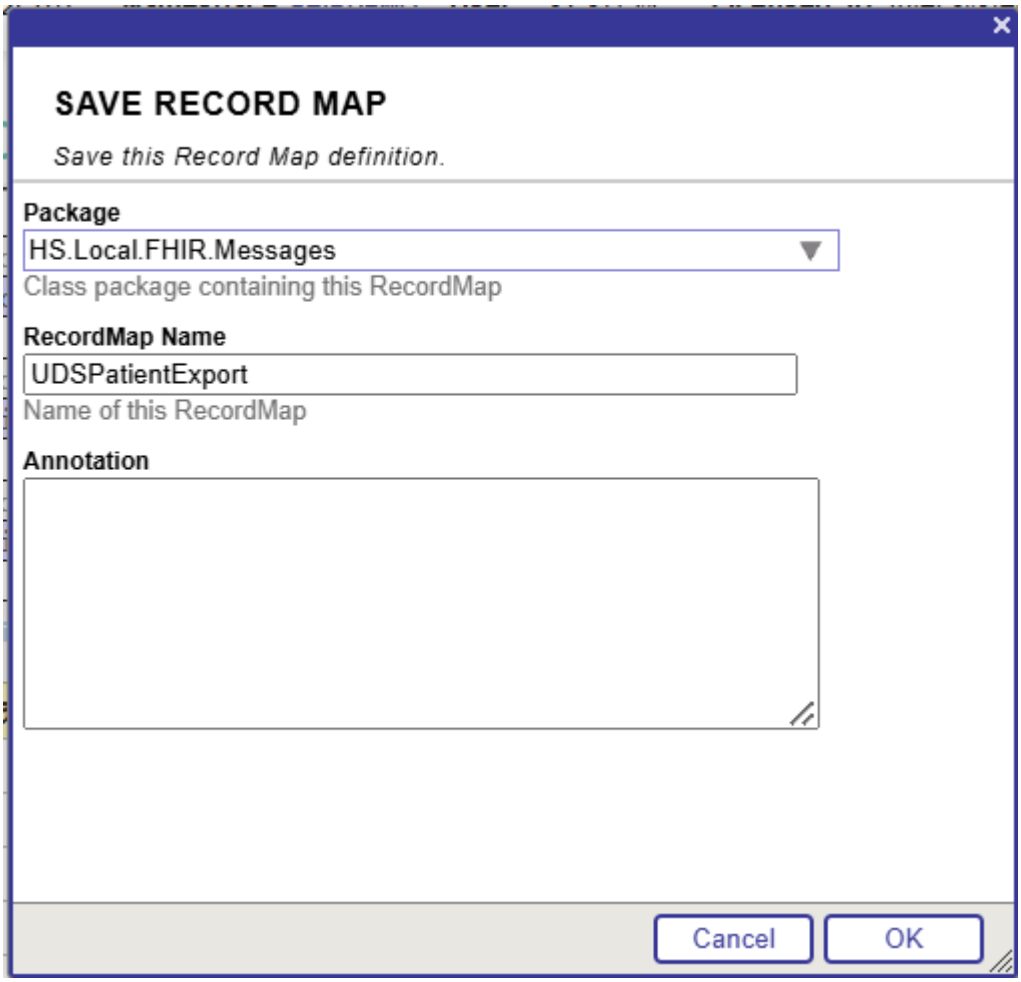
2. Locate the sample file located in `iris-container/data/samples/UDS_patient_export_sample.txt`. Copy the file and paste it into the new folder created in Step 1. (This is necessary because the IRIS instance can only "see" files that are in the container environment. The `/data/durable` folder is mounted so it is visible from both the external and in-container environment)

3. **Open the Record Mapper Utility:** From the Home page of the Management Portal, go to **Interoperability -> Build -> CSV Record Wizard**.

4. **Fill out the form with the following values:** First select the `UDS_patient_export_sample.txt` file by navigating `/irisdata/module7-exercise1-inbound`. You will have to change the filter to “All files”.

- RecordMap name: UDSPatientExport
- Separator: | (found on right side, SHIFT-backslash key)
- Record Terminator: CRLF
- Character Encoding: UTF-8
- Sample has header row: *checked*
- Click **Create RecordMap**

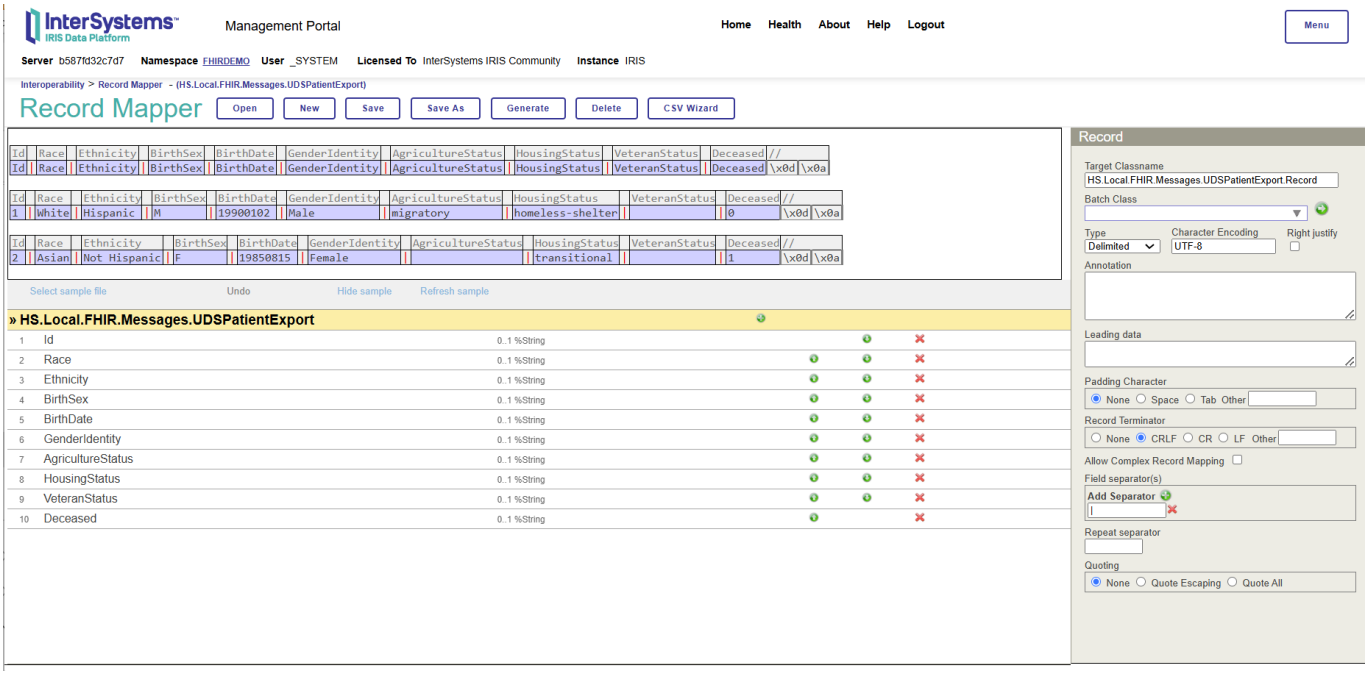
5. Click on the **Save As** button at the top. Rename the class name to package value: `HS.Local.FHIR.Messages` and Classname: `UDSPatientExport` so that the class saves in **HS.Local**.



6. Click on the **Generate** button. This will generate two classes:

- HS.Local.FHIR.Messages.UDSPatientExport
- HS.Local.FHIR.Messages.UDSPatientExport.Record

7. **Final Record Mapper Screen** This screen can be used to edit the data type of the fields and add annotations for documentation as well deal with more complex formats with repeating structures, but for this exercise, you will deal with everything just as Strings and keep things flat.



Task 2 - Map to SDA

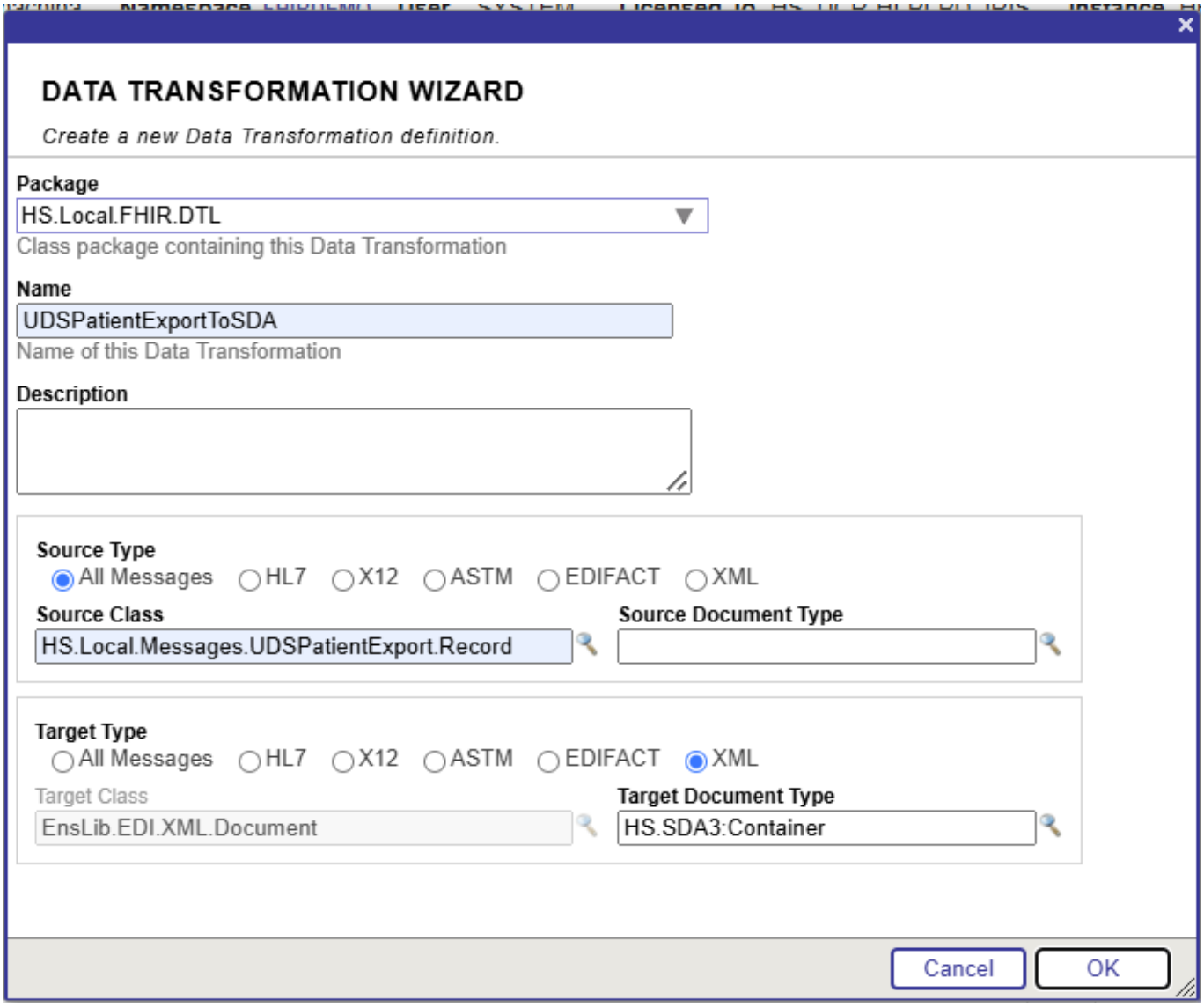
Now that you have the source format built, you will need to determine where the fields need to be mapped in the SDA in order to map to the relevant location in the FHIR resource.

1. Use the **FHIR Annotations** lookup from the Module 5 exercise to fill out the mapping table. Whenever you find a field that doesn't have a direct mapping to the current SDA or FHIR R4/USCDI Resource, make a note about it. This will be addressed in the next exercise.

Source Field	SDA Location	FHIR R4/USCDI Resource
Id		
Race		
Ethnicity		
BirthSex		
BirthDate		
GenderIdentity		
AgricultureStatus		
HousingStatus		
VeteranStatus		
Deceased		

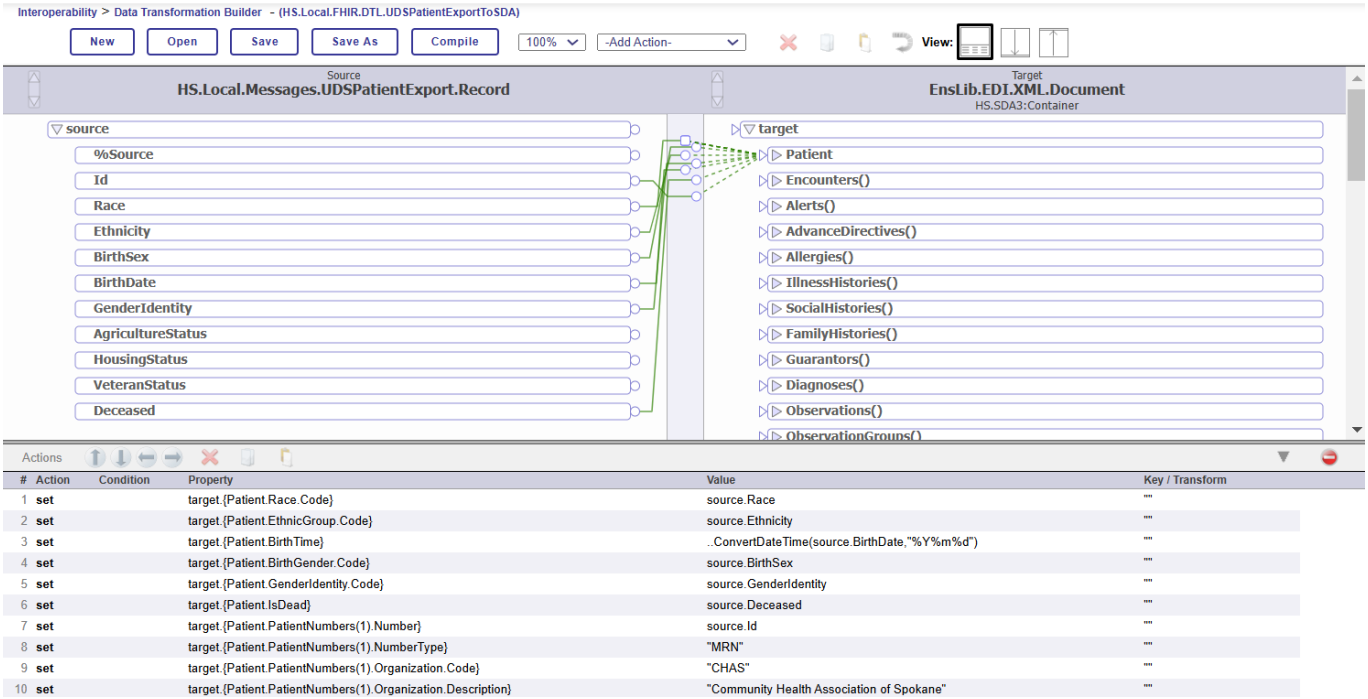
2. Create a transformation DTL to map from **UDSPatientExport** record to the **HS.SDA3.Patient**.

- Open the DTL Editor from the Management Portal: **Interoperability -> Build -> Data Transformations**.
- Select New.
- Fill out the Data Transformation Wizard:
  - Package: HS.Local.FHIR.DTL
  - Name: UDSPatientExportToSDA
  - Description: Map from Patient Record to SDA
  - Source Type: All Messages
  - Source Class: HS.Local.FHIR.Messages.UDSPatientExport.Record
  - Target Type: XML
  - Target Class: EnsLib.EDI.XML.Document
  - Targe Document Type: HS.SDA3.Container
  - Click OK



4. Transforming Data

Use the mapping table you worked on previously to map the inbound message to the SDA Patient section. (Note: It can be done with `set` statements)



Task 3 - Create an end-to-end routing

1. **Open the FHIRDEMO Production:** Navigate to the production by selecting **HOME->Interoperability**. You will see **FHIRDEMO** in the list of namespaces. Select **FHIRDEMO** and then in the next screen, click the **Go** button to view the production.

2. **Add a Business Service:** Click on the plus + next to the **Services** header.

Configure these **Business Service** settings in the wizard:

Configuration Name	Value
Service Class	EnsLib.RecordMap.Service.FileService
Service Name	UDSPatientExportFileService
Display Category	Module7-Exercise1
Enable Now	Selected

The service should be added to the production now.

6. **Add a Business Process:** Click on the + symbol next to the **Processes** header.

Configure these **Business Process** settings in the wizard:

Configuration Name	Value
Business Process Class	HS.Local.FHIR.RecordMappertoSDAProcess
Business Process Name	FHIR.RecordMappertoSDA
Display Category	Module7-Exercise1
Enable Now	Selected

7. **Add the Standard FHIR Business Process:** Click on the + symbol next to the **Processes** header. (We will have two business prcesses).

Configure these **Business Process** settings in the wizard:

Configuration Name	Value
Business Process Class	HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process
Business Process Name	HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process2
Display Category	Module7-Exercise1
Enable Now	Selected

8. **Add a Standard FHIR Server Operation:** Click on the + symbol next to the **Operations** header.

Configure these **Business Operation** settings in the wizard:

Configuration Name	Value
Operation Class	UDSFHIRServer.Operation
Operation Name	HS.FHIRServer.Interop.Operation
Display Category	Module7-Exercise1
Enable Now	Selected

9. **Build End-to-End**: Now you have four business components. In order to hook them together to run and end-to-end, we'll cover the **Properties** for each of the components.

9-1. Start on the left by clicking on the icon/name for the **UDSPatientExportFileService** service:

Then click on the **Settings** tab on the right panel to configure the service properties:

Property Name	Value
File Path	/irisdata/module7-exercise1-inbound/
TargetConfigNames	FHIR.RecordMappertoSDA

Make sure to click **Apply** to save your Settings.

9-2. Click on the icon/name for the **FHIR.HL7toSDA1** Process.

This is a custom process that is identifying what Record Field will be set as the `PatientResourceId`, which is required in the Patient Resource. Configure the following Settings.

Property Name	Value
PatientIdLocation	Id
DTLTransformationClass	HS.Local.FHIR.DTL.UDSPatientExportToSDA
TargetConfigNames	HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process2

Make sure to click **Apply** to save your Settings.

9-3. Click on the icon/name for the **HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process2** Process.

Configure the following Settings.

Property Name	Value
TargetConfigNames	UDSFHIRServer.Operation
TransmissionMode	transaction
FHIRMetadataSet	HL7v40 / FHIR R4 Core Specification
FHIREndpoint*	/csp/healthshare/fhirdemo/fhir/uds
LogTraceEvents	*checked*
TraceOperations	`*FULL*`

- The **FHIREndpoint** path doesn't yet exist, but you will configure this in a few steps.

Make sure to click **Apply** to save your Settings.

9-4. Click on the icon/name for the **UDSFHIRServer.Operation** Operation.

Configure the following Settings.

Property Name	Value
LogTraceEvents	*checked*
TraceOperations	`*FULL*`

Make sure to click **Apply** to save your Settings.

10. **Make Sure `HS.Util.Trace.Operations` is added**

This may already be done from previous exercise, if not...

Click on the `+` on the Operations header to the right and add `HS.Util.Trace.Operations` as the Class name and Operation name. Select `Enable` and **Apply**.

11. **Configure the FHIR Server**

With the **FHIRDEMO** namespace selected, click on **Home** and then select **HEALTH** either on the panel on the left or at the top of the System Management portal.

Select the **FHIR Configuration** section:

11-1. Log in with the same username and password for IRIS. User:`_system` Password: `SYS`

11-2. Select **Server Configuration** and then click the **Add Endpoint** button.

Enter these configurations: Configure the following Settings.

Property Name	Value
CORE FHIR package	hl7.fhir.r4.core@4.0.1
URL	/csp/healthshare/fhirdemo/fhir/uds
Additional packages	hl7.fhir.us.core@3.1.0
Interactions Strategy Class	HS.FHIRServer.Storage.Json.Interactions.Strategy
Storage	*Keep all default values*

Click **Add**. It will take a few minutes to build the endpoint. You can leave this screen and return to VSCode while the endpoint builds.

12. **Create the data iput directory in VS Code**

Typically, the file system in the Docker container is completely separate from the folders and files you have in the project folder. However, the container has been set up with a bind mount that connects the `iris-container/data/durable` folder in the FHIR-UDS-TRAINING project with the `/irisdata` folder in the IRIS instance.

12-1. In VSCode, right click on the `iris-continer/data/durable` folder and select `Add Folder`. Name the folder `module7-exercise1-inbound` in order to match what you configured in the Business Service Settings in the IRIS production.

Your folders will look like this (Don't worry if the top-level name is more detailed than `FHIR-UDS-TRAINING`):

13. **Check the Production:** Return to your System Management Portal. If you are looking at the "FHIR Server" screen, you can click on the profile icon for the `_system` user in the right corner. Once you click on the icon, select **Management Portal** to retrn **Home**.

Go to **Home -> Interoperability -> Select FHIRDEMO -> Configuration -> Production**.

Click on the **Messages** tab. You should see the available message traces. Click on the link under **Header** to trace the activity.

If you see errors, read the error messages, doouble check settings, and try to fix things so you get a complete message trace (see below). If you ever need to re-run the message, you can re-send from the **Message Viewer** or drop the file again.

> Solution: The comleted classes are located in the Module 7 Solutions Folder. This contains the production configuration, but does not contain the FHIR Server configuration.