HL7 FHIR

HL7 International

FHIR Intermediate Course

MODULE 2: FHIR CLIENTS

Assignments

Language: English

V1.0

Course Overview

Module I: Implementation Guides

Most relevant FHIR Implementation Guides: Argonaut & IPS
Argonaut Development and Roadmap
Argonaut Data Query IG: Scope, Use Cases
Argonaut Provider Directory IG: Scope, Use Cases
IPS FHIR IG: Scope, Use Cases

Module II: FHIR Clients

General Guidelines for FHIR Clients
FHIR Clients in JavaScript / C# / Java [1 - Elective]

Module III: FHIR Facades

Why use FHIR Server Facade: your system on FHIR
Specific FHIR Servers (FHIR Facade)
Facade Use Case / Scenarios
Facade Architecture / Patterns
Where to put the FHIR Facade
System Integration / Integration Engine / Bus / Messaging
Facade in Java / Node.JS [1 - Elective]

Module IV: FHIR Applications

Smart-On-FHIR
CDS-Hooks
Integration with Smart-On-FHIR / CDS-Hooks [1 - Elective]

Table of Contents

Goals and Options

In this week will have formal “programming” assignments, so the assignments in this unit are mostly “Build your Own”. If you want to review code, just review the code described below, in the “Where to Look” section.

We will ask you to create some functions to retrieve information stored in a FHIR Server, using the Java, C# or JS FHIR client libraries.

You will consume and create Argonaut and IPS conformant resources, and search FHIR servers for patient demographics, provider information and terminology

Platform

Platform	Description
P01	Java
P02	C#
P03	node.js

Assignment Levels

L01	Fetch Demographic data	20 points
L02	Process Extensions	10 points
L03	Fetch and Process Clinical Data	30 points
L04	Create New Resources	30 points
L05	Use Terminology Services	10 points

Total grade for each platform is 100 points
You need to obtain at least 60 points in order to complete this unit.
You can try more than one platform and/or track if you are able to, we will grade you with the maximum grade you obtained (up to 100).

Methodology

You will be asked to write **ONE** function for each assignment (with the exception of L03-3 IPS Clinical data where you will write two functions)
This function will always return a single string, with the required information or resource, or a specific error message.
Notes:

- We've chosen returning a string to make everything easier.
- In order to give minimum structure to the returned string we use **"|"** as field delimiter and **\n** as record delimiter, when you are required to return several records. *Think about a markdown table, but without headers.*

You can run your functions against any FHIR reference server from
https://wiki.hl7.org/Publicly_Available_FHIR_Servers_for_testing

We include several tests for each function
If you are not familiar with **Test Driven Development (TDD),** please read
https://en.wikipedia.org/wiki/Test-driven_development

Remember: Your assignment will be automatically graded based on the tests, but one of our tutors will review your code to make suggestions or review the automatic grade.

All non-demo functions in our projects are **stubs**: they just return an empty string.
Thus, all tests will fail (with the exception of the demo tests)

Example (node.js)

```
const Client = require('fhir-kit-client');
module.exports={GetPatientPhoneAndEmail};
async function
GetPatientPhoneAndEmail(server,patientidentifiersystem,patientiden
tifiervalue
    )
    {
        var aux="";
        return aux;
    }
```

After you are happy with the test results, you need to submit your test report and your code to the course site for grading and review (see "How to Submit")

Preconditions

If you run this full Postman Collection on any reference server, all the required information will be uploaded or updated in the server, as needed.
https://www.getpostman.com/collections/c9ccca7bce498a4d78e0

We recommend using our own server or Aegis' because we periodically run the Postman collection there (you don't need to), and they can validate US Core R4 resources.

http://wildfhir4.aegis.net/fhir4-0-0

http://fhir.hl7fundamentals.org/r4

If you want to run your tests against a local server, you can either

- store the resources in a command line FHIR HAPI server, see
  https://hapifhir.io/hapi-fhir/docs/tools/hapi_fhir_cli.html

- use a docker based FHIR HAPI server,
  see:
  https://github.com/hapifhir/hapi-fhir-jpaserver-starter

In any case, if you are starting a new server, remember to

- run the Postman collection before you begin testing and developing functions.
- Install the US-CORE validation package for validation of US Core resources.

Configuration

All projects have a **config** class. In this class you need to configure your name and course id (for the submission), and also you can configure which FHIR terminology and resource servers to use for testing/running the functions
Remember to change the configuration as needed.

Example (node.js)

```
const ServerEndpoint = ()=> {
    return "http://wildfhir4.aegis.net/fhir4-0-0";
}
const TerminologyServerEndpoint = () => {
    return "https://snowstorm.ihtsdotools.org/fhir";
}
const AssignmentSubmissionFHIRServer = () => {
    return "http://fhir.hl7fundamentals.org/r4";
}
const StudentId= () => {
    return "kaminker.diego@gmail.com";
}
const StudentName=()=>{
    return "Diego Kaminker";
}
const PatientIdentifierSystem=()=>{
    return "http://fhirintermediate.org/patient_id";
}
```

General Introduction

Disclaimer: The code included for these assignments is not supposed to be production-ready and of course is not ready for storing or retrieving information about actual patients. It's only used to illustrate FHIR concepts and techniques and make you think on possible strategies and implementation of the methods described through this unit.

For this assignment we are providing you with very small console projects or libraries aimed to test your functions and submit your results to our course site.

- Java (Plain Java Function and Testing Packages)
- .NET (.NET Core 3.1 Console App+Testing Library+Function Library)
- Node.JS (Modules + Jasmine Tests - specs)

These projects are located in the FHIR Clients course block.

You can edit them using your favorite editor/IDE (we used Visual Studio Code 3 for ASP.NET and Node.JS, and IntelliJ Idea Community Edition for Java)

Some of the extensions we have installed in our **Visual Studio Code for C#**:

- .NET Core Test Explorer
- Omnisharp C#

If you use **Visual Studio Code for Java**, we recommend installing:
- Java Extension Pack
- Java Test Runner
- Maven for Java
- Project Manager for Java

If you work in **Visual Studio Code with node.js:**
- Jasmine Test Explorer

We tried to use the bare minimum dependencies, to let you develop and test the FHIR functions without worries about learning OTHER stuff too.

Node.Js: Only the FHIR Client Library (fhir-kit-client), Axios (when the library is not enough), and jasmine to enable testing.

.Net Core: Only the FHIR Client Library – now called 'The Fire.Ly FHIR Client'- (Hl7.Fhir.R4)

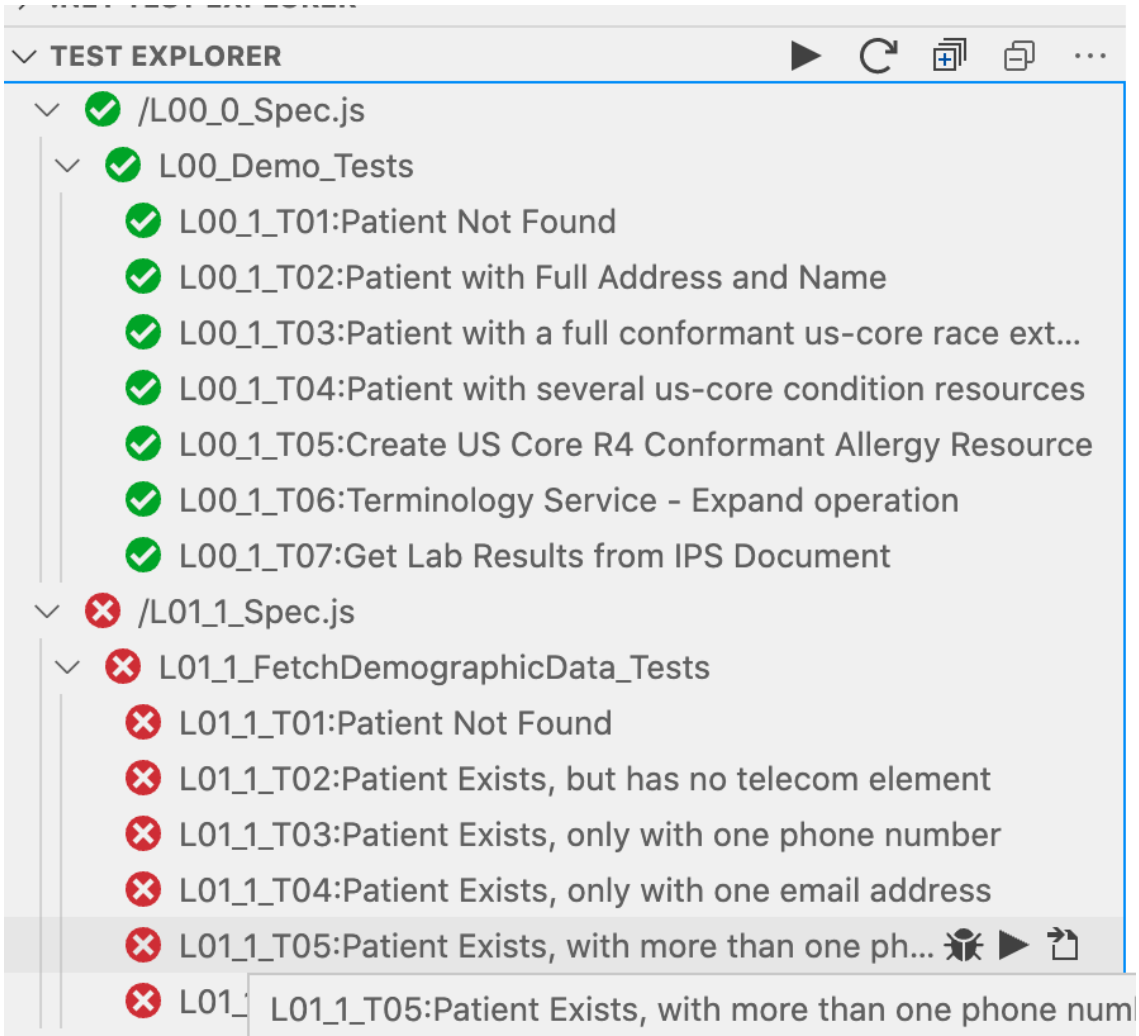
Java: ca.uhn.hapi.fhir (FHIR HAPI Client) and org.junit.jupiter (testing)

Demo

All the projects include:

- **Demo library (class / module),** with functions already implemented. These functions are similar to what you are supposed to create for our assignments.
- **Demo test (class / module)**, containing one test for each implemented function in the Demo class

Example (node.js/Jasmine)



Where to Look

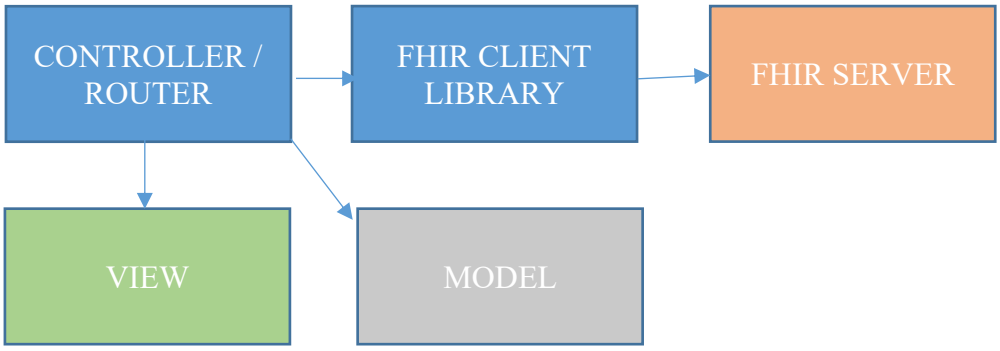
We recommend you the following learning path:

- Review the **Reading Materials** for an introduction to FHIR Clients
- Spend some time with our **CodeLabs** for the language you want to work with, to know how to write code using the libraries.
- **Inspect** the **DEMO class** and tests for actual code doing something similar to what we are asking you to do.
  Remember that you are free to reuse/copy/edit this code for your assignment submission functions.
  But…If you think that the code/style in the demos is "not your favorite" for any reason…just stay away from it and write your own from scratch!

- These are bare bones console applications; you will not be able to see the results of the function calls through a page or form.

- You can use any debugging technique you've mastered for your IDE/language of choice.

- The functions you are creating would pertain to the FHIR CLIENT LIBRARY / CONTROLLER in a web application architecture



Since these are simple projects, you only need to focus on the modules/functions you need to develop, test and submit.

Where to Look – Java

a) How to read this table: In order to fulfill assignment **[Assn Id]** you need to transform function **[Function]** in **[Java Class File]** from just a stub to a real, FHIR-enabled function; tests for this function are in **[Java Tests Class File]**)
b) Java Package for the Classes: **main / java / com / fhirintcourse / u2**
c) Java Package for the Tests: **test / java / com / fhirintcourse / u2**
d) Configuration: The **config.java** class is in the main package. Remember to change your id and server preferences
e) Submission: The **CreateSubmission.java** class is in the main package. You need to run this class to create the submission

Function	AssnId	Java Class File	Java Tests Class File
GetPatientPhoneAndEmail	L01-1	L01_1_FetchDemographics.java	L01_1_Tests.java
GetDemographicComparison	L01-2	L01_2_CompareDemographics.java	L01_2_Tests.java
GetProvidersNearCity	L01-3	L01_3_GetProvidersNearPatient.java	L01_3_Tests.java
GetEthnicity	L02-1	L02_1_FetchEthnicity.java	L02_1_Tests.java
GetImmunizations	L03-1	L03_1_FetchImmunization.java	L03_1_Tests.java
GetMedications	L03-2	L03_2_FetchMedication.java	L03_2_Tests.java
GetIPSImmunizations / GetIPSMedications	L03-3	L03_3_FetchIPSClinicalData.java	L03_3_Tests.java
CreateUSCoreR4LabObservation	L04-1	L04_1_CreateUSCoreObservation.java	L04_1_Tests.java
CreateUSCoreR4Immunization	L04-2	L04_2_CreateUSCoreImmunization.java	L04_2_Tests.java
ExpandValueSetForCombo	L05-1	L05_1_TerminologyService.java	L05_1_Tests.java

Where to Look – C#/.NET Core

For C#/.NET core, there are 3 projects.
You may want to work with all of them in the same workspace (FI_U2):

- **library (fi_u2_lib.csproj):** with all the functions you need to implement
- **tests (fi_u2_tests.csproj):** with the xUnit tests for each function
- **submission (submission.csproj):** run this project to create your submission resource file

a) How to read this table: In order to fulfill assignment **[Assn Id]** you need to transform function **[Function]** in **[.NET .cs Class File]** from just a stub to a real, FHIR-enabled function; tests for this function are in **[.NET .cs XUnit Tests File]**)
b) .NET Project for the Classes: **fi_us_lib**
c) .NET Project for the Tests: **test / java / com / fhirintcourse / u2**
d) Configuration: There is one **config.cs** class for tests and another for submission. Remember to reflect your changes in the submission config.cs file BEFORE submitting.
e) Submission: The **program.cs** class in the submission.csproj project creates the submission. You need to run this project to create the submission

Function	AssnId	.NET .cs File	.NET .cs XUnit Tests File
GetPatientPhoneAndEmail	L01-1	L01_1_FetchDemographics.cs	L01_1_tests.cs
GetDemographicComparison	L01-2	L01_2_CompareDemographics.cs	L01_2_tests.cs
GetProvidersNearCity	L01-3	L01_3_GetProvidersNearPatient.cs	L01_3_tests.cs
GetEthnicity	L02-1	L02_1_FetchEthnicity.cs	L02_1_tests.cs
GetImmunizations	L03-1	L03_1_FetchImmunization.cs	L03_1_tests.cs
GetMedications	L03-2	L03_2_FetchMedication.cs	L03_2_tests.cs
GetIPSImmunizations / GetIPSMedications	L03-3	L03_3_FetchIPSClinicalData.cs	L03_3_tests.cs
CreateUSCoreR4LabObservation	L04-1	L04_1_CreateUSCoreObservation.cs	L04_1_tests.cs
CreateUSCoreR4Immunization	L04-2	L04_2_CreateUSCoreImmunization.cs	L04_2_tests.cs
ExpandValueSetForCombo	L05-1	L05_1_TerminologyService.cs	L05_1_tests.cs

Where to Look – Node.js

a) How to read this table: In order to fulfill assignment **[Assn Id]** you need to transform function **[Function]** in **[node.js Class File]** from just a stub to a real, FHIR-enabled function; tests for this function are in **[node.js Jasmine _Spec File]**)
b) Folder for the Tests: **spec**
c) Configuration: There is one **config.js** file for configuration.
d) Submission: The **CreateSubmission.cs** module creates the submission. You need to run this file to create it.

Function	AssnId	Node.js Module File	Jasmine Spec Test File
GetPatientPhoneAndEmail	L01-1	L01_1_fetch_demographics.js	L01_1_spec.js
GetDemographicComparison	L01-2	L01_2_compare_demographics.js	L01_2_spec.js
GetProvidersNearCity	L01-3	L01_3_providers_near_patient.js	L01_3_spec.js
GetEthnicity	L02-1	L02_1_fetch_ethnicity.js	L02_1_spec.js
GetImmunizations	L03-1	L03_1_fetch_immunization.js	L03_1_spec.js
GetMedications	L03-2	L03_2_fetch_medication.js	L03_2_spec.js
GetIPSImmunizations / GetIPSMedications	L03-3	L03_3_get_clinical_data_from_ips.js	L03_3_spec.js
CreateUSCoreR4LabObservation	L04-1	L04_1_create_uscore_labresult.js	L04_1_spec.js
CreateUSCoreR4Immunization	L04-2	L04_2_create_uscore_immunization.cs	L04_2_spec.js
ExpandValueSetForCombo	L05-1	L05_1_expand_valueset.cs	L05_1_spec.js

Assignment L01-1: Fetch demographic data: Phone & Email

BUILD YOUR OWN

The demo project includes a function to retrieve full name and address of a patient from a FHIR server.

Customers are also requiring seeing all the **Phone** numbers and **E-Mail** addresses for the athlete.

Your job

Implement a function returning the phones and emails of a patient from a FHIR server

Function Name

`GetPatientPhoneAndEmail(server,IdentifierSystem,IdentifierValue)`
server: Endpoint
IdentifierSystem: System for the Patient Identifier
IdentifierValue: Value for the Patient Identifier

Returns

1- If the patient does not exist, the returned string will contain literally this:
`Error:Patient_Not_Found`
2- If the patient does not have any phone or mails, the returned string will contain:
`Emails:-\nPhones:-\n`
3- If the patient has any phone or mail, the returned string will contain the title 'Emails:' and all the emails delimited by , the new line separator and then the title 'Phones' and all the phone numbers delimited by ,
Each phone or email will be followed by its use enclosed in parenthesis:

Full Example

```
Emails:mymail@patientt05job.com (Work) , mymail@patientt05.com
(Home) \nPhones:+15555555555 (Work) , +16666666666 (Home) \n
```

Unit Tests

For all tests, Patient.identifier.system = http://fhirintermediate.org/patient_id

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L01_1_T01
T02	Patient Exists, but has no telecom element	identifier.value=L01_1_T02
T03	Patient Exists, only with one phone number	identifier.value=L01_1_T03
T04	Patient Exists, only with one email address	identifier.value=L01_1_T04
T05	Patient Exists, with more than one phone number and email addresses	identifier.value=L01_1_T05
T06	Patient Exists, has telecom element, but no phone or email address	identifier.value=L01_1_T06

Assignment L01-2: Fetch demographic data - Reconciliation

BUILD YOUR OWN

Some customers are angry because the information in their records does not match their demographic data in institutions
They want to compare the information in an easy way.
The information to compare is the minimal information stored locally: Family Name, Given Name, Gender, Birth Date, matching by Identifier
The color code for each comparison is matches (green) and mismatches (red).

Your job

Implement a function to reconcile basic demographic data between local information and the FHIR resource in the server

Function Name

```
GetDemographicComparison(server,IdentifierSystem,IdentifierValue,myFamily,myGiven,myGender,myBirth)
```

Server: Endpoint
IdentifierSystem: System for the Patient Identifier
IdentifierValue: Value for the Patient Identifier
my*: Family, Given, Gender and Birth to compare with

Returns

1- If the patient does not exist, the returned string will contain literally this:
`Error:Patient_Not_Found`

2- If the patient exists, the function will return four rows in a simplified table
```
{family}|(local family name)|(FHIR Resource family name)|{color}\n
{given}|(local given names delimited by space)|(FHIR Resource
given names delimited by space)|{color}\n
{gender}|(local gender code in upper case)|(FHIR Resource gender
code in upper case)|{color}\n
{birthDate}|(local birthdate YYYY-MM-DD)|(FHIR Resource birth date
YYYY-MM-DD)|{color}\n
```

{color} = {red} if the local value is different from the FHIR Resource value
{green} otherwise

Full Example
```
{family}|Douglas|Douglas|{green}/n{given}|Jamieson|Jamieson
Harris|{red}/n{gender}|MALE|MALE|{green}/n{birthDate}|1968-07-
23|1968-07-23|{green}/n
```

Unit Tests

For all tests, Patient.identifier.system = http://fhirintermediate.org/patient_id

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L01_2_T01
T02	Patient Exists, different family name	identifier.value=L01_2_T02
T03	Patient Exists, different given name	identifier.value=L01_2_T02
T04	Patient Exists, different birth date	identifier.value=L01_2_T02
T05	Patient Exists, different gender	identifier.value=L01_2_T02
T06	Patient Exists, all demographic matches	identifier.value=L01_2_T02

Assignment L01-3: Fetch demographic data – Provider Search

BUILD YOUR OWN

Our Sport application needs to search for a provider near the athlete home, as a way to enable the coach and the medical team to speed up care for a sick or wound athlete when they are at home.
Your task: extract the information you deem relevant and do a provider search.
Note: to make this easier, do not need to use 'PractitionerRole' for the search, just use Practitioner and Qualifier. In the real world, you would use PractitionerRole and Specialty (Argonaut Provider Directory)
Return a list with all the information about the matching provider (Names, Address, Telecom, Qualifier)

Provider Name	Address	Telecom	Qualifier
Frank, Michael F.	28 Peters Dr., Smithers, AK, 20992	555 666-7777	Ob/Gyn
Marenson, Alan J.	1209 Discovery Road, Smithers, AK, 20992	555 777-6666	Ob/Gyn

Your job

Implement a function to obtain the providers' address, phone and qualification for practitioners in the same city of a given patient.

Function Name

```
GetProvidersNearCity(server,IdentifierSystem,IdentifierValue)
```

Server: Endpoint
IdentifierSystem: System for the Patient Identifier
IdentifierValue: Value for the Patient Identifier

Returns

1- If the patient does not exist, the returned string will contain literally this:
`Error:Patient_Not_Found`

2-If the patient has no reported city, the returned string will contain literally this:
`Error:Patient_w/o_City`

3-If there are no providers in the patient's city, the returned string will contain literally this:
`Error:No_Provider_In_Patient_City`

4- If there is one or more providers in the patient's city, the function will return as many rows as providers in a simplified table

```
(provider_full_name)|(address lines)|(telecom)|{qualification}\n
```

Address Lines: First address, all lines concatenated with space
Provider Full Name: Family and all given concatenated with space

Full Example

```
OnlyPhysician,InTown|Phone:+402-772-7777|2000 ONE PROVIDER
DRIVE|OB/GYN\n
```

Unit Tests

For all tests, Patient.identifier.system = http://fhirintermediate.org/patient_id

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L01_3_T01
T02	Patient Exists, No City Element	identifier.value=L01_3_T02
T03	Patient Exists, No Provider in the City	identifier.value=L01_3_T03
T04	Patient Exists, One Provider in the City	identifier.value=L01_3_T04
T05	Patient Exists, More than one provider	identifier.value=L01_3_T05

Assignment L02-1: Process extensions – Ethnicity

BUILD YOUR OWN

The demo function retrieves and returns the US CORE IG FHIR R4 'race' extension.

Your task: extract the information for the 'Ethnicity' extension and show it (text, code and detail) – watch out for repetitions.

Your job

Implement a function to obtain the ethnicity codes/display and text for a given patient. The ethnicity is included using the US CORE R4 ethnicity extension as defined in http://hl7.org/fhir/us/core/StructureDefinition-us-core-ethnicity.html

Function Name

```
GetEthnicity(server,IdentifierSystem,IdentifierValue)
```

Server: Endpoint
IdentifierSystem: System for the Patient Identifier
IdentifierValue: Value for the Patient Identifier

Returns

1- If the patient does not exist, the returned string will contain literally this:
`Error:Patient_Not_Found`

2-If the patient has no ethnicity extension, the returned string will contain literally this:
`Error:No_us-core-ethnicity_Extension`

3-If the extension is included but is not conformant (for example, it does not contain a text or OMBCategory code), the returned string will contain literally this:
`Error:Non_Conformant_us-core-ethnicity_Extension`

4- If the extension could be processed, the function will return (in this order), and in a simplified table format:

a) One row for the mandatory **text** component of the extension
text|{valueString}

b) One row for the code and display extracted from the mandatory **OMBCategory** component of the extension
text|{valueCoding.code:valueCoding:display}

c) One row for the code and display extracted from each optional **Detailed** component of the extension
detail|{valueCoding.code:valueCoding:display}

Full Example

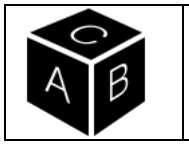
```
text|Hispanic or Latino\n
code|2135-2:Hispanic or Latino\n
detail|2184-0:Dominican\n
detail|2148-5:Mexican\n
detail|2151-9:Chicano\n
```

Unit Tests

For all tests, Patient.identifier.system = http://fhirintermediate.org/patient_id

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L02_1_T01
T02	Patient Exists, No Ethnicity Extension	identifier.value=L02_1_T02
T03	Patient Exists, No Valid Ethnicity Extension	identifier.value=L02_1_T03
T04	Patient Exists, Minimum Ethnicity Extension	identifier.value=L02_1_T04
T05	Patient Exists, Complete Ethnicity Extension	identifier.value=L02_1_T05

Assignment L03-1: Fetch and process clinical data- Immunization



BUILD YOUR OWN

The demo function allows retrieving the patient's current (US-Core conformant) Conditions.

The requirement for this assignment is to retrieve and return the patient's **immunizations**.

Check out the resource, and which elements are MUST-SUPPORT in the implementation guide and make sure you receive them.

Your job

Implement a function to obtain the immunizations record for a given patient. Immunizations are stored in the same server, and the structure is defined by the US CORE R4 profile http://hl7.org/fhir/us/core/StructureDefinition-us-core-immunization.html

Function Name

```
GetImmunizations(server,IdentifierSystem,IdentifierValue)
```

Server: Endpoint
IdentifierSystem: System for the Patient Identifier
IdentifierValue: Value for the Patient Identifier

Returns

1- If the patient does not exist, the returned string will contain literally this:
`Error:Patient_Not_Found`

2-If the patient has no immunization record, the returned string will contain literally this:
`Error:No_Immunizations`

3- If the immunization resources were processed, the function will return the records in a simplified table format (one row per immunization resource)

```
status|code:display|immunization_date\n
```

Full Example

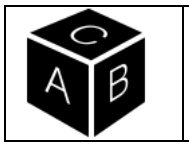
```
Completed|207:COVID-19, mRNA, LNP-S, PF, 100 mcg/0.5 mL dose|2020-01-10\n
Completed|173:cholera, BivWC|2019-10-20\n
```

Unit Tests

For all tests, Patient.identifier.system = http://fhirintermediate.org/patient_id

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L03_1_T01
T02	Patient Exists, No Immunizations	identifier.value=L03_1_T02
T03	Patient Exists, One Immunization Record	identifier.value=L03_1_T03
T04	Patient Exists, Several Immunization Records	identifier.value=L03_1_T04

Assignment L03-2: Fetch and process clinical data- Medications



BUILD YOUR OWN

The demo function allows retrieving the patient's current (US-Core conformant) Conditions.

The requirement for this assignment is to retrieve and return the patient's **medications.**

Check out the resource, and which elements are MUST-SUPPORT in the implementation guide and make sure you receive them.

Your job

Implement a function to obtain the medications records for a given patient. Medications are stored in the same server, and the structure is defined by the US CORE R4 profile http://hl7.org/fhir/us/core/StructureDefinition-us-core-medicationrequest.html

Function Name

```
GetMedications(server,IdentifierSystem,IdentifierValue)
```

Server: Endpoint
IdentifierSystem: System for the Patient Identifier
IdentifierValue: Value for the Patient Identifier

Returns

1- If the patient does not exist, the returned string will contain literally this:
`Error:Patient_Not_Found`

```
2-If the patient has no immunization record, the returned string will contain literally this:
Error:No_Medications
```

3- If the records could be processed, the function will return the records in a simplified table format (one row per medicationrequest resource)
`status|intent|authored_on|code:display|requester\n`

a. For Medication code:display we use the inline CodeableConcept, so the medication code/display can be extracted from MedicationRequest.medicationCodeableConcept.coding

   [In real life, there are three possible strategies: inline, contained, bundle]

b. For Requester, you can just extract the MedicationRequest.requester.display
   [In real life, you may have to resolve the Requester reference to practitioner]

Full Example

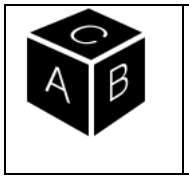
```
Active|Order|2021-01-05|582620:Nizatidine 15 MG/ML Oral Solution
[Axid]|Mary Requesting, MD
Active|Order|2021-01-05|198436:Acetaminophen 325 MG Oral
Capsule|Mary Requesting, MD
```

Unit Tests

For all tests, Patient.identifier.system = `http://fhirintermediate.org/patient_id`

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L03_2_T01
T02	Patient Exists, No Medications	identifier.value=L03_2_T02
T03	Patient Exists, One Medication Record	identifier.value=L03_2_T03
T04	Patient Exists, Several Medication Records	identifier.value=L03_2_T04

Assignment L03-3: Fetch and process clinical data- IPS



BUILD YOUR OWN

A hospital group in Italy is ready to return IPS documents for our app.
Your task: retrieve medications and immunizations info from an IPS Document.

How:

1) Retrieve the document using a Composition search by patient id and document code (using the LOINC code for IPS

2) Retrieve the medication and immunization data from the document.

You can review the Demo functions and see how we retrieve a document and extract all Laboratory results from an IPS document for a given patient.

Your job

Implement two functions returning the Medications and Immunizations for a Patient, extracted from an IPS document stored in a FHIR Server

For the MedicationStatement IPS Resource see:
http://hl7.org/fhir/uv/ips/StructureDefinition-MedicationStatement-uv-ips.html

For the Immunization IPS Resource see:
http://hl7.org/fhir/uv/ips/StructureDefinition-Immunization-uv-ips.html

To search for the IPS document use the 'composition' search parameter with the attributes patient and code

```
Bundle?composition.patient={Patient_Id}&composition.code={loinc for IPS}
```

Function Names

```
GetIPSMedications(server,IdentifierSystem,IdentifierValue)
GetIPSImmunizations(server,IdentifierSystem,IdentifierValue)
```

server: Endpoint
IdentifierSystem: System for the Patient Identifier
IdentifierValue: Value for the Patient Identifier

Returns

1- If the patient does not exist, the returned string will contain literally this:
`Error:Patient_Not_Found`

2- If there is no IPS for the patient stored in the server, the returned string will contain literally this:
`Error:No_IPS`

3- If there are no Medications or no Immunizations the IPS IG mandates the inclusion of a special code/display in the Entry.
If not even this entry is included, the returned string will contain literally one of these, depending on the function:
`Error:IPS_NO_Immunizations`

`Error:IPS_NO_Medications`

4- If there are Medications or Immunizations in the IPS IG (or the special code/display for no info was included) then the string will return a simplified table with the following information:

IPS Immunizations:
`i_status+|imm_date|i_code:i_display`

IPS Medications:
`m_status+|m_date_period|m_code:m_display`

a. For Medication code:display we use both the **inline CodeableConcept**, so the medication code/display can be extracted from MedicationRequest.medicationCodeableConcept.coding and the **Medication resource in the bundle**, as defined by the IPS Spec

b. For Medication dates in IPS there are two options, just the date or a period. So you need to support either method.

Full Example

IPS Medication

```
Active|2015-03|108774000:Product containing anastrozole (medicinal
product)\n
Active|2016-01|412588001:Cimicifuga racemosa extract (substance)\n
```

IPS Immunization

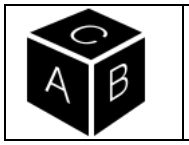
```
Completed|2020-01-08|10292158:influenza, injectable,
quadrivalent\n
```

Unit Tests

For all tests, Patient.identifier.system = `http://fhirintermediate.org/patient_id`

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L03_3_T01
T02	Patient Exists, No IPS	identifier.value=L03_3_T02
T03	Patient Exists, Medication Record(s) in IPS	identifier.value=L03_3_T03
T04	Patient Exists, No Medication Records in IPS	identifier.value=L03_3_T04
T05	Patient Exists, Immunization Record(s) in IPS	identifier.value=L03_3_T04
T06	Patient Exists, No Immunization Records in IPS	identifier.value=L03_3_T03

Assignment L04-2: Create a new resource – US Core Laboratory Observation



BUILD YOUR OWN

Create a function capable of creating a Laboratory Observation resource conformant to the Argonaut US CORE R4 IG.
We will validate your resource against the IG, so you should, too.

Your job

Implement one function returning a Laboratory Observation resource conformant with the US CORE FHIR 4 specification given a specific set of input parameters with the attributes of the laboratory result.
The resource should be in JSON format.
For the Laboratory Observation US FHIR Core R4 Resource see:
http://hl7.org/fhir/us/core/StructureDefinition-us-core-observation-lab.html

Function Name

```
CreateUSCoreR4LabObservation
(ObservationStatusCode,
 ObservationDateTime,
 ObservationLOINCCode,
 PatientIdentifierSystem,
 PatientIdentifierValue,
 ResultType (Numeric, Coding),
 NumericResultValue,
 NumericResultUCUMUnit,
 CodedResultSNOMEDCode
)
```

Returns

If the patient does not exist, a literal string:

```
Error:Patient_Not_Found
```

If the patient exists in the server, a JSON string with the FHIR R4 Conformant Laboratory Observation Resource

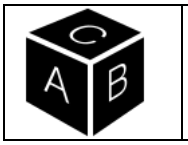
Unit Tests

For all tests:

Patient.identifier.system: http://fhirintermediate.org/patient_id
Patient.identifier.value:
L04_1_T01 (not found)
L04_1_T02 (found)
Result Date: **06/07/2021 10:45:33 AM**

Test #	Code	Value	Reference Range
T01	-	-	(Patient Not Found)
T02	1975-2: Bilirubin, Serum	8.6 mg/dL	2.0 mg/dL - 7.0 mg/dL for Normal Range
T03	5778-6: Color of Urine	Yellow (Snomed CT: 371244009)	Yellow

Assignment L04-2: Create a new resource - US Core Immunization



BUILD YOUR OWN

Create a function capable of creating an Immunization resource conformant to the Argonaut US CORE R4 IG.

We will validate your resource against the IG, so you should, too.

Your job

Implement one function returning an Immunization resource conformant with the US CORE FHIR 4 specification given a specific set of input parameters with the attributes of the immunization record.
The resource should be in JSON format.
For the Immunization US FHIR Core R4 Resource see:
http://hl7.org/fhir/us/core/StructureDefinition-us-core-immunization.html

Function Name

```
CreateUSCoreR4Immunization(
                      server,
                      IdentifierSystem,
                      IdentifierValue,
                      ImmunizationStatusCode,
                      ImmunizationDateTime,
                      ProductCVXCode,
                      ProductCVXDisplay,
                      ReasonCode);
```

Returns

1. If the patient does not exist, a literal string:
   `Error:Patient_Not_Found`

2. If the patient exists in the server, a JSON string with the FHIR R4 Conformant Immunization Resource

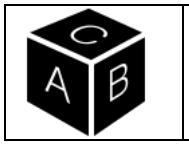
Unit Tests

For all tests:

Patient.identifier.system: http://fhirintermediate.org/patient_id

#	Test	Patient/Data
T01	Patient Does Not Exist	identifier.value=L04_2_T01
T02	Immunization Completed	patient.identifier.value=L04_2_T02 StatusCode=completed DateTime=2021-10-25 Product CVXCode=173 Display=cholera, BivWC
T03	Immunization Not-Done	patient.identifier.value=L04_2_T02 StatusCode=not-done / ReasonCode: IMMUNE DateTime=2021-10-30T10:30:00 Product CVXCode=207 / Display=COVID-19, mRNA, LNP-S, PF, 100 mcg/0.5 mL dose

Assignment L05-1: Terminology Services



BUILD YOUR OWN

The clinical team is asking for a way to populate a combo with SNOMED CT symptoms and conditions using a FHIR Server ($expand function).

Create a class that enable this from a FHIR Terminology Server

Your job

Implement a function to retrieve all the concepts and codes for a specific SNOMED CT expression, filtered by an optional string.

To search for the expansion ValueSet use the $expand operation with the [url] and [filter] parameters.
ValueSet/$operation?url=[url]&filter=[filter]

We recommend using the official SnowStorm Terminology Server from SNOMED International: https://snowstorm.ihtsdotools.org/fhir

You can use any other FHIR-enabled SNOMED CT server.

Function Name

`ExpandValueSetForCombo(server,url,filter)`
server: Endpoint
url: ValueSet and FHIR SNOMED CT Valid Expression
filter: Additional filter for the expansion

Returns

1- If the expression/valueset+filter does not return any concept, the returned string will contain literally this:
`Error:ValueSet_Filter_Not_Found`

4- If there are one or more matching concepts returned, then the string will return a simplified table with the following information:

`code|display\n`

Unit Tests

For all tests:

Terminology Server: https://snowstorm.ihtsdotools.org/fhir

You can change it, but always use a SNOMED CT compatible FHIR Server with terminology capabilities (Example: implementing $expand)

Test #	Parameters	Returns
T01	url: http://snomed.info/sct?fhir_vs=isa/73211009 filter: diaxetes	`ValueSet_Filter_Not_Found`
T02	url: http://snomed.info/sct?fhir_vs=isa/73211009 filter: drug-induced diabetes	`5368009|Drug-induced diabetes mellitus`

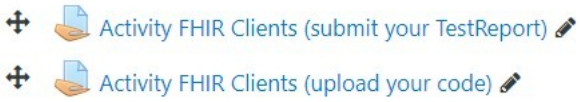
How to Submit

All the projects have a special class/package or module called **CreateSubmission**
Note: You need to RUN this project, it's not a library or test (i.e.: it has a 'main')

This function runs all your functions, and packages the results along with your student name and id in a FHIR TestReport resource (https://www.hl7.org/fhir/testreport.html)

This resource will be stored in your local drive (in the project folder) with the filename:
`FHIR_INTERMEDIATE_U2_SUBMISSION_<<YOUR_STUDENT_ID>><<DATE_TIME>>.JSON`

You need to submit **this resource** along with **your code in a zip file**, to our course site.



Your assignment will be graded automatically by our site, but a tutor will need to review the code to make sure that your functions are really doing the FHIR work and maybe make suggestions.

This is an example of our course site feedback:

Important:
The automatic grade MAY BE overridden by your tutor's code review.

References

Information Specific for .NET (C#)

We use xUnit and .NET Test Explorer for our C# tests in VS Code.

More about xUnit at https://xunit.net/

Information Specific for Node.JS

We use Jasmine and the Test Explorer for our Node.JS tests in VS Code.

More about Jasmine at https://jasmine.github.io/

A nice video about testing in Node.js with Jasmine can be found here
https://www.youtube.com/watch?v=aBYwNqiWYmU

Information Specific for Java

We use JUnit 5 in VS Code, IntelliJ and Eclipse

More about JUnit 5 at https://junit.org/junit5/docs/current/user-guide/#overview-getting-started

If we want to understand more about JUnit5, see this video
https://www.youtube.com/watch?v=we3zJE3hlWE