Interoperability Land™

User Guide

Version v2 | March 2021



Table of Contents

[Interoperability Land™ Overview 2](#_Toc66113795)

[Introduction to FHIR 3](#_Toc66113796)

[Fast Healthcare Interoperability Resources (FHIR) 3](#_Toc66113797)

[Why FHIR Is Important 3](#_Toc66113798)

[How FHIR Works and Understanding FHIR Resources 3](#_Toc66113799)

[Other FHIR Initiatives 3](#_Toc66113800)

[Getting Started with IOL 4](#_Toc66113801)

[Getting Access 4](#_Toc66113802)

[Logging On 4](#_Toc66113803)

[Exploring the Welcome Page and Site Navigation 5](#_Toc66113804)

[Welcome 5](#_Toc66113805)

[Navigation Menu 5](#_Toc66113806)

[Settings and Help Menu 5](#_Toc66113807)

[Dashboard 5](#_Toc66113808)

[Exploring the Sandbox 6](#_Toc66113809)

[Accessing your Sandbox 6](#_Toc66113810)

[Sandbox Tab Menu 6](#_Toc66113811)

[Exploring the PITs 7](#_Toc66113812)

[InterOp PIT Tiles 7](#_Toc66113813)

[PIT Tab Menu 7](#_Toc66113814)

[Connecting a SMART on FHIR App 8](#_Toc66113815)

[Completing a Query 12](#_Toc66113816)

[Accessing the PIT and HAPI FHIR API 12](#_Toc66113817)

[Navigating the Resources 12](#_Toc66113818)

[Using the Basic Query 13](#_Toc66113819)

[Using the Advanced Query 15](#_Toc66113820)

[Using the HAPI FHIR Interface Query 16](#_Toc66113821)

[Using the Raw Message Information Query 18](#_Toc66113822)

[Creating Medication Reconciliation Queries 18](#_Toc66113823)

[Creating a Basic Query in Postman 20](#_Toc66113824)

[Using Postman and the RESTful API to create more complex queries 21](#_Toc66113825)

[Appendix A: Terminology 22](#_Toc66113826)

[Appendix B: Querying FHIR Resources 23](#_Toc66113827)

[Example query structure 26](#_Toc66113828)

[Advanced query formatting 27](#_Toc66113829)

[Querying for a range of values 27](#_Toc66113830)

[Data linkage 29](#_Toc66113831)

[Crafting more complex queries to answer questions 30](#_Toc66113832)

[Limitations to out-of-the-box queries 31](#_Toc66113833)

[Workarounds 31](#_Toc66113834)

All rights reserved. The contents of the this document are copyrighted by The Constellation and permission should be obtained from The Constellation prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For permission information regarding this document please mails you written request to The Constellation,

3001 Miller Rd #315, Ann Arbor, MI 48103.

Unless otherwise indicate herein, any third-party trademarks that may appear in this work are the property of their respective owners and any references to third-party trademarks, logos, or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, promotion of The Constellation products by the owners of such marks, or any relationship between the owner and The Constellation or its affiliates, authors, licensees, or distributors.

Code samples found herein are copyright 2015, University Health Network.

Modifications for InterOpathon by MiHIN/Interoperability Institute Licensed under the Apache License, Version 2.0 (the "License"); you may not use this work except in compliance with the License. You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

# Interoperability Land™ Overview

Interoperability Land (IOL) is a cloud-hosted digital collaboration platform designed to power the future of multi-organization development, integration, acceptance, and testing of innovative technologies and open standards. Leaders in healthcare information technology can use interoperability solutions needed for compliance with new federal government healthcare regulations. IOL allows organizations to learn, build, and test healthcare applications and services by using synthetic patient data that is realistic and clinically relevant with no risk of Personal Health Information (PHI) disclosure.

Two types of synthetic data are available in IOL:

* **PatientGen™** is a test data generator compliant with the HL7® Fast Healthcare Interoperability Resources (FHIR®) standard. PatientGen models a simulated healthcare network of providers, practices, and hospitals as well as a large population of patients who experience changes in their health status and mortality risks for important medical conditions and procedures. PatientGen data is a good fit for developers and IT staff who need synthetic data samples to test interoperability. PatientGen data is based on probable patient health trajectories and is not clinically authenticated.
* **Personas** are realistic synthetic representations of a patient based on qualitative and quantitative research. Each persona represents unique attitudes, conditions, and environments enabling a clear understanding of their goals, problems, and background. These user stories test specific scenarios and patient histories are authenticated by a clinician to ensure accuracy. This data is best for an audience in a practicing clinical environment.

Both types of synthetic data are designed to showcase new technology, promote standards, and accelerate interoperability.

This synthetic ecosystem allows healthcare organizations to:

* Demonstrate apps and services in an engaging and meaningful way using data visualization to reveal interoperability between systems.
* Rigorously test and certify that applications meet standards, performance, and scalability requirements.
* Collaborate with other organizations to develop and test interoperable, standards-compliant solutions.
* Host collaborative events to promote learning and standards-based technology adoption.
* Deliver higher quality applications and services to market faster.

# Introduction to FHIR

## Fast Healthcare Interoperability Resources (FHIR)

Health Level Seven International (HL7) is a not-for-profit organization that develops and standardizes the data framework for the exchange of electronic health information. One set of these standards is Fast Healthcare Interoperability Resources (FHIR).

### Why FHIR Is Important

FHIR is designed to help health information organizations quickly and easily retrieve and exchange data from EHR systems. Health IT developers use the standard to efficiently build applications for data exchange.

### How FHIR Works and Understanding FHIR Resources

FHIR frameworks are built around the concept of Resources. Resources are basic objects or modular units that can be assembled into working systems. These systems resolve clinical, administrative and infrastructure problems in healthcare.

Administrative concepts (e.g., patients, providers, organizations, and devices) as well as a variety of clinical concepts (e.g., conditions, medications, diagnostics, care plans, and claims information) are translated into FHIR Resources.

FHIR is designed specifically for the web with Resources and foundations in two formats: XML and JSON. The Resources use structured and standardized data for easy interoperability between EHR vendors and other software development resources and tools.

More information on FHIR can be found at: <https://www.hl7.org/fhir/overview.html>

### Other FHIR Initiatives

**SMART on FHIR** has also gained broad industry support. The SMART on FHIR initiative is based at Boston Children’s Hospital and features a set of open specifications to integrate apps with EHRs, portals, health information exchanges (HIE) and other health IT systems.

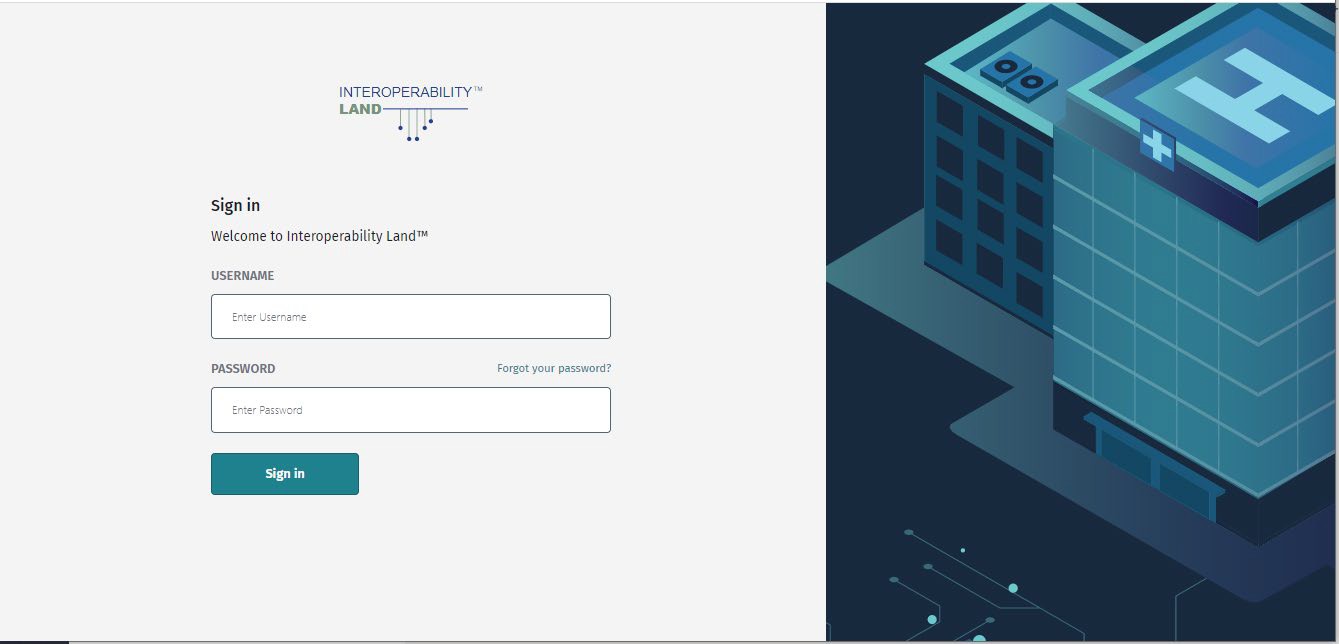
**HAPI FHIR**, otherwise known as *Happy Fire*, is a free, open-source Java implementation of the FHIR specification. It was developed at University Health Network in Ontario, Canada.

# Getting Started with IOL

## Getting Access

Participants in an event will receive a temporary login sent to the email address used to register for the event. Click the link in the email, or open your internet browser and enter the following URL: <https://www.interopland.com/login>

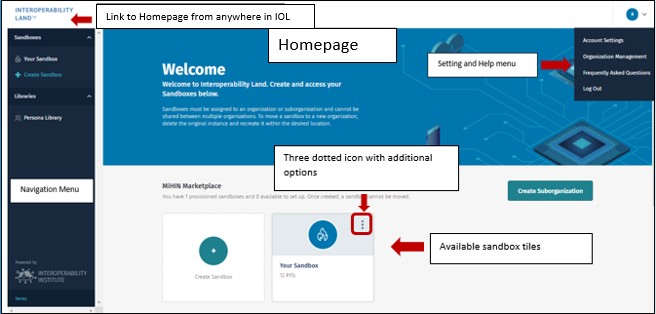
## Logging On

Log onto Interoperability Land using the provided temporary credentials.

Upon logging in, you will be prompted to enter a new password and accept the Terms and Conditions.

## Exploring the Welcome Page and Site Navigation

### Welcome

Your IOL Welcome page can be accessed at any time by clicking the **Interoperability Land logo** located above the Navigation menu.

### Navigation Menu

The left sidebar Navigation menu contains links to your **Sandboxes** and **Persona Library** based on youraccess privileges.

### Settings and Help Menu

The upper right of the IOL Welcome page displays your **Profile** icon. Click to display a drop-down menu containing **Account Settings**, **Organization Management**, **Frequently Asked Questions**, and **Log Out**.

### Dashboard

The main section of the Dashboard displays your organizations, sub-organizations, and Sandboxes. The tiles on your Home page may include the **More** tool () with additional options.

## Exploring the Sandbox

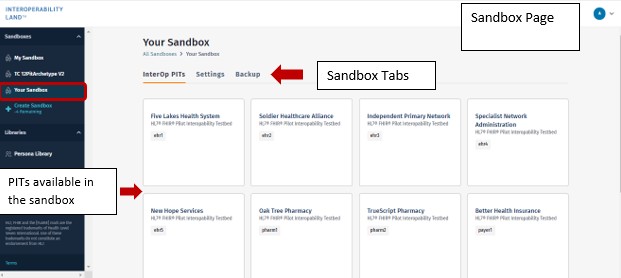
### Accessing your Sandbox

You can access your Sandbox by clicking the tile on the Home page or the link in the Navigation menu.

### Sandbox Tab Menu

The tab menu at the top allows you to cycle through the **InterOp PITs** display, Sandbox **Settings**, and Sandbox **Backup**. Use the **Settings** tab to delete your Sandbox. Use the **Backup** tab to create up to twobackups of your Sandbox or Reset to Initial State.

***Note:*** *Reset to Initial State does not**clear your backups*.

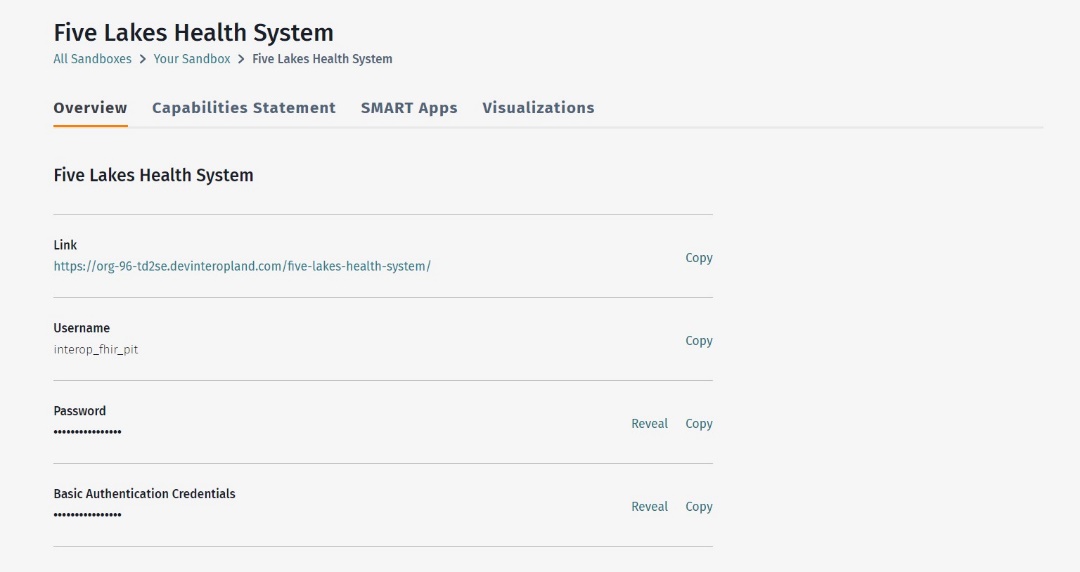


## Exploring the PITs

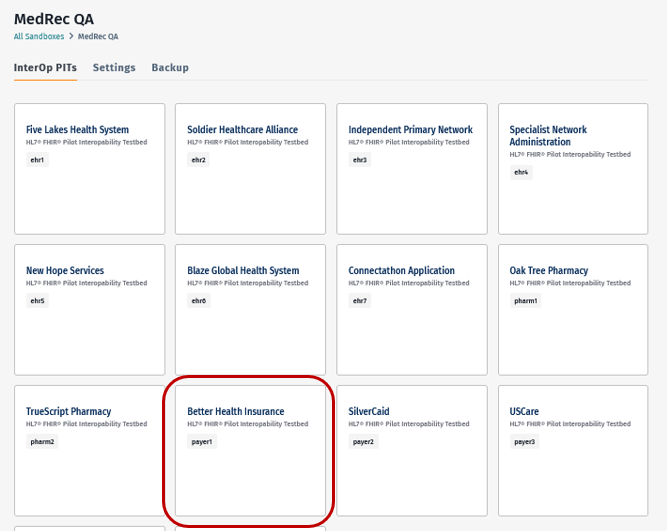
### InterOp PIT Tiles

Each tile under the **InterOp PITs** tab contains basic information about the PIT and is also a button to access the tile details menu. Click a tile to access the tile details menu for additional information about the PIT.

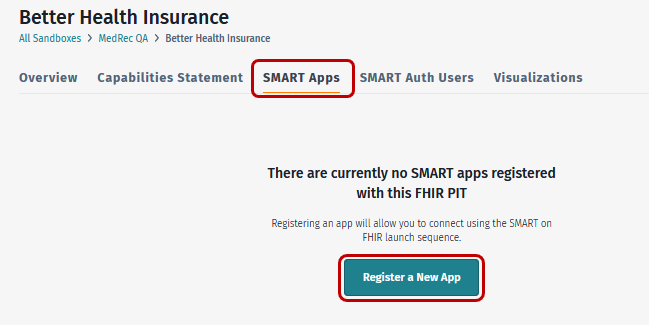
### PIT Tab Menu

The tab menu under the PIT contains the **Overview**, **Capabilities Statement**, **SMART Apps**, and **Visualizations**. The **Overview** tab contains a **Link** to access the PIT, along with the PIT’s **Username**, **Password**, and **Basic Authentication Credentials**.

## Connecting a SMART on FHIR App

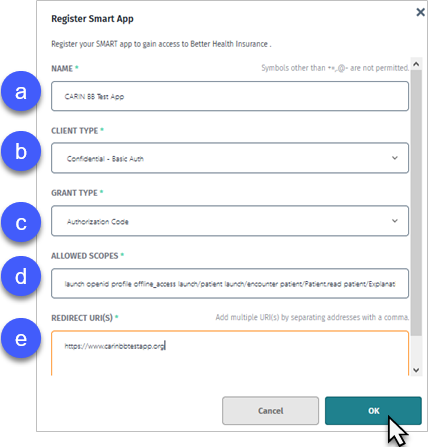
1. Choose the **PIT** you would like to connect to the SMART on FHIR application. For this Use Case, choose a **Payer PIT**.

2. Click the **SMART Apps** tab within the PIT.

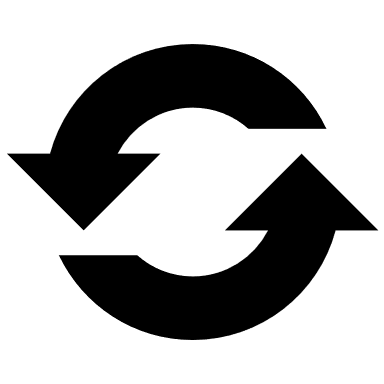


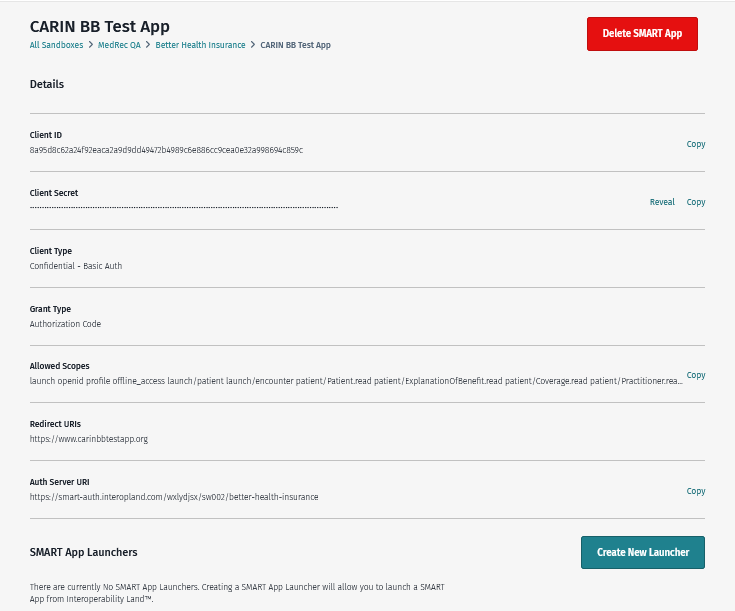
3. Then click Register a New App.

4. When the **Register SMART App** menu displays, type your information in the corresponding boxes.

1. **Name**. Name for identification of this SMART app.
2. **Client Type**. Configures how to make calls to the token endpoint. Client ID (username) and Secret (password) generated by IOL will be passed to the endpoint via this selection. Default should be fine unless it is known that another form of authentication is used by the app.
3. **Grant Type**. Allows the application to request and receive the authorization token.
4. **Allowed Scopes**. CARIN Blue Button FHIR Smart authorization scope of Resources being requested for access.
5. **Redirect URI(s).** Redirect the Uniform Resource Indicator (URI) of the application being connected.

5. Click **OK** to display the Application Details.

6. Click **Refresh** ()to populate the Client Secret.



7. Copy the **Log in URL** into your Notepad to edit.

***Note:*** *In the following example, the highlighted sections are placeholders that will be replaced with information from the application details page*.

Example URL:

<AuthServer URI from IOL>/oauth2/login?login\_type=patient&client\_id=<client\_id>&redirect\_uri=<redirect\_uri>&scope=<scope\_list>&response\_type=code&response\_mode=fragments&nonce=<random>

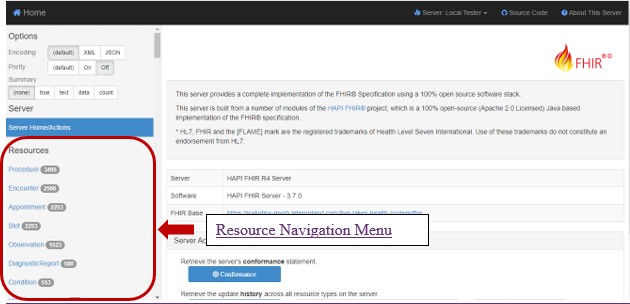
|  |  |  |
| --- | --- | --- |
| 1 | <AuthServer URI from IOL> | This is a placeholder for the Auth Server URI. |
| 2 | <client\_id> | This is a place holder for the Client ID . |
| 3 | <redirect\_uri> | This placeholder is for the Redirect URI. |
| 4 | <scope\_list> | This is a place holder for the Allowed Scopes information. |
| 5 | <random> | Any randomly generated alpha-numeric sequence of 15 characters. |

## Completing a Query

### Accessing the PIT and HAPI FHIR API

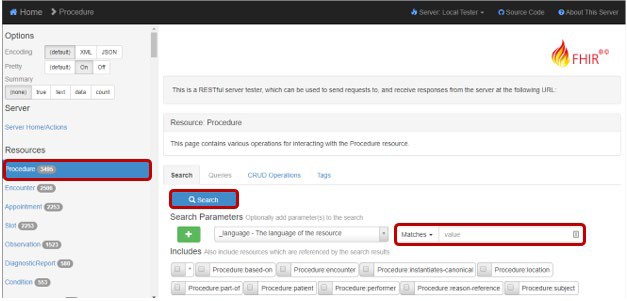
Follow the link located under the **Overview** tab within the PIT you need to query. This will access the HAPI FHIR API’s user interface.

### Navigating the Resources

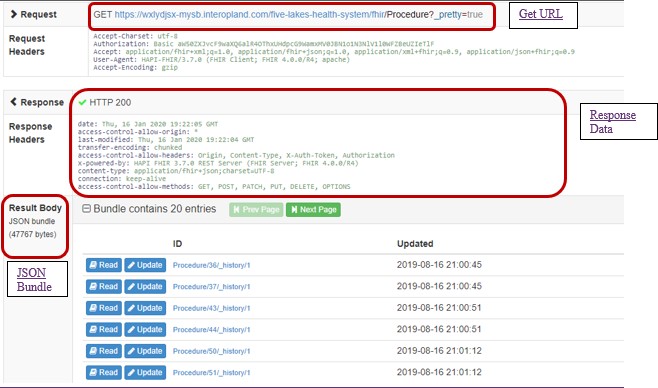
The FHIR Resources are displayed in the left-hand navigation menu. Next to each resource is a count of associated records.

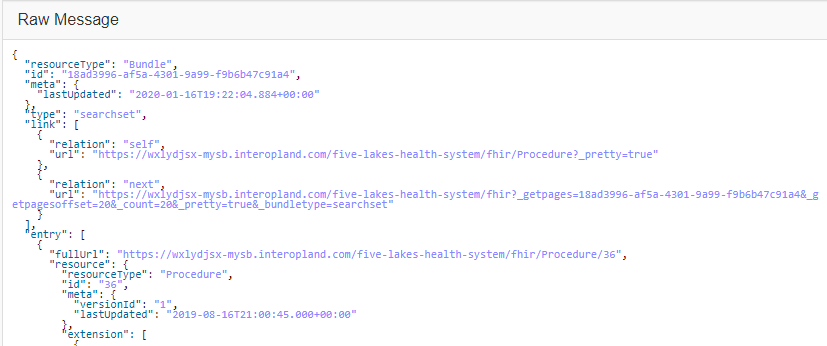
### Using the Basic Query

Choose the resource you need to query using your Resource Navigation menu. The **Search** tool will display several options to create your query. For the Basic query, which pulls every resource of the type you chose, use the default values then click **Search**.



In the **Result Body** section, you will find a **GET URL**, **Response Data**, and a **JSON Bundle** in both individual files and raw bundle formats.





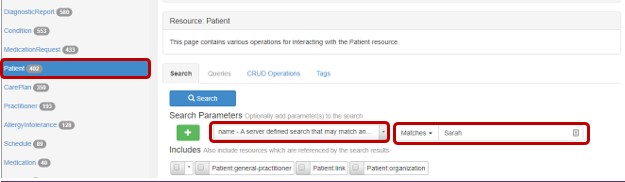
### Using the Advanced Query

Using your Resource search tool, click the **Search Parameters** dropdown menu next to the green **Add** tool to select a resource datapoint for your query. Then use the **Matches** dropdown menu to choose or type a value for your query. The server will return the same data type as before (e.g. GET URL, Response data, and a JSON bundle) in both individual files and raw bundle formats.

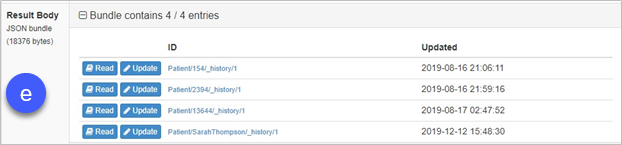
Searches for a patient can be submitted with at least one identifying piece of information such as Name, Social Security Number (SSN), or Patient ID.

Example:

1. Click the **Patient** resource.
2. Use the **Search Parameter** dropdown menu to choose **Name.**
3. Use **Matches** to type the value **Sarah**.
4. Click **Search**.



1. A **Bundle** of Patient Resources will display all entries that contain Sarah within the Name value array as shown below.



### Using the HAPI FHIR Interface Query

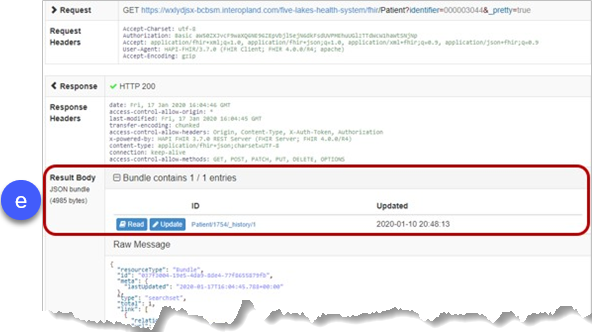
Because the HAPI FHIR Interface only supports a limited set of search parameters, an initial query to retrieve a Patient record by SSN provides the system’s internal ID associated with the patient. The ID of the Patient record will be used in future queries.

To search for a patient by SSN:

1. Choose your **Patient** option from the **Resource Navigation** sidebar.
2. Using you Search Parameter dropdown menu, choose Identifier – A patient identifier.
3. Then use your **Code** box to type your **SSN** without dashes. This example uses SSN 000003044.
4. Click **Search**.
5. The SSN example query result is shown in the **Response** block.

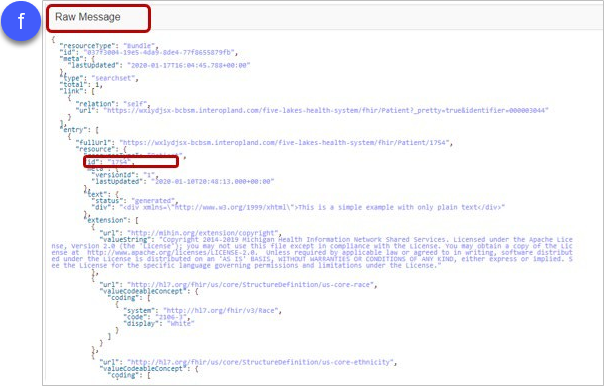
Query - Finding a Patient by SSN



The lower-right area of your window contains a section labeled **Result Body**. It displays how many results matched the specified query. The first number is the count of results retrieved and the second is the total number of records matching your query in this PIT.

When a query matches a large number of records, the numbers can change as the FHIR server has a limit on the number of records it can return in a single response.

1. Below Result Body is the **Raw Message** section displaying the raw data results. In this case, a Patient record using an SSN.



### Using the Raw Message Information Query

The unique identifier, or **id**, for each patient record within the PIT is specified under the **resource** element of a member in the **entry** collection. The **id** is what other records will use to refer to this patient and can only be used to search for records relevant to this patient.

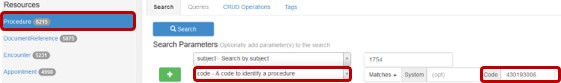
Query - Finding Medication Reconciliation Procedures by Patient



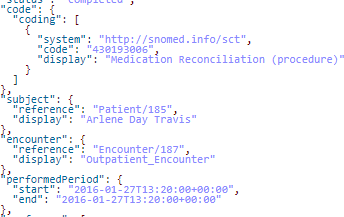
### Creating Medication Reconciliation Queries

Use your **Procedure** resource for making Medication Reconciliation queries. Use the Search Parameters to restrict the results to procedures performed on a specified patient. The search results can be further refined by using the **Code** search parameter. The filter retrieves only Medication Reconciliation procedures based on the Snomed code for that procedure: 430193006.

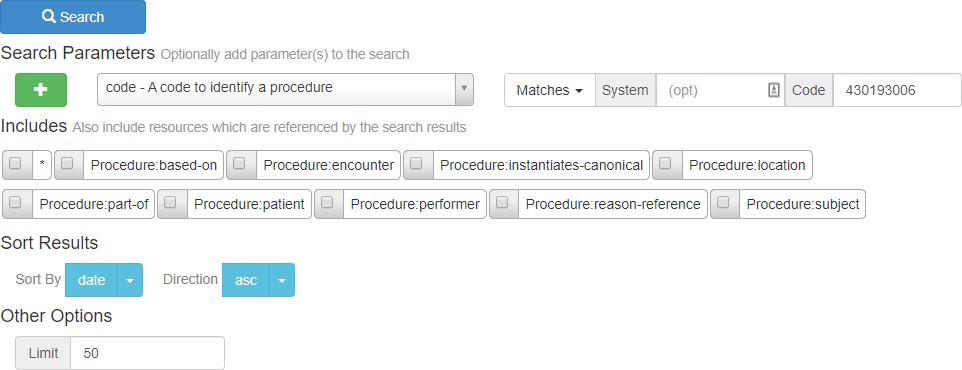
Query - All Medication Reconciliation Procedures



The result details each procedure that fits the specified criteria.



Querying for all Medication Reconciliation procedures follows the same basic format as the DocumentReference bulk queries.



HEDIS TRC compliance can now be determined by comparing the medication reconciliation procedures returned by this query to the list of discharge documents returned by the previous one.

***Note:*** *The patient name displayed in the subject field is not guaranteed to be unique. Comparisons should rely on the reference ID instead*.

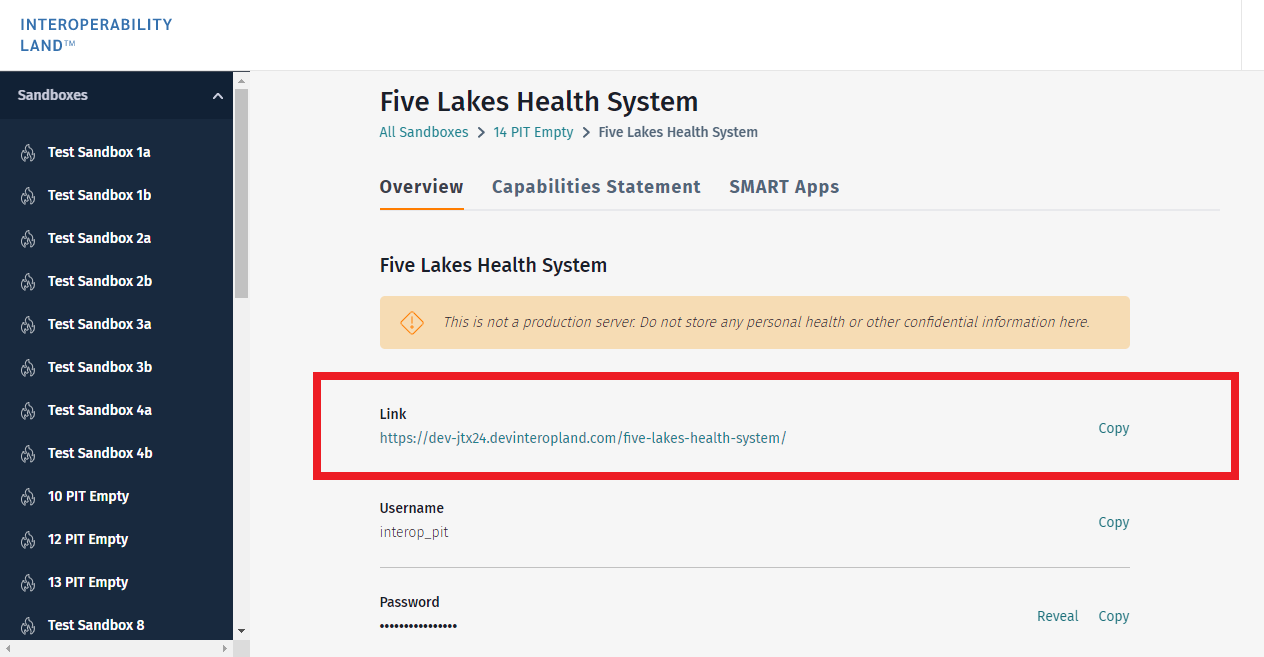
### Creating a Basic Query in Postman

Postman is a tool for API testing that organizes HTTP requests before sending them to a server. It can be used it as an alternative to the front-end query tool.

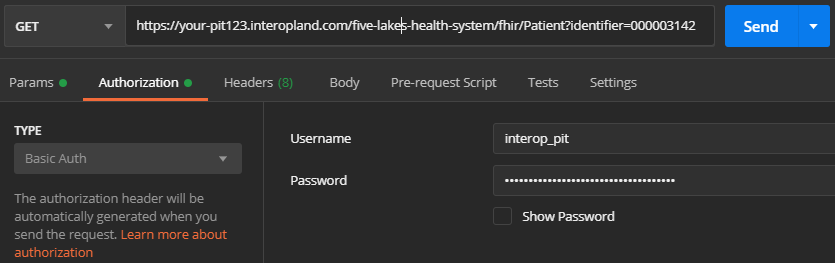
1. Locate your URL as shown in the following example:

https://<<Interoperability Land PIT details>>.interopland.com/<<FHIR-PIT-NAME>> /fhir/Patient?identifier=000003142

***Note:*** *The PIT URL and other required values can be found in the Interoperability Land PIT* ***Overview*** *tab for a given PIT. See the PIT Tab Menu section for detailed instructions. By copying the PIT details from the Interoperability Land interface, you can easily transfer the extended version of the PIT URL to Postman.*

****

1. Copy the **Username** and **Password** from the IOL PIT’s **Overview** tab and paste into the corresponding fields on the **Authorization** tab in Postman.
2. In the **TYPE** dropdown, select **Basic Auth**.
3. Click **Send** to generate an HTTP GET request and route it to the server to query for any **Patient** Resources with an identifier SSN of 000003142.



### Using Postman and the RESTful API to create more complex queries

In Appendix B, advanced examples show how to query for data over a range of dates as well as how to retrieve data from other patient related FHIR Resources.

Despite the versatility of the RESTful API, external code libraries and APIs may be required in the cases where dynamic queries need to be built from the results of an initial query. Appendix B contains a small snippet generated from the HAPI FHIR for Java project to show a more complex querying scenario. InterOpathon participants can use their own preferred FHIR R4 libraries.

# Appendix A: Terminology

| Term | Definition |
| --- | --- |
| HAPI FHIR® | A User Interface (UI) that assists in interacting with a FHIR server through querying and displaying FHIR data. |
| HL7® FHIR® | Health Level Seven’s Fast Healthcare Interoperability Resources (FHIR). |
| IOL Ring | A group of FHIR servers on the Interoperability Land platform that emulates a real-world healthcare ecosystem, including Electronic Health Records (EHRs), Pharmacies, Payers, and a Health Information Exchange (HIE), populated with synthetic data. |
| PatientGen™ | A synthetic patient data generator. PatientGen data is based on probable patient health trajectories across time and is not clinically authenticated. Synthetic data does not contain Personal Health Information of real patients. |
| Persona | A synthetic patient based on a user story and customized to align with specific use cases. These are authenticated by a clinician for accuracy and relevance. Synthetic data does not contain Personal Health Information of real patients. |
| PIT | InterOp Station’s Pilot Interoperability Testbed, a FHIR server that represents a real-world healthcare organization, populated with synthetic patient data. |
| Resource | Any data file containing synthetic patient data. Formatted in JSON. |
| Sandbox | The development environment that contains PITs and IOL Ring. |

# Appendix B: Querying FHIR Resources

FHIR servers provide Resources, or health-care related data objects that can be created, read, updated, or deleted. These Resources have a variety of uses and industry adoption. A full list of Resources and their level of maturity can be found at <https://www.hl7.org/fhir/resourcelist.html>.

Resources can represent a variety of topics which may be useful to different organizations. For example, hospitals need the Patient resource, as well as Observations, Immunizations, and AllergyIntolerances. Insurance companies focus on Procedures, Claims, and Coverage Resources. University research labs might use a broad category like ResearchSubject or a narrow one like SubstanceNucleicAcid.

There are also Resources needed for server infrastructure to exchange data. For example, the infrastructure uses the CapabilityStatement to advertise known versions and Resources. The OperationDefinition, advertises which operations the server can perform on Resources.

One useful way to work with FHIR Resources is by utilizing ‘GET’ requests in Postman to query for the Resources using the RESTful API.

In Postman, the following GET request:

https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/Patient

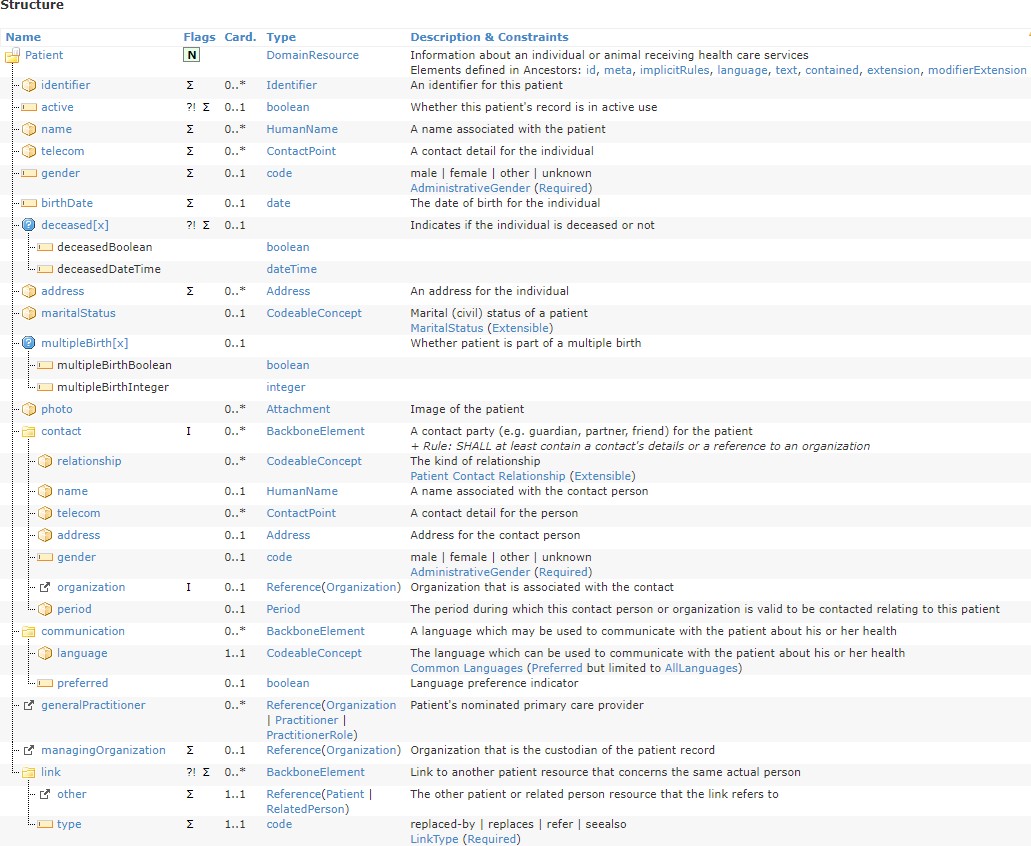
returns this data set:



***Note:*** *It may be difficult to read query responses due to the amount of data returned and displayed*.

***Tip:*** *The keyboard shortcut to fold results is Alt-0/Shift-Alt-0. This makes FHIR query responses easier to navigate and understand*.

The fields in the data set map to the Resource Patient section of the HL7 FHIR standard ([Section 8.1.2 Resource Content](https://www.hl7.org/fhir/patient.html#resource)). The Σ flag in the below screenshot indicates that these fields will show up in a patient **summary** request. Some fields are absent in the results – this is either because there is no data, or you need to request a non-summary version using a more focused request (e.g. http://[url]/patient/[**id**]).



## Example query structure

[base URL] [organization name] [“/fhir/”] [resourceType]

The following query returns all DocumentReference Resources from the Five Lakes Health System PIT:

https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/DocumentReference

***Note:*** *The default number of records is 20 and maximum records possible for return is 200*.

Example of a returned Document Reference:

|  |  |
| --- | --- |
| **ResourceType** | **DocumentReference** |
| Id | 29106 |
| masterIdentifier | 2deaf4a0-1b3a-4568-b129-35852ea49c2d |
| Identifier | 2deaf4a0-1b3a-4568-b129-35852ea49c2d |
| Description | A03 of encounter Id = 2deaf4a0-1b3a-4568-  b129-35852ea49c2d |
| Subject.reference | Patient/424 |
| Context.encounter.reference | Encounter/29102 |
| Period.start | 2023-05-12T00:00:00+00:00 |
| Period.end | 2023-05-12T00:00:00+00:00 |

### Advanced query formatting

If the SSN for a Patient is known, it is possible to look up any DocumentReference Resources referencing that patient:

https://<<PIT DETAILS>>/new-hope-services/fhir/DocumentReference?subject:Patient.identifier=000002885

***Note:*** *The ‘000002885’ can be replaced with another SSN to find a different patient’s DocumentReference Resources*.

|  |  |
| --- | --- |
| **ResourceType** | **DocumentReference** |
| Id | 29106 |
| masterIdentifier | 2deaf4a0-1b3a-4568-b129-35852ea49c2d |
| Identifier | 2deaf4a0-1b3a-4568-b129-35852ea49c2d |
| Description | A03 of encounter Id = 2deaf4a0-1b3a-4568-  b129-35852ea49c2d |
| Subject.reference | Patient/424 |
| Context.encounter.reference | Encounter/29102 |
| Period.start | 2023-05-12T00:00:00+00:00 |
| Period.end | 2023-05-12T00:00:00+00:00 |

## Querying for a range of values

The following example queries for any **Encounter** Resources within a date range: “greater than or equal to 2025-01-01 and less than or equal to 2025-01-31”.

https://<< PIT DETAILS>>.interopland.com/five-lakes-health-system /fhir/Encounter?date=ge2025-01-01&date=le2025-01- 31&patient=170

The next example queries for patients linked to a **Procedure** having Snomed medication reconciliation code 430193006.

https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/ fhir/Patient?\_has:Procedure:subject:code=430193006

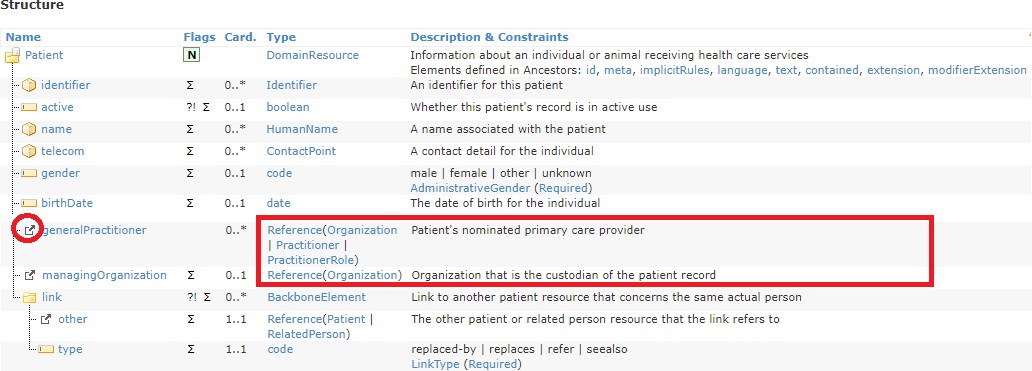
To include additional Resources, use the **\_include** or **\_revinclude** query parameters. Use these parameters when the base resource (e.g. **Patient**) has a link to an external resource or if the external resource (e.g. **Encounter**) links back to it.

https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/ /fhir/Patient?identifier=000002875&\_count=1&\_revinclude=Encounter:subject

## Data linkage

It may be useful to conceptualize the data as a SQL table, despite the back-end technology being NoSQL. In doing this, the concept of Foreign Keys can be used to navigate and discover data relationships.

References are valuable when crafting more complex queries, such as the those seen in the following section. See the individual resource’s documentation on the HL7 FHIR site, and search for ‘Reference’ under the type column <https://www.hl7.org/fhir/patient.html>.



For a complete list of resource references, see Section 8.1.1 Scope and Usage on the [HL7 FHIR](https://www.hl7.org/fhir/patient.html#scope) site in the paragraph beginning with “This resource is referenced by… “

In theory, you can get a detailed picture of a Patient resource in one JSON request by using **\_revinclude** on the whole list.

***Note:*** *It is not recommended to use* ***\_revinclude*** *in a single JSON request to retrieve Patient resource data for several reasons:*

* *Not all systems implement all Resources.*
* *Security features may be in place to limit retrieval, impacting results.*
* *Queries for large data sets can stress the server*.

For an example of a targeted query, note the parameters used in the following query:

https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/Patient?identifier=000002867&\_count=1&\_revinclude=DocumentRefere nce:subject&\_revinclude=Encounter:subject

The query uses the following limits:

|  |  |
| --- | --- |
| **\_count**=1 | Limits our result set to only one patient, to save on server load. |
| **identifier**=000002867 | Patient resource has SSN as identifier, for targeted searching. |
| **\_revinclude**=DocumentReference:subject | Fetches and includes DocumentReferences that reference the patient. |
| **\_revinclude**=Encounter:subject | Fetches and includes Encounters that reference the patient. |

## Crafting more complex queries to answer questions

The FHIR server offers querying functionality accessible through the endpoints used in previous exercises. It is capable of simple join operations and simple filtering. The mechanisms are like the JOIN and WHERE statements in SQL.

The following is a basic summary query for 20 patients:

https:// <<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/Patient

The following is a more complex query for 20 patients that have DocumentReferences linking back to the patient \_and\_ has A01 in the description. Note that the query does NOT include the DocumentReferences in the result.

https:// <<PIT DETAILS>>.interopland.com/five-lakes-health-system /fhir/Patient?has:DocumentReference:patient:description=A01

As seen in the previous section, the **\_has** parameter uses references on the **DocumentReference** that point to patient. You can use **\_revinclude**, but this would include the **DocumentReference** in the JSON results.

***Note:******\_has*** *can also achieve this if \_****contained=true*** *is in the query***.**

### Limitations to out-of-the-box queries

* No sub-queries without customizing the server i.e., unable to query the result set from a complex query.
* No programmatically extended ranges.
* No way to dynamically query ‘between 1/1/2020 and 30 days from then’.

### Workarounds

* FHIR client libraries implementing the desired Resources for your desired standard version, i.e., R4.
* Available in Java, Python, .NET.
* Use lists/arrays to hold the results, then use forEach to filter/process or programmatically perform follow-up queries against FHIR server as needed.

**public static void** main(String[] args) { FhirContext ctx = FhirContext.*forR4*();

String serverBase = "https://replace-this-with-your-details.interopland.com/five-lakes-health- system/fhir/";

IGenericClient gclient = ctx.newRestfulGenericClient(serverBase); gclient.registerInterceptor(

**new** BasicAuthInterceptor("interop\_pit", "your-password-goes-here"));

**class** userDischargeDate { String userId;

String ADT30DayRangeStart; String ADT03DischargeEndDate;

**boolean** followupEncounter = **false**;

**public** userDischargeDate(String inUserId, Date ADT03DischargeDate) { userId = inUserId;

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy-MM-dd"); ADT30DayRangeStart = sdf.format(ADT03DischargeDate); ADT03DischargeEndDate = sdf

.format(DateUtils.*addDays*(ADT03DischargeDate, 30));

}

}

List<userDischargeDate> Adt03sWithUsers = **new** ArrayList<userDischargeDate>(); List<String> followUpEncountersWithin30Days;

Bundle ADT03Results = gclient.search().forResource(DocumentReference.**class**)

.where(DocumentReference.***DESCRIPTION***.contains().value("A03"))

.returnBundle(Bundle.**class**).execute();

ADT03Results.getEntry().forEach((entry) -> {

DocumentReference docRef = (DocumentReference) entry.getResource();

**if** (docRef.hasSubject()) {

Reference subject = docRef.getSubject(); Adt03sWithUsers

.add(**new** userDischargeDate(subject.getId(), docRef.getDate()));

}

});

Adt03sWithUsers.forEach((userDischargeDate) -> {

Bundle EncounterResults = gclient.search().forResource(Encounter.**class**)

.where(Encounter.***PATIENT***.hasId(userDischargeDate.userId))

.and(Encounter.***DATE***.afterOrEquals()

.day(userDischargeDate.ADT30DayRangeStart))

.and(Encounter.***DATE***.beforeOrEquals()

.day(userDischargeDate.ADT30DayRangeStart))

.returnBundle(Bundle.**class**).execute();

**if** (EncounterResults.hasEntry()) { userDischargeDate.followupEncounter = **true**; System.***out***.println("Patient: " + userDischargeDate.userId

+ " had a follow-up visit within 30 days");}