Setting up CRD, DTR and PAS Reference Implementations (RIs)

The following documentation are instructions on how to set up the complete Coverage Requirement Discovery (CRD), Documentation Templates and Rule (DTR) and Prior Authorization Support (PAS) Reference Implementations (RIs). At the end of this you will be able to use the CRD Request Generator to determine if Prior Authorization and/or required documentation is necessary (CRD), launch a SMART on FHIR application to complete a Questionnaire (DTR), and submit the Claim to request Prior Authorization (PAS).

Since there are many projects being used in this sequence it is highly recommended to place all of them in the same root directory.

Prerequisites

CRD and PAS require Java and Node. The current approved versions are:

Java	Oracle or OpenJDK – v8 or v11
Node	12+
Gradle	5.6.2 or 6.3

Setting up the CRD Request Generator

The CRD Request Generator is a helper client application used to send CDS Hooks requests to CRD, launch the SMART on FHIR application, and submit a Prior Authorization Request. It can be found at https://github.com/HL7-DaVinci/crd-request-generator. Clone the repo and install dependencies by running:

```
$ git clone https://github.com/HL7-DaVinci/crd-request-generator.git
$ cd crd-request-generator
$ npm install
```

Setting up CRD and CDS Library

The CRD RI can be found at https://github.com/HL7-DaVinci/CRD. Clone and build the repo by running

```
$ git clone https://github.com/HL7-DaVinci/CRD.git
$ cd CRD
$ gradle build
```

A copy of the CDS Library repository needs to be cloned into the CRD repository. Running embedCdsLibrary will do this by running the steps described below.

```
$ cd CRD/server
$ gradle embedCdsLibrary
```

Setting up the Test EHR and Keycloak

To run CRD and DTR a test EHR Server is required. The test EHR Server used in the RIs can be found at https://github.com/HL7-DaVinci/test-ehr. Clone the repo by running:

```
$ git clone https://github.com/HL7-DaVinci/test-ehr.git
```

The EHR Server uses Keycloak to authenticate users. Install and configure Keycloak by doing the following.

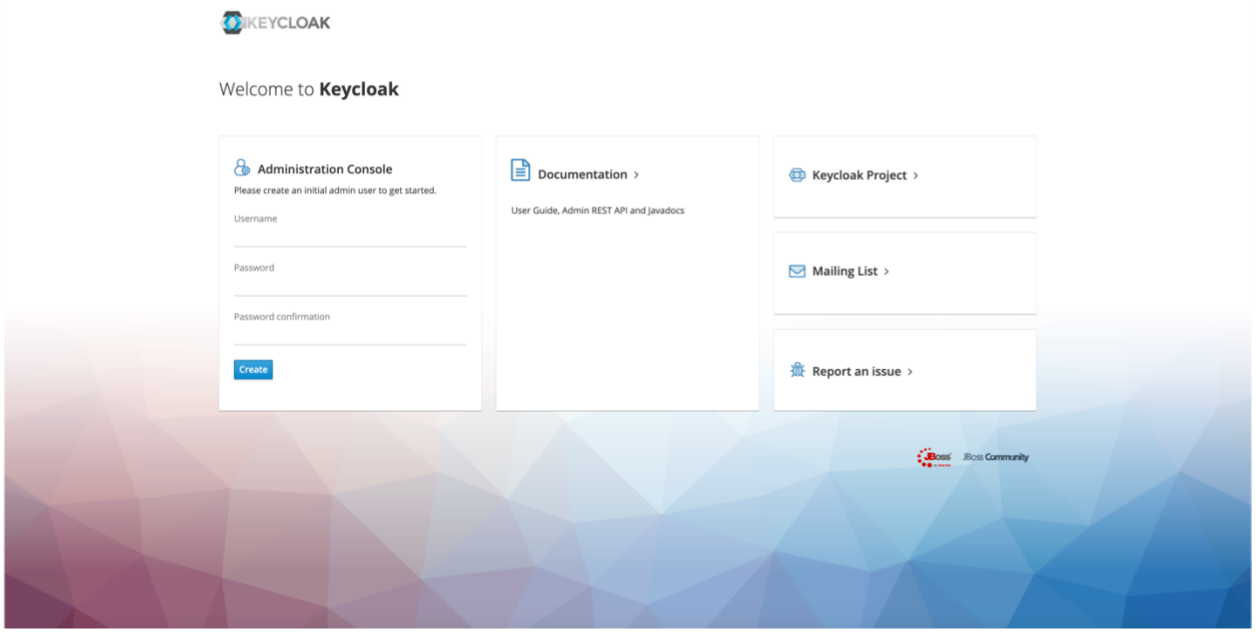
1. Download the Server (Standalone server distribution) from https://www.keycloak.org/downloads.html
2. Move the downloaded zip file (or tar ball) to the location of the other projects. Unzip using either

   ```
   $ unzip keycloak-<version>.zip
   $ tar -xf keycloak-<version>.tar.gz
   ```

   depending on which file was downloaded. The zipped file can now be deleted if desired.
3. Run the Keycloak server:

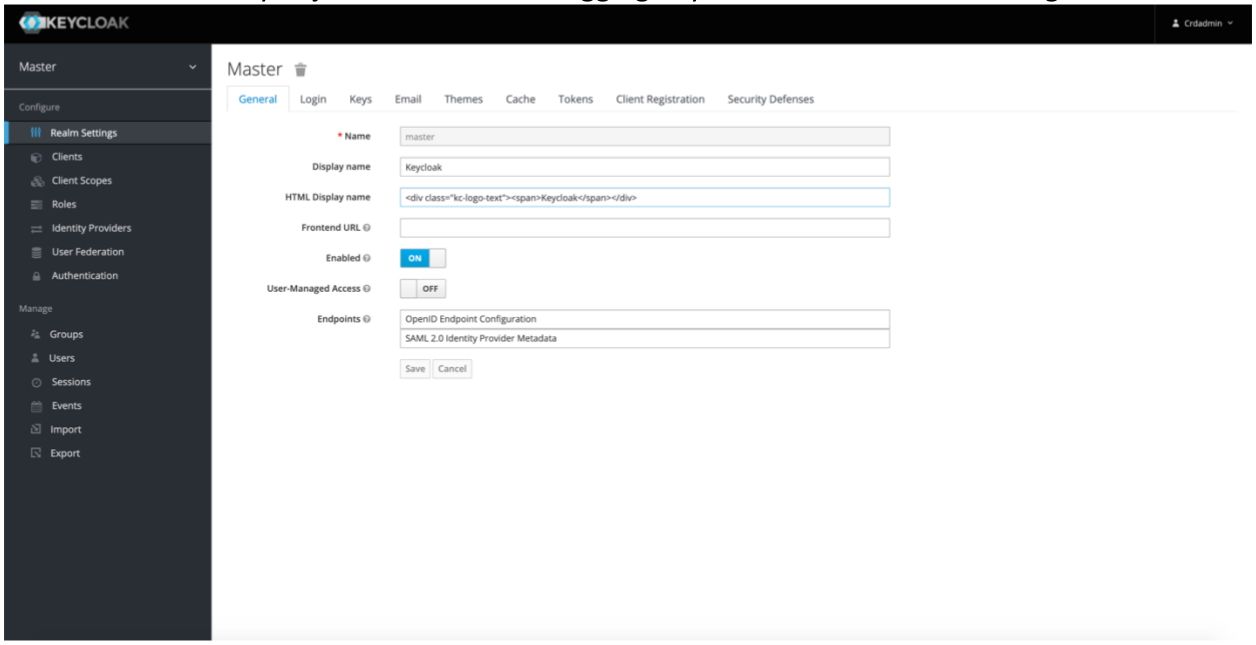
   ```
   $ cd keycloak-<version>/bin
   $ ./standalone.sh -Djboss.socket.binding.port-offset=100
   ```

4. Once the Keycloak instance is up and running navigate to http://localhost:8180/ to view it. You should see a screen similar to the one below.



5. On the left side under “Administration Console” create a new admin user.

6. Click “Administration Console” to login into the Admin Console using the admin user credentials you just created. After logging in you should see the following screen.



* Name: master
Display name: Keycloak
HTML Display name: <div class="kc-logo-text"><span>Keycloak</span></div>
Frontend URL:
Enabled: ON
User-Managed Access: OFF
Endpoints: OpenID Endpoint Configuration
SAML 2.0 Identity Provider Metadata

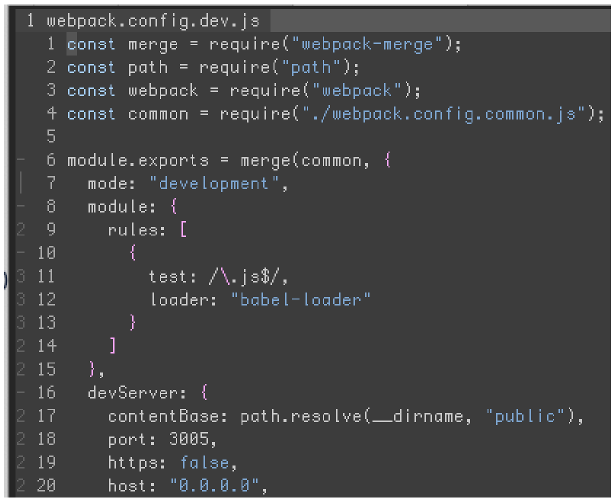
7. Create a new Realm to protect the test-ehr by:
   a. Hover over “Master” in the top left and click the “Add Realm” button.
   b. Import the realm by clicking “Select file” next to import. Select {Path to test-ehr} /src/main/resources/ClientFhirServerRealm.json This will automatically fill in the details on the Add Relam page. Click “Create” to continue.
8. Create a new user for the ClientFhirServer Realm:
   a. Next create a new user by navigating to the “Users” tab under the “Manage” section of the left sidebar. There should currently be no users listed. On the right click “Add User” to create the new user.
   b. Give the new user a Username and click “Save”.
   c. Give the new user a password by navigating to the newly visible “Credentials” tab. Before setting the password ensure the slider next to “Temporary” is off.
   d. Give the new user the role of “user” by navigating to the “Role Mappings” tab, clicking “user” in the “Available Roles” selection, and then clicking “Add Selected”.

Setting up DTR

The DTR RI can be found at https://github.com/HL7-DaVinci/dtr. Clone the repo and install dependencies by running

```
$ git clone https://github.com/HL7-DaVinci/dtr.git
$ cd dtr
$ npm install
```

By default, DTR will run as https. When running the server locally you will most likely want to run it as http instead. Do this by opening {Path to DTR}/webpack.config.dev.js and editing line 19 to set https to false.



```
 1 webpack.config.dev.js
 1 const merge = require("webpack-merge");
 2 const path = require("path");
 3 const webpack = require("webpack");
 4 const common = require("./webpack.config.common.js");
 5
 6 module.exports = merge(common, {
 7   mode: "development",
 8   module: {
 9     rules: [
10       {
11         test: /\.js$/,
12         loader: "babel-loader"
13       }
14     ]
15   },
16   devServer: {
17     contentBase: path.resolve(__dirname, "public"),
18     port: 3005,
19     https: false,
20     host: "0.0.0.0",
```

Setting up PAS

The PAS RI can be found at https://github.com/HL7-DaVinci/prior-auth. Clone and build the repo by running

```
$ git clone https://github.com/HL7-DaVinci/prior-auth.git
$ cd prior-auth
$ ./gradlew installBootDist
```

Using the Request Generator to Submit a Prior Authorization

To use the CRD Request Generator to submit a Claim to the Prior Auth RI we begin by launching each service in a new terminal window:

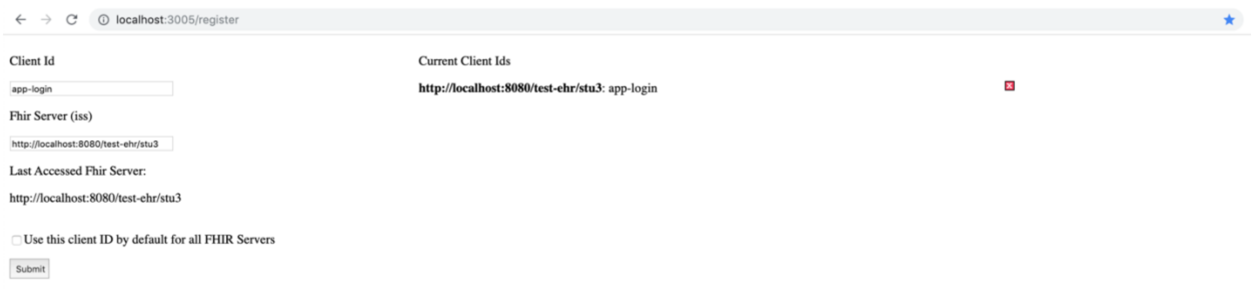
```
$ cd {Path to CRD}/server
$ gradle bootRun

$ cd {Path to test-ehr}
$ gradle appRun

$ cd {Path to Keycloak}/bin
$ ./standalone.sh -Djboss.socket.binding.port-offset=100

$ cd {Path to DTR}
$ npm start

$ cd {Path to PAS}
$ ./gradlew bootRun

$ cd {Path to CRD Request Generator}
$ npm start
```

The final command will launch the Request Generator in a new browser window. If it does not open automatically navigate to http://localhost:3000/ehr-server/reqgen to view the Request Generator. At the end of this document there is a table to map service name to location (port).

Before using the Request Generator there are a few more steps to configure the running servers. First test data is loaded into the EHR Server. Execute the following once the EHR Server is up and running:

```
$ cd {Path to test-ehr}
$ gradle loadData --stacktrace
```

Finally, we must update DTR to use the test EHR. Navigate to http://localhost:3005/register and add http://localhost:8080/test-ehr/r4 as the FHIR Server (iss) and "app-login" for the Client Id. If a previous FHIR Server is listed under "Current Client Ids" for the Client Id "app-login" then remove it by clicking the "x" next to it.


localhost:3005/register

Client Id: app-login

Fhir Server (iss): http://localhost:8080/test-ehr/stu3

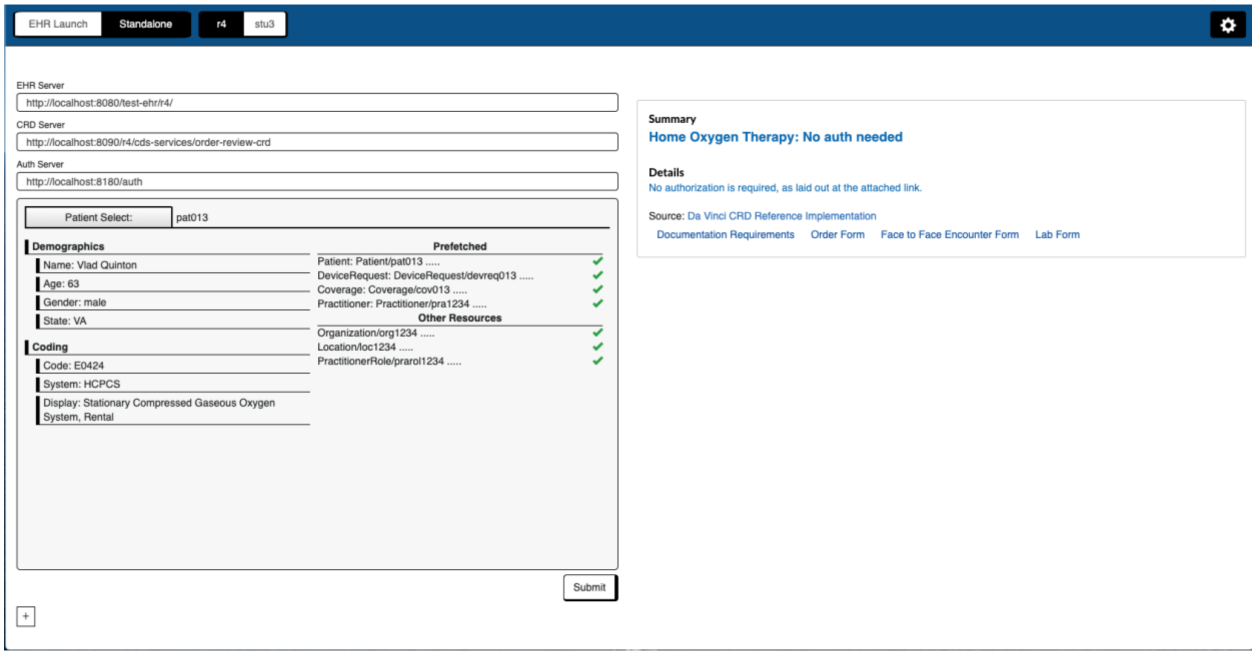
Last Accessed Fhir Server:
http://localhost:8080/test-ehr/stu3

Use this client ID by default for all FHIR Servers

Current Client Ids

http://localhost:8080/test-ehr/stu3: app-login

Now we are finally ready to proceed. Navigate to the Request Generator in your browser. After selecting R4 in the nav bar, select a patient. For this demo we will select Vlad Quinton (pat013) and in the dropdown for DeviceRequest select E0424. The information on the left panel will update automatically. Click "Submit" and the right side will populate with at least one card as shown below:



EHR Server
http://localhost:8080/test-ehr/r4/

CRD Server
http://localhost:8090/r4/cds-services/order-review-crd

Auth Server
http://localhost:8180/auth

Patient Select: pat013

Demographics

- Name: Vlad Quinton
- Age: 63
- Gender: male
- State: VA

Coding

- Code: E0424
- System: HCPCS
- Display: Stationary Compressed Gaseous Oxygen System, Rental

Prefetched

- Patient: Patient/pat013 ..... ✓
- DeviceRequest: DeviceRequest/devreq013 ..... ✓
- Coverage: Coverage/cov013 ..... ✓
- Practitioner: Practitioner/pra1234 ..... ✓

Other Resources

- Organization/org1234 ..... ✓
- Location/loc1234 ..... ✓
- PractitionerRole/prarol1234 ..... ✓

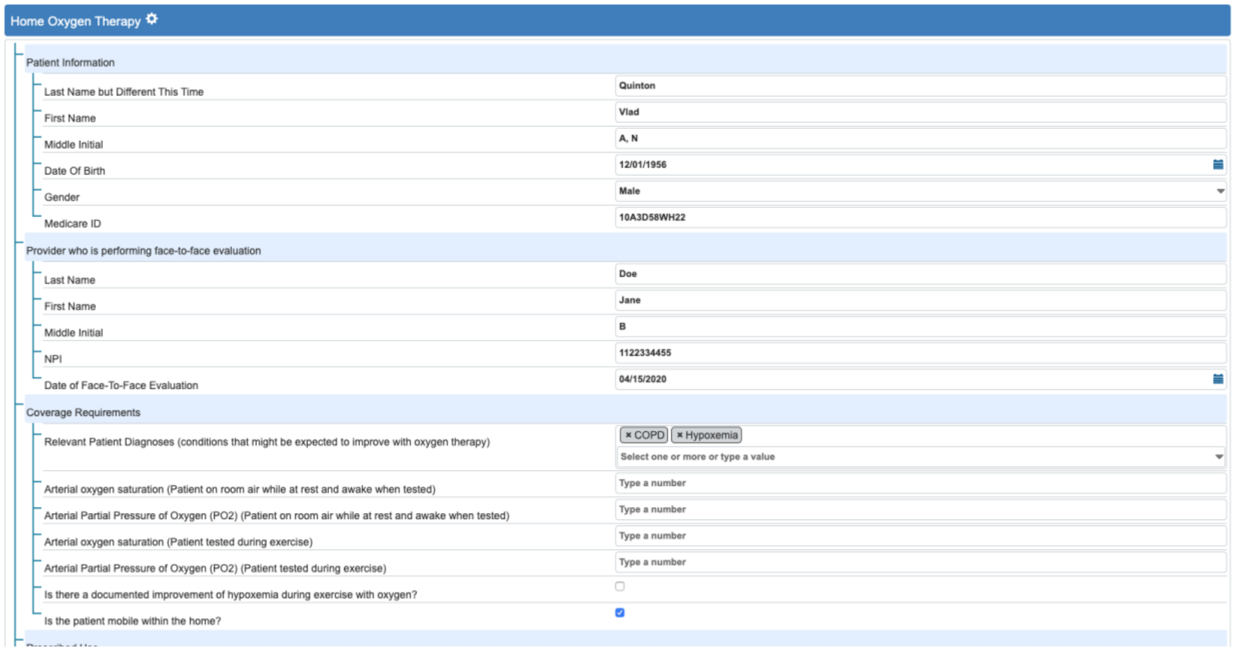
Summary
Home Oxygen Therapy: No auth needed

Details
No authorization is required, as laid out at the attached link.

Source: Da Vinci CRD Reference Implementation

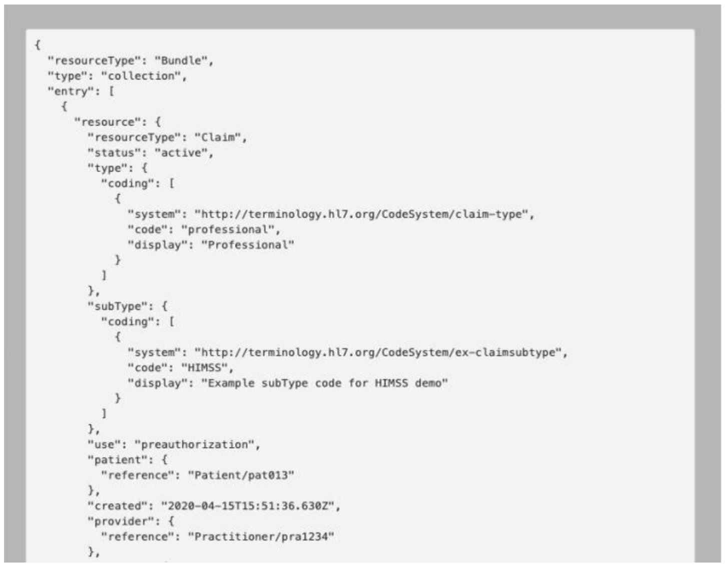
Documentation Requirements   Order Form   Face to Face Encounter Form   Lab Form

Since authorization is required, we launch the SMART App to complete the Questionnaire by clicking on "Open Form". This will open in a new page and Keycloak will ask you to authenticate. Log in with the user created in Step 8 of Setting up CRD and Keycloak. After successfully authenticating the Questionnaire will be displayed with as many responses preloaded from the Patient (as shown below).


Home Oxygen Therapy

Patient Information	
Last Name but Different This Time	Quinton
First Name	Vlad
Middle Initial	A, N
Date Of Birth	12/01/1956
Gender	Male
Medicare ID	10A3D58WH22
Provider who is performing face-to-face evaluation	
Last Name	Doe
First Name	Jane
Middle Initial	B
NPI	1122334455
Date of Face-To-Face Evaluation	04/15/2020
Coverage Requirements	
Relevant Patient Diagnoses (conditions that might be expected to improve with oxygen therapy)	× COPD × Hypoxemia; Select one or more or type a value
Arterial oxygen saturation (Patient on room air while at rest and awake when tested)	Type a number
Arterial Partial Pressure of Oxygen (PO2) (Patient on room air while at rest and awake when tested)	Type a number
Arterial oxygen saturation (Patient tested during exercise)	Type a number
Arterial Partial Pressure of Oxygen (PO2) (Patient tested during exercise)	Type a number
Is there a documented improvement of hypoxemia during exercise with oxygen?	☐
Is the patient mobile within the home?	✓

After all responses have been provided click "Next" at the end of the form. You will now be presented with a page to select the Prior Authorization Endpoint. Select the desired endpoint from the drown down and then send the prior auth to the server by clicking "Submit".

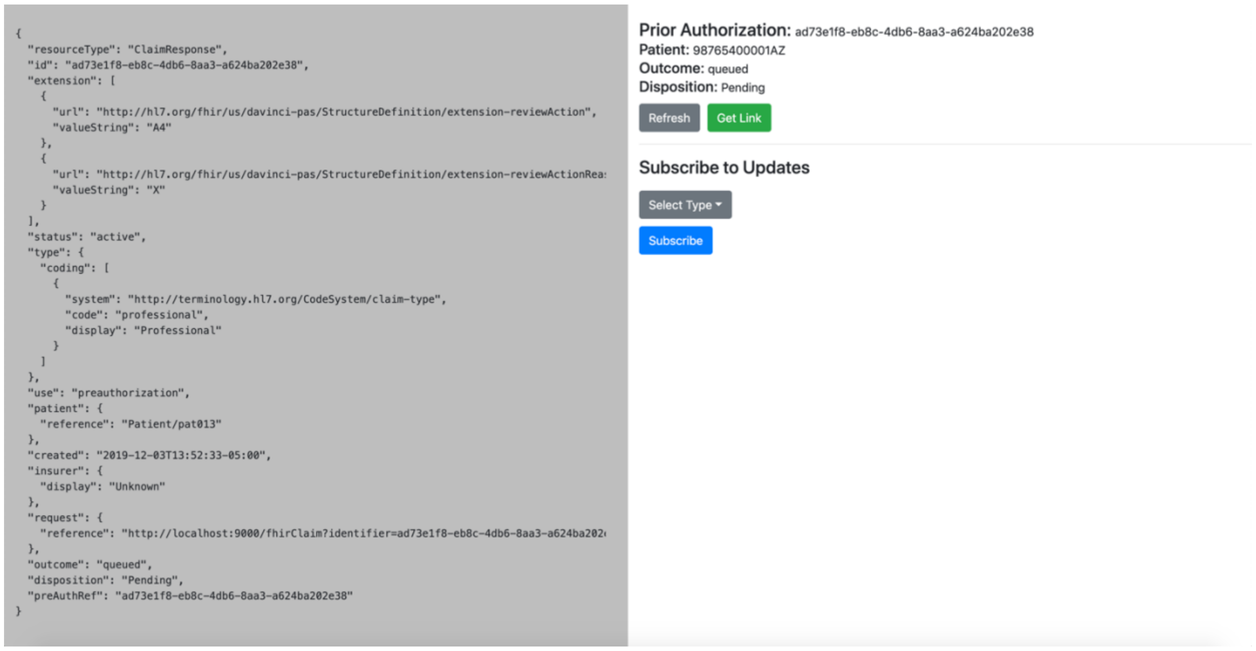


```
{
  "resourceType": "Bundle",
  "type": "collection",
  "entry": [
    {
      "resource": {
        "resourceType": "Claim",
        "status": "active",
        "type": {
          "coding": [
            {
              "system": "http://terminology.hl7.org/CodeSystem/claim-type",
              "code": "professional",
              "display": "Professional"
            }
          ]
        },
        "subType": {
          "coding": [
            {
              "system": "http://terminology.hl7.org/CodeSystem/ex-claimsubtype",
              "code": "HIMSS",
              "display": "Example subType code for HIMSS demo"
            }
          ]
        },
        "use": "preauthorization",
        "patient": {
          "reference": "Patient/pat013"
        },
        "created": "2020-04-15T15:51:36.630Z",
        "provider": {
          "reference": "Practitioner/pra1234"
        },
```

Submit Prior Auth

Select PriorAuth Endpoint: Logica Health: https://davinci-prior-auth.logicahealth.org/fhir

After the request is sent and a response is received the disposition from the payer will be displayed on a page similar to the one below:



```
{
  "resourceType": "ClaimResponse",
  "id": "ad73e1f8-eb8c-4db6-8aa3-a624ba202e38",
  "extension": [
    {
      "url": "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-reviewAction",
      "valueString": "A4"
    },
    {
      "url": "http://hl7.org/fhir/us/davinci-pas/StructureDefinition/extension-reviewActionRea
      "valueString": "X"
    }
  ],
  "status": "active",
  "type": {
    "coding": [
      {
        "system": "http://terminology.hl7.org/CodeSystem/claim-type",
        "code": "professional",
        "display": "Professional"
      }
    ]
  },
  "use": "preauthorization",
  "patient": {
    "reference": "Patient/pat013"
  },
  "created": "2019-12-03T13:52:33-05:00",
  "insurer": {
    "display": "Unknown"
  },
  "request": {
    "reference": "http://localhost:9000/fhirClaim?identifier=ad73e1f8-eb8c-4db6-8aa3-a624ba202
  },
  "outcome": "queued",
  "disposition": "Pending",
  "preAuthRef": "ad73e1f8-eb8c-4db6-8aa3-a624ba202e38"
}
```

Prior Authorization: ad73e1f8-eb8c-4db6-8aa3-a624ba202e38
Patient: 98765400001AZ
Outcome: queued
Disposition: Pending

On the left is the raw ClaimResponse resource returned by the payer. The right panel contains human readable details as well as action items. If the disposition is “Pending” (or the outcome is “queued”) then you can subscribe to updates. Select the type of Subscription from the “Select Type” dropdown and then confirm the subscription by clicking the “Subscribe” button.

If WebSockets are selected, a ws connection is established between your browser and the PAS server. As soon as an update is available the browser will update automatically. Another option is Polling which will periodically check the PAS server for an update. The RestHook subscriptions require a server which exposes REST endpoints. This functionality is not available through the browser. However, if you had another server configured to accept and process these updates then the subscription can still be created through this UI.

CRD, DTR, and PAS Service to Location (Port) Mappings

Service Name	**Location**
CRD Server	http://localhost:8090
EHR Server	http://localhost:8080/test-ehr
Keycloak	http://localhost:8180
DTR	http://localhost:3005
Prior Auth	http://localhost:9000
CRD Request Generator	http://localhost:3000