Purple Dragon FHIR Resource Aggregation App Manual (Team: Purple Dragons)

Team Members:
Godswill Oletu (goletu3)
Lucas Vollherbst (lvollherbst3)
Pingying Zeng (pzeng7)

Project:
CDC Population Health Informatics Framework: Population Health Management Data for Public Health Use

Github:
Private GA Tech location:
https://github.gatech.edu/gt-hit-fall2016/Population-Health-Management-Data

Public github with private project for non-GA Tech access:
https://github.com/Purpledragonapp/Management-Health-Data-For-Public-Health-Use

CS 6440: Into to Health Informatics
Fall 2016

Table of Contents

1.0 General Information

1.1 System Overview

PurpleDragon is a Javascript web app built on using the FHIR standards framework. The app provides three main features:

- **Resource aggregation:** The app aims to provide easy aggregation of FHIR resources for a number of clients in a geographical region and provide these resources to a centralized public health agency for analysis.
- **De-duplication:** PurpleDragon aims to prevent duplication of patient and other resources that may exist in multiple client databases within the region to provide an accurate representation of the region to the public health agency.
- **Anonymization:** The app also interfaces with patient masking services to provide anonymization of patient resources to protect patient confidentiality when data is passed to organizations outside of the patients' care facilities.

This project is a model implemented by Graduate students from Georgia Tech Institute of Technology.

1.2 Organization of this Document

Section 1.0 General Information provides an overview of the system and provides a brief description of the contents of this document.

Section 2.0 System Components outlines the different pieces of the app, including the core Javascript app as well as the supplied example interface and system model.

Section 3.0 Using the App outlines how each component of the app can be used.

2.0 System Components

2.1 The Core PurpleDragon app

The main component of the PurpleDragon FHIR Resource aggregation app is the core Javascript app, purpledragon.js. This file provides a Javascript class which should be instantiated once for each population health management client that is part of the system. This class requires URLs for 3 FHIR servers:

1. The Unmasked PurpleDragon FHIR server that will maintain unmasked aggregate data for de-duplication and re-linking services.
2. The public health agency's FHIR server where the patient-masked FHIR resources will be placed.
3. An individual population health management client FHIR server that contains the resources that will be masked and sent to the public health agency.

This class also contains all the methods for processing resources, carrying them through the different stages of the application (these stages are expressed in more detail in section 3.0 Using the app). In its current state, the app can process only the following FHIR resources.

- Observation
- Condition
- Medication
- MedicationOrder

The app will also automatically retrieve, mask, and send any Patient resources that are referenced in the above resources. The app does not provide the ability to process solely Patient resources as they will be masked, and therefore would not provide meaningful data on their own.

2.2 The included example interface and modeled system

Included in the PurpleDragon package are the files "exampleinterface.js" and "index.html", which serve as an example interface which show how the PurpleDragon app can be used. Their use is discussed further in section 3.2 Using the example interface.

- index.html - sets up the HTML web page which the user will use to interact with the app. Provides an interface to begin the resource transfer process.

- exampleinterface.js - Javascript which is used as the back-end for the web interface. Creates the PurpleDragon instances for each modeled client and calls the methods of the instances when the “Process Resources” button is clicked.

In addition, there are also 5 public FHIR servers to model each FHIR server component utilized within the app. They are as follows:

- 3 Population health management system servers.
- 1 Public Health Agency server:
- 1 PurpleDragon Unmasked FHIR server:

To access a web interface for the above servers where the resources in each may be viewed and maintained independently of the app, visit the links as shown in section 4.6.

The interface and system provided by the above components show one potential implementation of the PurpleDragon. The web page created provides a place for each of the population health management systems in the region a simple way to provide the public health agency any data they wish. Simply by providing a list of IDs for resources in their own FHIR server, they are able to send data to the public health agency without worrying about sending duplicate information or maintaining privacy for their patients.

For example, a public health agency could be interested in knowing about patients in the region diagnosed with Type 1 diabetes, and could ask population health management systems in the area to provide them with data.

To do this, the population health management would need to internally determine the Condition resources that exist in their database for Type 1 diabetes. Then, after determining their bundle of resources they would like to contribute, they would only need to enter a list of the IDs into the “Condition” box on the interface, click the process button, and the public health agency would receive their data.

2.2.1 Patient Masking service

This app assumes that it will be integrating with a 3rd party service to properly and securely mask patient information. In its current state, this service is modeled by replacing patient names with random strings and removing other identifying information. The app will need to be updated to properly interact with the service that is ultimately used.

3.0 Solution Architecture

Our solution comprises of six platform implementations, which includes the following components:

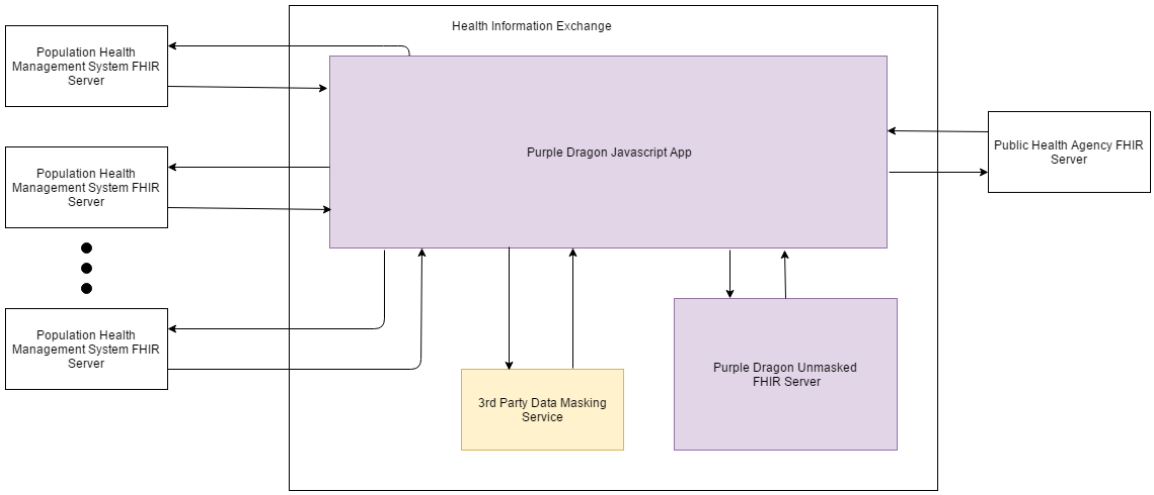
- Purple Dragon App
- Purple Dragon Agency FHIR Server
- Purple Dragon Management FHIR Server One
- Purple Dragon Management FHIR Server Two
- Purple Dragon Management FHIR Server Three
- Purple Dragon Unmasked FHIR Server.

Each of the above components are distinct and separate server implementations.

- The Purple Dragon App & Unmasked FHIR Server are modeled as residing together in an HIE (Health Information Exchanges).
- Each of the Purple Dragon Management FHIR servers models an independently owned and managed FHIR server of a clinic, hospital, doctor's office, etc.
- The Purple Dragon Agency FHIR server models the Public Health Agency server. This server collects all data from the various independently owned FHIR Management servers.
- The Purple Dragon App can be implemented on a single server, but our model installed them on two different server platforms.

3.1 Architecture Diagram

The Architecture implemented is shown below:



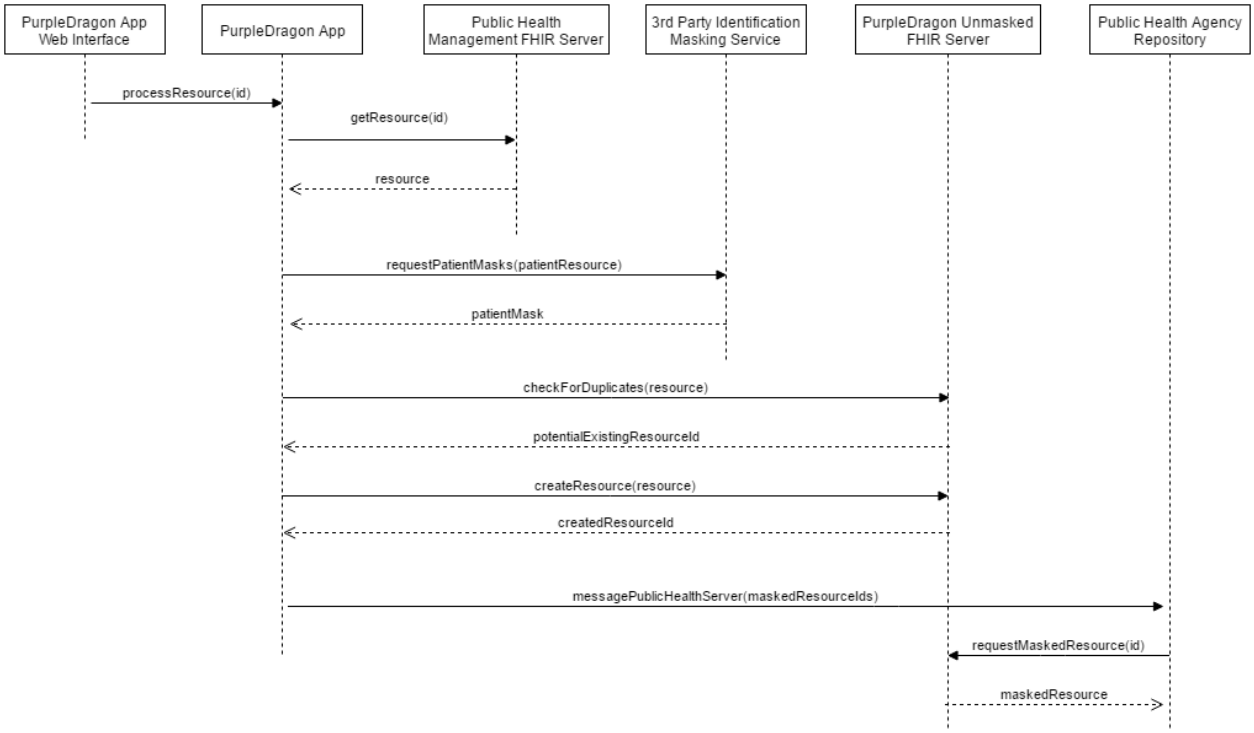
3.2 System Workflow

The system workflow for the solution is described below:

- Purple Dragon App requests relevant data from a select management FHIR servers (i.e. clinical care, hospitals, etc FHIR servers).
- Purple Dragon App server performs de-duplication on all received data, storing any non-previously existing resources in the Purple Dragon Unmasked FHIR server.
- Purple Dragon App sends each Patient resource to the masking service and updates Patient references in other resources.
- Purple Dragon App then constructs a message with masked resources ready to be stored on the Public health agency server. The public health agency then can retrieve the masked and deduplicated data by resource ID.

3.3 Sequence Diagram

The sequence diagram for the implemented solution is shown below. This diagram shows the sequence for the PurpleDragon app processing a single resource.



- Fig 3.3.1: Purple Dragon Sequence Diagram

4.0 Solution Implementation

The architecture of the implementation solution is made of many component parts as indicated in section 3.0 above.

The following components/servers were implemented:

- Purple Dragon App web server
- Purple Dragon Agency FHIR Server
- Purple Dragon Management FHIR Server One
- Purple Dragon Management FHIR Server Two
- Purple Dragon Management FHIR Server Three
- Purple Dragon Unmasked FHIR Server.

4.1 FHIR App

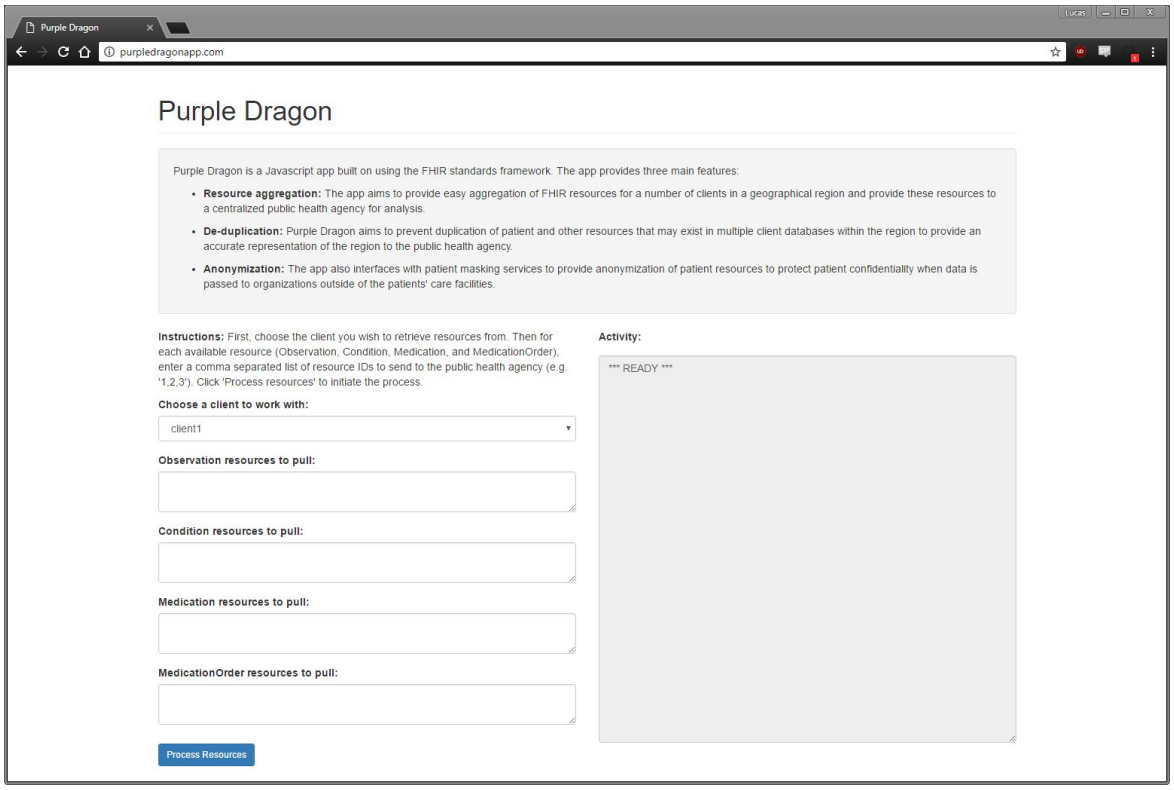
The FHIR App was hosted on Godaddy UNIX hosting platform utilizing port 80 for http access. Apart from the standard libraries provided by Godaddy, none other were implemented.

The URL for the FHIR App is: http://wwww.purpledragonapp.com/

The domain name Purpledragonapp.com was purchased from www.unicpage.com domain name services and redirected to our hosting platform at Godaddy.

The code for the FHIR App can be found on the github pages at: Docs/codes/fhir-app/

When implementing this solution, all the files/folders found in the above directory should be uploaded to the root directory of your web hosting platform.



- Fig 4.1.1: Purple Dragon FHIR App Screen Shot

4.2 Agency FHIR Server

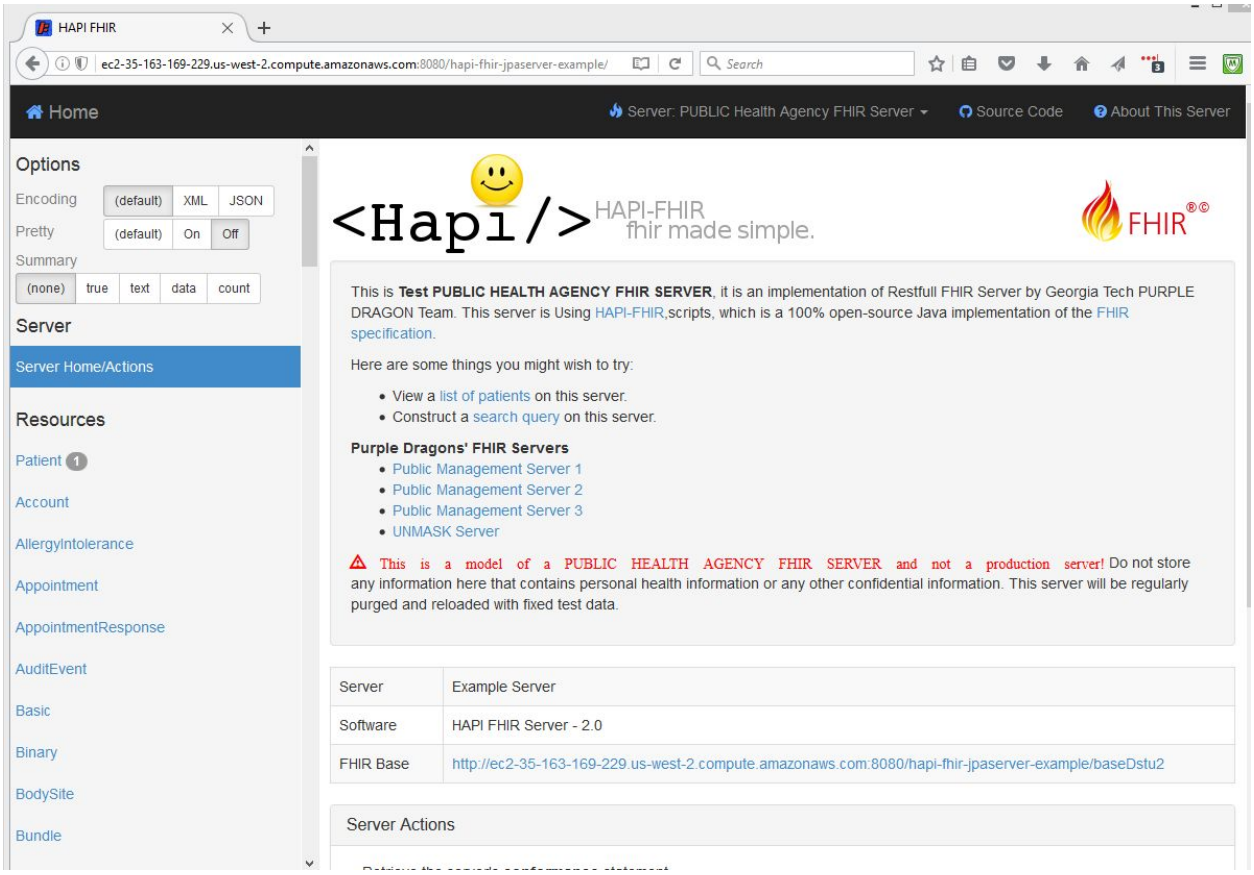
We utilized Amazon EC2 cloud services for the Agency FHIR server, the specifics of the server are:

- Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-a9d276c9, 64-bits
- t2.micro
- Volume size: 8Gig

To implement the Agency FHIR Server:

- Create an instance with the above information.
- set admin password sudo passwd
- Install at least Java 1.8 sudo apt update sudo apt-get install openjdk-8*
- Install maven sudo apt install maven
- Copy the fold hapi-fhir-jpaserver-example from Docs/codes/fhir-agency-server/ to your current ubuntu home directory Use your favorite FTP program to copy this folder or configure samba in ubuntu for this
- Change directory to 'hapi-fhir-jpaserver-example' cd hapi-fhir-jpaserver-example
- Use maven to compile the source code mvn install

- Use 'screen', maven & Jetty to host the FHIR server for persistency screen (press ENTER or SPACE BARD to Escape Screen) mvn jetty:run
- The server takes a while to finish compilation and installation, from implementation, this took about three to five minutes.
- If you are using Amazon, get your Amazon DNS name for this server and browse to e.g. http://localhost:8080/hapi-fhir-jpaserver-example
- Where 'localhost' is your server's Amazon DNS name or local IP address.
- For our impelementation, our Agency server was reached at: http://ec2-35-163-169-229.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/



- Fig 4.2.1: Purple Dragon Agency Server Screenshot
- From the screenshot above, the links to 'Public Management Servers 1, 2, 3 and the UNMASK servers' will not work in your case. You have to edit the following file to change them:
- cd ~/hapi-fhir-jpaserver-example/src/main/webapp/WEB-INF/templates/
- nano tmpl-home-welcome.html

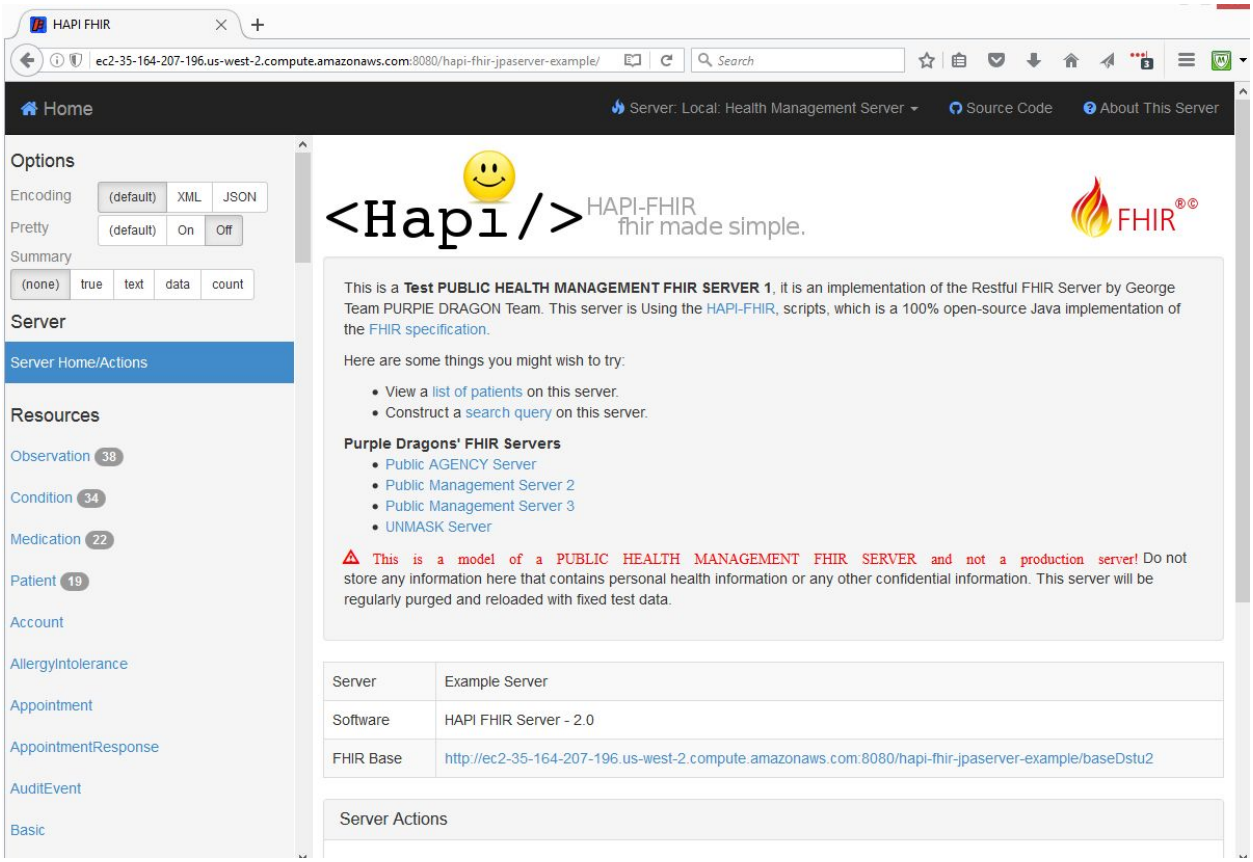
- In directory /hapi-fhir-jpaserver-example, we have renamed our project compiled codes as project-target for reference. When you run mvn install a new directory called target will be created for you.

4.3 Management FHIR server One

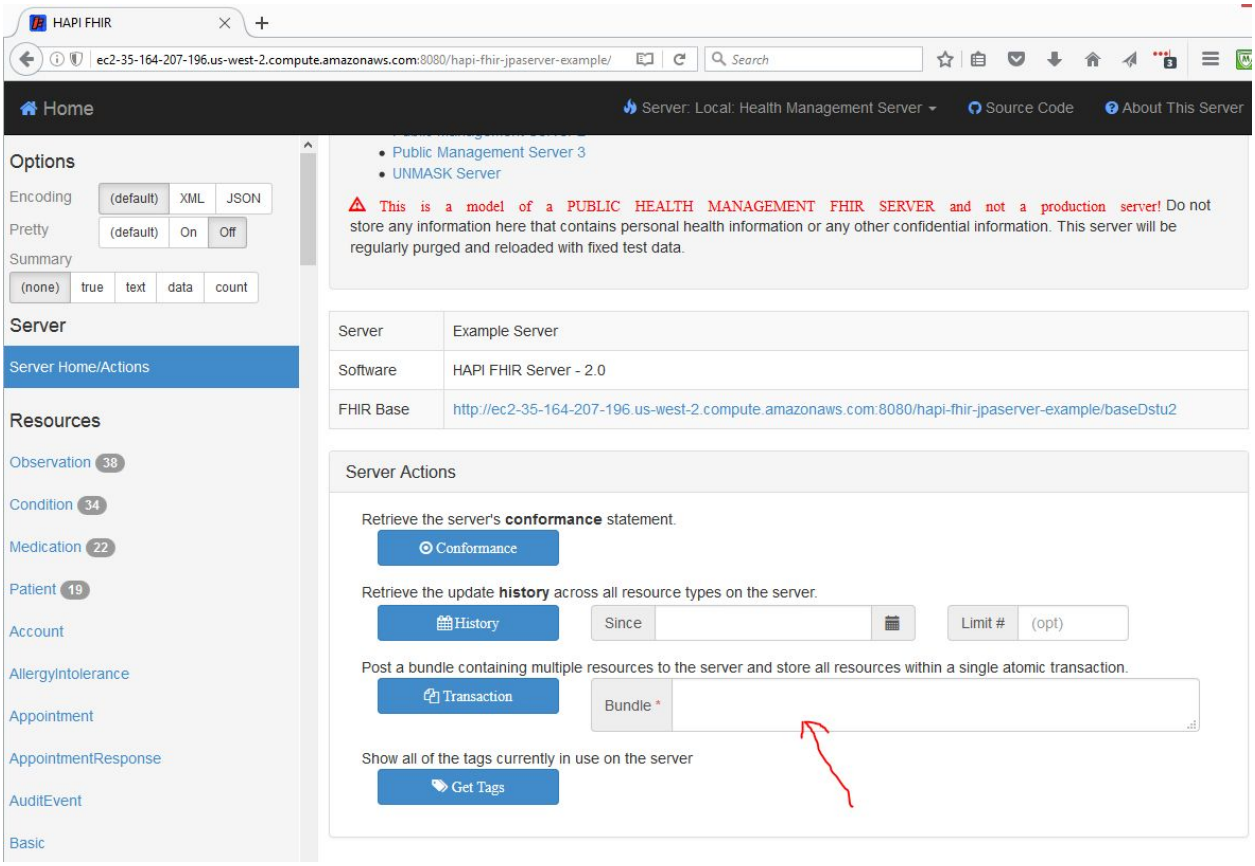
The implementation for Management FHIR One is similar to the FHRI Agency Server above.

However a few files were edited to give this server it's unique look and feel. To get this unique feel:

- Repeats steps from section 4.2 above and stop after implementing the step repeated below for reference:
    - Copy the fold hapi-fhir-jpaserver-example from Docs/codes/fhir-agency-server/ to your current ubuntu home directory Use your favority FTP problem to copy this folder or configure samba in ubuntu for this
- Goto ~/hapi-fhir-jpaserver-example/main/java/ca/uhn/fhir/jpa/demo and rename the file "Fhir TesterConfig.java" to something else e.g. "Fhir TesterConfig.java.OLD"
- Copy file: Docs/codes/management-fhir-one/Fhir TesterConfig.java To: ~/hapi-fhir-jpaserver-example/main/java/ca/uhn/fhir/jpa/demo
- Goto ~/hapi-fhir-jpaserver-example/main/webapp/WEB-INF and rename directory templates to e.g. templates.OLD
- Copy directory: Docs/codes/management-fhir-one/templates To: ~/hapi-fhir-jpaserver-example/main/webapp/WEB-INF
- Change directory to 'hapi-fhir-jpaserver-example' cd hapi-fhir-jpaserver-example
- Use maven to compile the source code mvn install
- Use 'screen', maven & Jetty to host the FHIR server for persistency screen (press ENTER or SPACE BAR to Escape Screen) mvn jetty:run
- The server takes a while to finish compilation and installation, from implementation, this took about three to five minutes.
- If you are using Amazon, get your Amazon DNS name for this server and browse to e.g. http://localhost:8080/hapi-fhir-jpaserver-example
- Where 'localhost' is your server's Amazon DNS name or local IP address.
- For our impelementation, our Agency server was reached at: http://ec2-35-164-207-196.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/



- Fig 4.3.1: Purple Dragon Management Server One Screenshot
- The management server will come with a default one patient resources.
- To populate the server with resources, we wrote some bundle scripts in xml.
- To populate the server with additional patient resources and some Condition, Medication and Observation resources:
- * Goto Docs/codes/fhir-resources/
- * Open the file call patient data bundle.xml with any text editor of your choice.
    - Copy the contents of the file to clipboard and past them in the highlight section of the diagram below on your Management FHIR Server One.

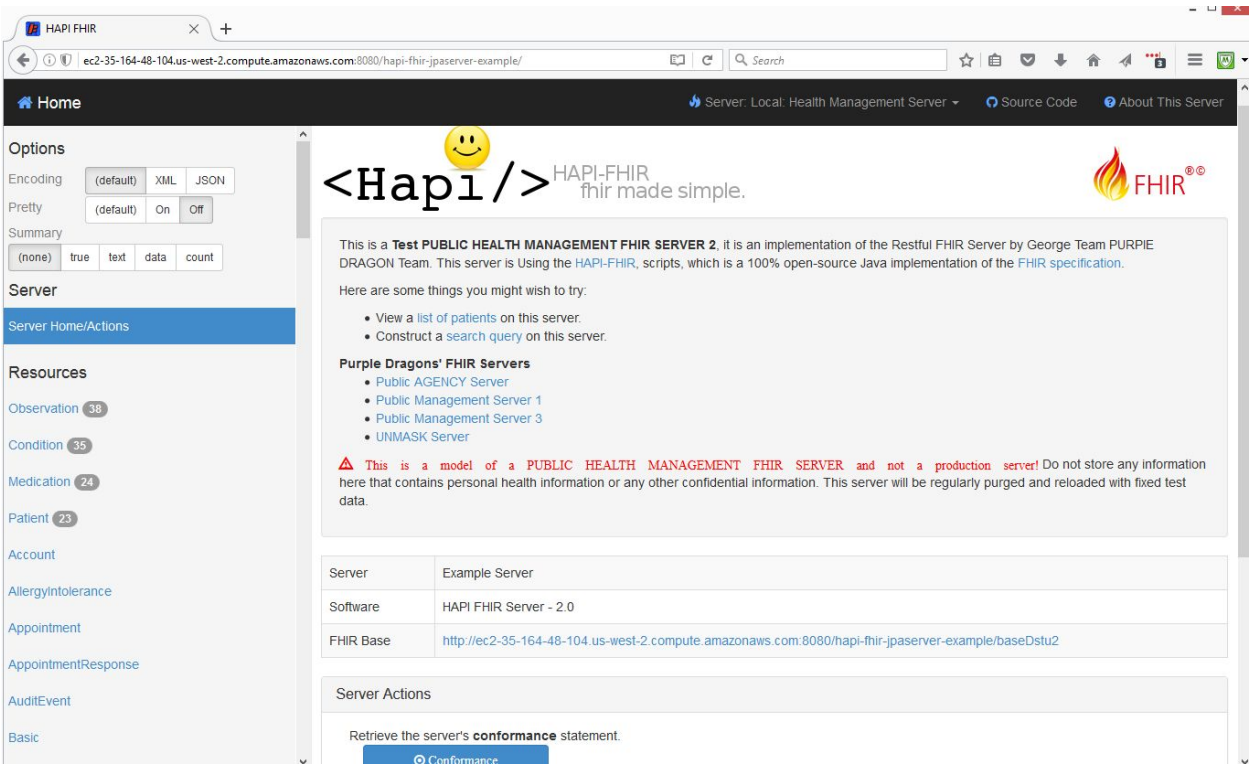


- Fig 4.3.2: Purple Dragon Management FHIR Server Resource Screenshot
- Then click on Transaction to post the data

4.4 Management FHIR server Two

Implementing Management server two and populating it with FHIR resources, is the same steps as illustrated in sections 4.2 and 4.3 above, except that:

- The replacement files/directory contents for Management Server Two can be found at: Docs/codes/management-fhir-two
- The URL of of our Management FHIR Two is: http://ec2-35-164-48-104.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/

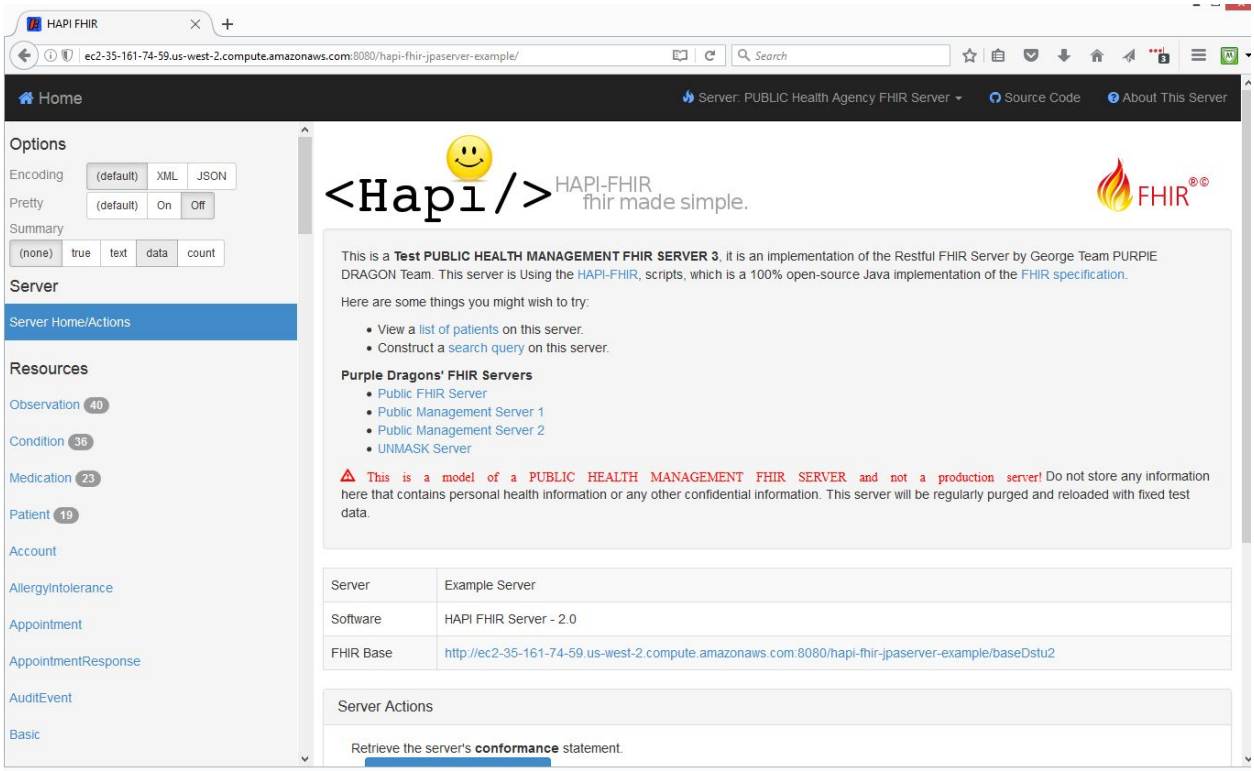


- Figure 4.4.1 Management Server Two Screenshot:

4.5 Management FHIR server Three

Implementing Management server three and populating it with FHIR resources, is the same steps as illustrated in sections 4.2 and 4.3 above, except that:

- The replacement files/directory contents for Management Server Three can be found at: Docs/codes/management-fhir-three
- The URL of of our Management FHIR Three is: http://ec2-35-161-74-59.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/



- Figure 4.5.1 Management Server Three Screen shot:

4.6 FHIR Servers URL Links

As discussed above, we implemented Five FHIR servers and one FHIR Application. To access each of our modeled servers, click on any of the links below.

- Example PurpleDragon web app implementation: http://purpledragonapp.com/
- PurpleDragon Population health Management FHIR Server 1 http://ec2-35-164-207-196.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/
- PurpleDragon Population health Management FHIR Server 2 http://ec2-35-164-48-104.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/
- PurpleDragon Population health Management FHIR Server 3 http://ec2-35-161-74-59.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/
- PurpleDragon Unmasked Resources FHIR Server http://ec2-35-162-112-117.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/

- PurpleDragon Public Health Agency FHIR Server
  http://ec2-35-163-169-229.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/

As this is a modeled project, the servers will eventually go offline. We apologize for any inconvenience this may cause.

5.0 Using the app

5.1 Using purpledragon.js

5.1.1 Installation

To install PurpleDragon, simply move the “purpledragon.js” file onto a web server hosting a web page.

Then, in the html page that is to contain the interface for the app, include a script tag which points to the file’s location on the web server. It will then be able to be used by subsequent scripts on the page.

For example, if the file “purpledragon.js” was placed in a /scripts/ directory at the root of your web server, you could include the app in your web interface’s html file by using the following script tag:

```
<script src="/scripts/purpledragon.js"></script>
```

Note: The purpledragon.js file has a dependency on jQuery. jQuery must be installed on the server and included on the page prior to including the purpledragon.js file or the app will not function properly. For more information on installing and jQuery, visit https://jquery.com/

5.1.2 Instantiating the PurpleDragon class

To make use of the PurpleDragon class, a Javascript variable must be created using the following constructor.

```
var pdInstance = new PurpleDragon(baseUrl, clientUrl, agencyUrl, logElementId);
```

The parameters are as follow:

- **`baseUrl`:** the FHIR base url for the Unmasked PurpleDragon FHIR server used to aggregate unmasked data for de-duplication and re-linking.
- **`clientUrl`:** the FHIR base url for the population health management system FHIR server that this PurpleDragon instance will interact with.
- **`agencyUrl`:** the FHIR base url for the public health agency FHIR server the masked data will be sent to.
- **`logElementId` (optional):** this optional parameter can be used to link the PurpleDragon instance to an HTML document element to provide logging visibility. It is

not required, and would most likely not be used in a production environment as to prevent IDs from each system being shared with each client.

The following code from the exampleinterface.js file shows how the PurpleDragon instances are created in the example interface.

```
var baseUrl =
"http://ec2-35-162-112-117.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/baseDstu2";

var agencyUrl =
'http://ec2-35-163-169-229.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/baseDstu2';

var clientUrl =
'http://ec2-35-164-207-196.us-west-2.compute.amazonaws.com:8080/hapi-fhir-jpaserver-example/baseDstu2'

var pdInstance = new PurpleDragon(baseUrl, clientUrl, agencyUrl, '#pageLog');
```

#pageLog is the id of a textarea html element in index.html that serves as the log in the example interface.

5.1.3 Available methods

The following methods are available within the PurpleDragon instance to transition different FHIR resources through the process to the public health agency:

- `processObservation(id)`
    - Parameter: id - the ID of the Observation resource as it exists in the population health management system the PurpleDragon instance is associated with.
    - Purpose: retrieves the Observation resource from the client, checks for duplication, masks any referenced Patient resources, updates Patient references to the masked resource, and sends the updated Observation resource to the public health agency FHIR server.
- `processCondition(id)`
    - Parameter: id - the ID of the Condition resource as it exists in the population health management system the PurpleDragon instance is associated with.
    - Purpose: retrieves the Condition resource from the client, checks for duplication, masks any referenced Patient resources, updates Patient references to the masked resource, and sends the updated Condition resource to the public health agency FHIR server.

- `processMedication(id)`
    - Parameter: id - the ID of the Medication resource as it exists in the population health management system the PurpleDragon instance is associated with.
    - Purpose: retrieves the Medication resource from the client, checks for duplication, and sends the updated Medication resource to the public health agency FHIR server.
- `processMedicationOrder(id)`
    - Parameter: id - the ID of the MedicationOrder resource as it exists in the population health management system the PurpleDragon instance is associated with.
    - Purpose: retrieves the MedicationOrder resource from the client, checks for duplication, masks any referenced Patient resources, updates Patient references to the masked resource, and sends the updated Condition resource to the public health agency FHIR server.

Each of the functions above can be called in the following manner (assuming a PurpleDragon instance called pdInstance has been properly created, and the associated client has a condition with ID 3257):

```
pdInstance.processCondition(“3257”);
```

Each of the functions above follows the same basic workflow:

1. Retrieve the resource with the provided id from the active client FHIR server.
2. Check the resource for Patient resource references.
3. If there are Patient resources referenced:
    a. Retrieve the Patient resource from the client.
    b. Create a mask for the client using the integrated 3rd party patient masking service.
    c. Create a new Patient resource for the masked patient.
    d. Link the masked Patient resource to the unmasked resource in the Unmasked PurpleDragon FHIR server to be able to re-link data in the future.
    e. Update the Patient references in the resource being processed to the newly created masked resource.

4. Check for data duplicates.
    a. Create a query using important and identifying information from the resource.
    b. Search for the resource in the Unmasked PurpleDragon FHIR server. If an entry is returned, the data is duplicated.
    c. If no resources are found, persist the resource (with updated Patient references if applicable) in the Unmasked PurpleDragon FHIR server.
5. Send the newly created resources to the public health agency FHIR server.
    a. If applicable, send any Patients referenced in the resource as well. Since the resources were updated with the masked references, only masked data will be sent to the public health agency.

5.2 Using the example interface

5.2.1 Modeled System

The web interface for this example has the capability to link to all 3 of the modeled population health management system FHIR servers. Each client has its own instance of the PurpleDragon app, which all utilize the Unmasked PurpleDragon unmasked FHIR server as well as the same public health agency FHIR server.

The links to each server can be found in section 4.6, and can be accessed using the web interface to check what resources exist in each server, and to verify the results of any resources processed.

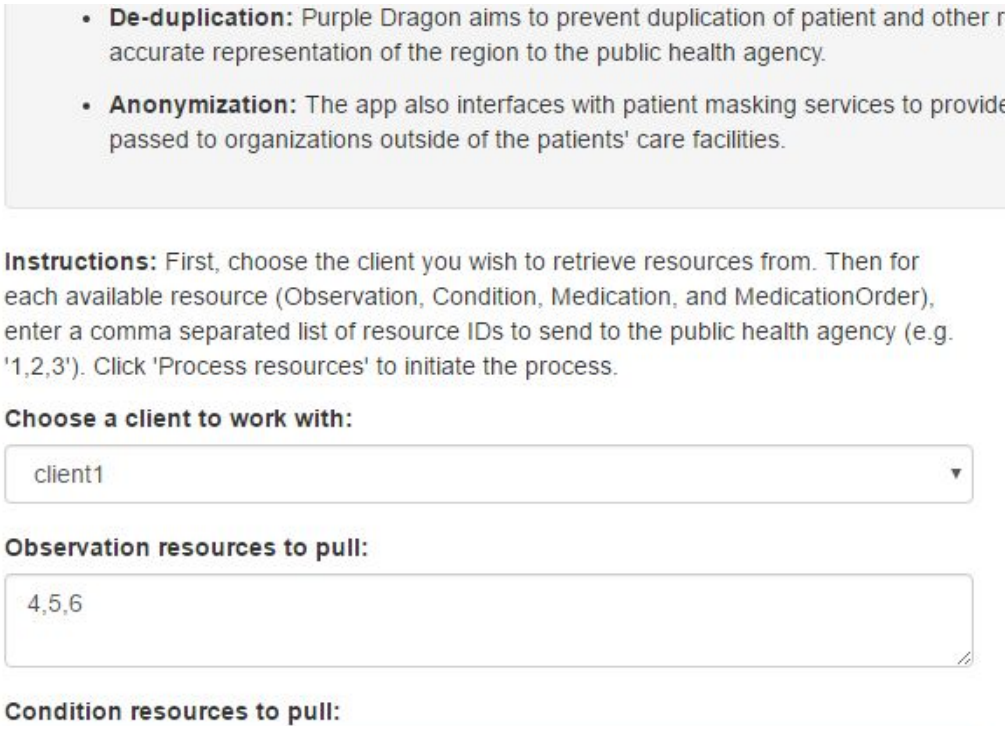
5.2.2 Selecting a client

To begin using the example interface, first select a client from which you wish to pull resources from the provided drop down list. Only one client may be processed at a time.

5.2.3 Inputting Resource IDs

For each resource, input a comma delimited list of ID numbers that you wish to pull. (e.g. “1,2,3”).

Each resource must be individually listed. Multiple IDs can be provided with multiple resources in a single run. (e.g. A single request can include Observations “4, 5, 6” and Conditions “2, 3, 7”)



5.2.4 Begin Processing

Once all the resources you wish to pull are listed, click the process button.

5.2.5 Activity Log

The Activity log can be seen on the right side of the page. This is where details of what the app is doing will be displayed. This is only for the purpose of clarity in the demonstration.