Building FHIR Applications with InterSystems API Manager

Exercise Outline

Exercise Guide

Building FHIR Applications with InterSystems API Manager

You have been tasked with building an application that displays data from patients who have tested positive for COVID-19. This application will provide essential data to physicians caring for these patients. Specifically, you will need to retrieve and chart a patient's lactate dehydrogenase levels, an enzyme in the blood that may predict the severity and mortality of COVID-19. Through this exercise, you will learn how to use InterSystems API Manager to view APIs so that you can write a FHIR® application that queries observations of this enzyme from a FHIR server over REST. InterSystems IRIS for Health™ provides a base FHIR server implementation with a FHIR resource repository, and for this exercise we have pre-configured a FHIR R4 server on an IRIS for Health instance and pre-populated its FHIR resource repository with sample patient data.

By the end of this exercise, you will be able to:

- Use InterSystems API Manager to view APIs and test FHIR requests
- Make a FHIR Request to InterSystems IRIS for Health's FHIR server using a REST client
- Connect a FHIR application to InterSystems IRIS for Health's FHIR server
- Monitor API calls by viewing statistics in the API Manager Developer Portal

Try It

You know that data is stored in InterSystems IRIS for Health, but what APIs can you call in your client-side application? Using InterSystems API Manager, you can browse available APIs and test them directly in the browser, then transition to your REST client to construct queries before embedding them in your application. Furthermore, these calls then all go through API Manager, a tool that allows you to track requests and errors, embed security into your applications, and much more. With API Manager, you have everything you need to properly manage your FHIR APIs.

Task 1 – View and Test FHIR APIs

In this section, you will explore the API Manager Developer Portal from the perspective of a client-side developer and view a list of available FHIR APIs. You will also find related, easy-to-use documentation that equips you to later build queries for your FHIR application.

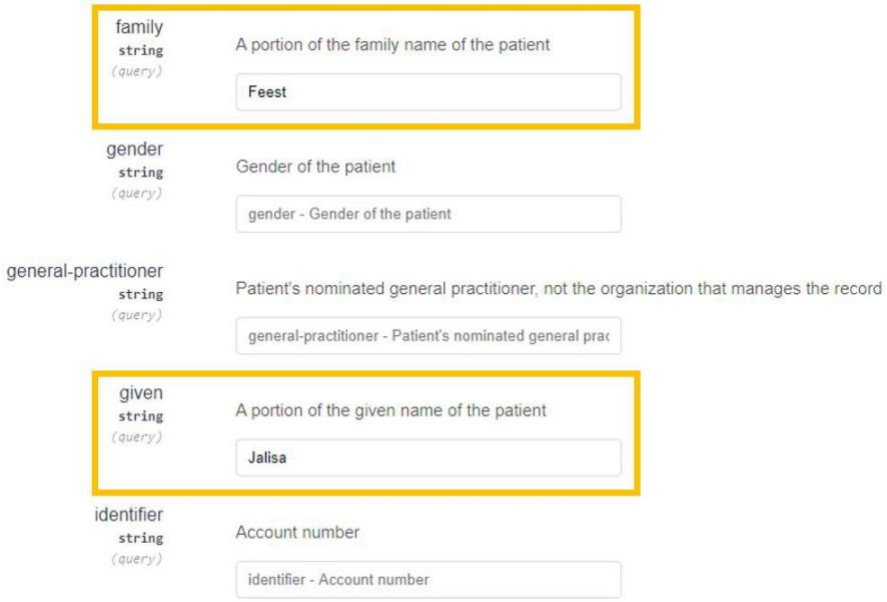
1. First, modify the way the VM displays within your browser by clicking  (Fit to Window). This will be beneficial throughout the exercise so you best see all the contents on your screen.

2. Open a Chrome browser. Navigate to and click the **IAM API Catalog** bookmark at the top, which brings you to the API catalog for the `FHIRServerWorkspace`.

3. InterSystems API Manager provides the ability to view REST specifications and test REST calls directly in the interface. Several unofficial OpenAPI-based artifacts have been pre-loaded for you, based on the FHIR R4 specification. Click **FHIR R4 Patient Resource** specification. Here, you

can view different operations supported by this API, see examples of how to call the route, and test the routes.

4. Scroll down to the route GET /Patient. Click **Test Endpoint**. Enter the family name *Feest* and given name *Jalisa*. This is a sample COVID-19 patient whose data we will use to test our endpoint. Click **Make API Call**.



5. Scroll all the way to the right and up a bit to see the response. In the Response body, you should see the information for Jalisa Feest. Note: Click inside the Response body to scroll contents.

You have just explored the IAM Dev Portal, viewed unofficial OpenAPI-based FHIR specs, and made an API call to find a Patient resource. Recall you can view the full official FHIR specification online at http://hl7.org/fhir.

Task 2 – Use a REST Client to Make FHIR Requests

Once you are familiar with the API calls available, you will likely use a REST client to test your queries before building them into your application. This simplifies development and lets you quickly adjust your queries to get the data you need. Recall that you first need to retrieve all patients diagnosed with COVID-19, indicated in a patient's Condition resource as SNOMED 840539006. Then, you will retrieve a particular patient's Observation values for lactate dehydrogenase (LOINC 14804-9), an enzyme required in the process of turning sugar into energy that is associated with the escalation of COVID-19 when elevated. These two queries will eventually be included in the FHIR application.

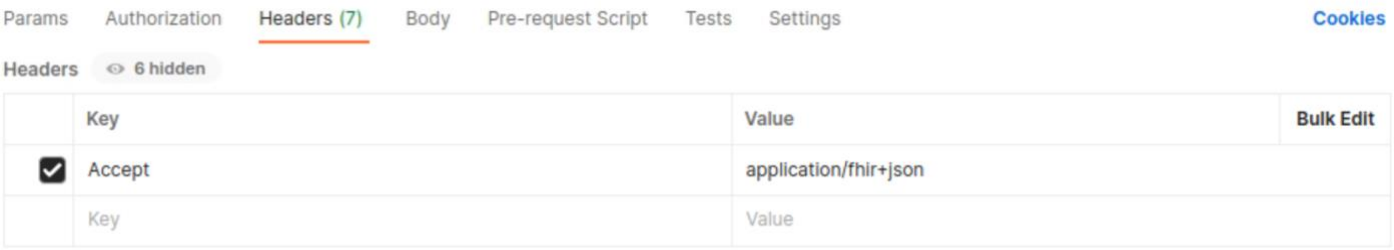
1. Within your VM, open Postman using the Postman icon in the taskbar: 

2. Start with a simple FHIR request to verify you can successfully make requests that go through API Manager to retrieve data from InterSystems IRIS for Health. Get data for patient Jalisa Feest and confirm that her ID is *4694* by completing the fields below with the following values:

   Headers:

   Key: *Accept*

   Value: `application/fhir+json` (**Note**: This value does not appear in the drop-down, so you need to type it in manually. You can use CommandsToCopy.txt on the Desktop to copy and paste this if you have trouble copying from this guide.)

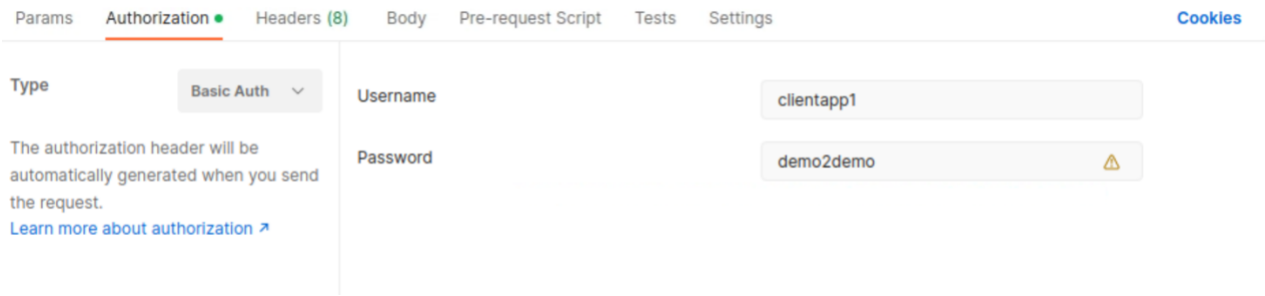


Authorization:

Select authorization type **Basic Auth**, and enter the credentials:

Username: *clientapp1*

Password: *demo2demo*



HTTP Request:

Method: `GET`

Request URL:
http://*localhost:8000*/r4/Patient?family=Feest&given=Jalisa

Click **SEND** to make the request.

3. Next, you will make a FHIR request for use in the FHIR application. Get a list of COVID-19 patients (SNOMED 840539006) above age 65. Notice that 3 patients in this database have a COVID-19 diagnosis. (Note: le1955-01-01 represents less than or equal to January 1st, 1955.)

HTTP Request:

Method: `GET`

Request URL:
`http://localhost:8000/r4/Patient?_has:Condition:patient:code=840539006&birthdate=le1955-01-01`

4. Health professionals are tracking many markers in their efforts to predict how a patient's COVID-19 symptoms may progress. In our scenario, clinicians would like to chart lactate dehydrogenase (LOINC 14804-9) level over time. To assist them, you will retrieve all Observation resources with this LOINC code, given a patient ID. Returning to patient Jalisa Feest, let's request Observation resources in her record using her ID, which was retrieved in step 2.

HTTP Request:

Method: `GET`

Request URL:
`http://localhost:8000/r4/Observation?patient=4694&code=14804-9`

You have now made FHIR requests for a Patient resource and related Observation resources. Next, we will learn how to send requests from a FHIR application.

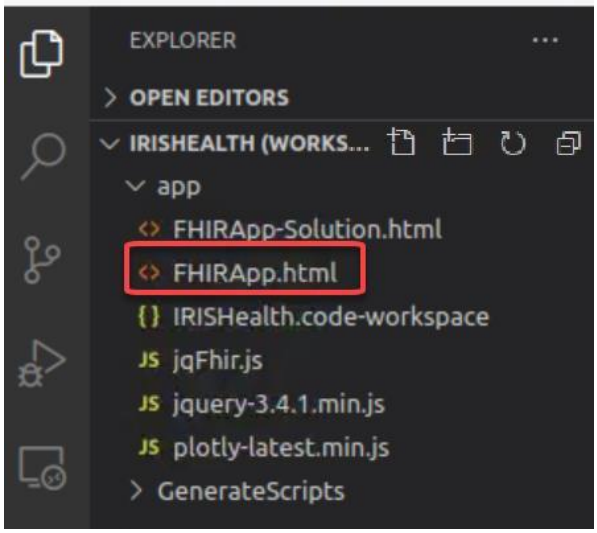
Task 3 – Build Your Application

Now it's time to embed these queries into the FHIR application. In this section, you'll modify an existing HTML application that has been mostly written by adding in the FHIR queries that you built in the last section. As mentioned previously, this application will first show a list of all patients who have been diagnosed with COVID-19. Then, given a patient ID, it will display a chart with lactate dehydrogenase test results.

1. First, you'll need to open the VSCode editor to edit this application. Open the editor in your VM using the VSCode icon in the task bar: 

2. To open the project workspace, click **File** > **Open Workspace from File…** and open **\home\student\studentFiles\IRIScontainer\durable\app\IRISHealth.code-workspace** .

3. In the Explorer pane, open the main HTML file that contains the application by navigating to **App** > **FHIRApp.html**:



4. Find the assignment for BaseURL (see line 47). Notice the URL that points to the server for the Ubuntu VM containing InterSystems API Manager and IRIS for Health. Take note of the IP and port that are used; these should be the same values used in the REST client in the previous section. You'll notice these values refer to API Manager because we are first sending requests there, and API Manager then forwards the requests to IRIS for Health's FHIR server.

5. Scroll down to the `client.search` function and uncomment the lines specified in the comment. It should look like this:

```
client.search({
    type:'Patient',
    query: {
    // Uncomment the following lines
       '_has:Condition:patient:code':'840539006',
       _sort: 'birthdate',
       birthdate: 'le1955-01-01',
       _count: 5
    }
```

   Don't forget to save the file.

6. Open the **FHIR App** web page using the bookmark in the browser. Notice in the URL that this is the page you were just editing, and it displays a list of five patients. Supply the patient ID for Jalisa Feest (4694) to see her lactate dehydrogenase observation values charted. How is Jalisa doing?

7. (Optional) Challenge: Oxygen saturation is an important factor in diagnosing COVID-19. Modify `FHIRApp.html` to get Jalisa's observation values of oxygen saturation (LOINC 59408-5) charted. Note that the value range for oxygen saturation is 75–100 mmHg. See `FHIRApp-Solution.html` for the answer. Note: To do this, edit the file in VSCode, save the file, and refresh the page in the browser.

Task 4 – Manage API Requests

Now that you have an application that allows clinicians to monitor lactate dehydrogenase for COVID-19 patients, an InterSystems IRIS developer wants to monitor all calls to InterSystems IRIS for Health to ensure the reliability and security of the applications. You know that more client-side developers will eventually be building several additional FHIR applications that aid research facilities, doctor's offices, pharmacies, and more, so setting this up now will be critical. Through InterSystems API Manager, you can view statistics about calls that have been made (including errors), build in load balancing to restrict certain applications from making too many calls, and embed security directly into your application. In this section, you will use API Manager to view successful and erroneous calls and resolve any issues quickly.

1. Let's first generate several test calls to see some interesting metrics. These calls will simulate other facilities making requests to InterSystems IRIS for Health and retrieving COVID-19 data.

    a. Open VSCode and open a Terminal by selecting **Terminal** in the main menu > **New Terminal** > **GenerateScripts**.
    b. Run the following command (you can let it run or type **Ctrl**+**C** to stop):
    `./pharmacy.sh`
    c. Open another Terminal session and run:
    `./research.sh`
    d. Open one last Terminal session and run:
    `./drsoffice.sh`

    Note: To open an additional Terminal session, select **Terminal** in the main menu > **New Terminal** > **GenerateScripts**.

2. Next, open the **IAM FHIRServerWorkspace** by using the bookmark in the browser.

3. Notice there are several successful calls, but a few with errors. Let's explore why these errors occurred and figure out how to fix them. Scroll down to view the list of consumers and click each consumer's name to figure out which consumer is generating the most errors. Is it the pharmacy, doctor's office, or research facility? You can now go back to the developer of that application and help them fix the error!

You have now used the IAM Developer Portal to manage API calls, identifying an error in an application and proactively helping to fix the error. With InterSystems API Manager, you have the tools to properly manage your APIs and ensure the reliability of all client-side applications.