# OMOPonFHIR – FHIR R4 to OMOP v6.0 Docker Deployment

Version 1.0.1 (2021-08-04)

Author: Elizabeth Shivers (Elizabeth.Shivers@gtri.gatech.edu)

## Overview

This guide is intended to aid in the setup of OMOPonFHIR using the R4 version of FHIR and OMOP CDM v6.0 as a base, including the setup of an OMOP CDM v6.0 database. While some technical experience is expected, the guide will attempt to explain what is happening at each step, in order to assist in troubleshooting if a problem does arise, and also to aid in avoidance of common pitfalls.

## Requirements

- Docker/Docker Desktop – Docker is used to host both the OMOP v6.0 database and OMOPonFHIR itself. While fully explaining Docker is outside the scope of this guide, it can be summarized as a means to deploy lightweight self-contained(containerized) environments. (Sometimes considered to be miniature Virtual Machines, though this is not technically precise it is nonetheless a simple way to think of Docker containers.) You can download and familiarize yourself with Docker at https://www.docker.com/. Pay particular note to the difference between a Dockerfile, an image, and a container.
    - For Windows 10 Users, the Windows Ubuntu environment is recommended. It is available in the Windows Store by searching "Ubuntu". For better integration with Docker Desktop, you may also wish to enable WSL2. (WSL2 may also be required on some versions of Windows 10.) PowerShell may also be used, but you will have to adjust your commands accordingly and you may additionally require modification of some scripts to reflect a Windows file system. The Windows Ubuntu environment provides a simpler means to approach this. If you have any issues with Ubuntu for Windows or WSL/WSL2, there are many online resources available to help solve problems around them.

## 1. Preparing the Vocabularies

A core part of OMOP is the use of standardized vocabularies provide by the OHDSI group through the Athena platform. These vocabularies represent many common healthcare coding systems, such as ICD or SNOMED, along with including some additional vocabularies for concepts such as gender.

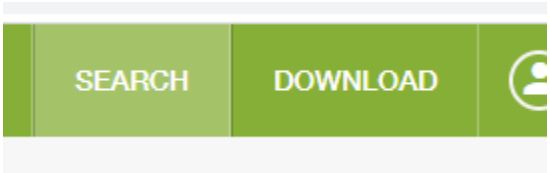
### Download Vocabularies from Athena

The vocabularies you will need for the OMOP database are obtained from the OHDSI Athena platform, located at https://athena.ohdsi.org.

1. Navigate to https://athena.ohdsi.org. You will be asked to accept a license agreement. After doing so, you will need to register a new account if you do not have an existing one. In either

case, press in the Login button in the top right corner. Select the appropriate button here and follow their instructions.

2. Once you have logged in, you will then proceed to obtaining the vocabularies. Start by clicking "Download" in the top bar.




3. This will bring up a list of all current vocabularies. In this list, select the vocabularies you require. (For a list of recommended vocabularies, please see appendix A.)
4. Once you are done choosing your vocabularies, select the "Download Vocabularies" button. This will pop up a confirmation window. At the top you will need to provide a name for your vocabulary set, you may name it whatever you like. To the right of the name, there will be a drop-down box which allows you to select your version of OMOP. Even though we are deploying version 6.0, for this, you will need to select "5.x" from the drop-down list. Scroll to the bottom of the pop-up window and select download.
5. Once Athena is done packaging your vocabularies you will receive an e-mail with a link to download your set of vocabularies. (Note: This may take a few hours.)

## Running CPT4 Script

Once you have the vocabularies downloaded, extract them into their own folder. After you have extracted the files, you will see a series of CSV files as well as a set of files related to the CPT vocabulary (including the readme.txt).

Due to licensing issues with CPT, you are required to have an UMLS account, and so the scripts and JAR are provided to have a way to setup the vocabulary using a key associated with an account. At this point, you will need to go to https://uts.nlm.nih.gov/uts/ and sign up for an account (it is free). You will be prompted to use a sign in service such as Google or Facebook.

*(Note: The shift to a sign on service is new as of January 2021. If you are using an old Athena package of vocabularies, you may need to obtain an updated version of the scripts and JAR that use the API Key instead of a username and password.)*

Once you have your account and are signed in, go to https://uts.nlm.nih.gov//uts.html#profile. You should have an API key shown here.

Now, go to the command line and navigate to the directory you extracted the vocabulary files into if you are not there already. Execute the command…

```
./cpt.sh YOUR_API_KEY
```

…where YOUR_API_KEY is the API Key provided by the UMLS profile page.

Windows users may execute instead "cpt.bat YOUR_API_KEY". If you encounter a permissions issue in Linux environments, first use the command "chmod +x ./cpt.sh" to give the file execution privileges. Please be aware this process may take a while.

*(Note: You may ignore the "sun.reflect.Reflection.getCallerClass" warning if you see it.)*

# 2. Setting up an OMOP v6.0 Database

Setting up your OMOP v6.0 database using a Docker container requires a slightly modified approach. To facilitate the process, modified versions of the OHDSI OMOP v6.0 DDL scripts have been provided.

## Preparation

The modified scripts can be located in the OMOPonFHIR documentation repo at: https://github.com/omoponfhir/omoponfhir-site-n-docs/raw/main/documentation/OMOPonFHIR-OMOPv6-PSQL-Scripts.zip

This zip includes 3 folders:

- f_tables
- OHDSI PostgreSQL DDLs - Fixed
- Vocabulary

The f_tables folder includes scripts needed to build OMOPonFHIR specific tables. The OHDSI PostgreSQL DDLs – Fixed directory includes the OHDSI scripts available from the OHDSI CommonDataModel repository, with a few minor changes to be more streamlined for this particular guide. The Vocabulary folder contains an additional script from the same OHDSI CommonDataModel repository for the purpose of importing the vocabulary into the OMOP v6.0 database, which has been modified to work using an external PSQL client connecting to a Docker container.

Now, **move your vocabulary CSV files** (from the first step of this guide) into the Vocabulary folder. They should be at the same level as the script already inside the folder. This is to allow the script to make a reference to the current location in the directory structure and avoid requiring any additional editing of the script.

## Deploy Postgres in Docker

To begin, you will need to pull the latest PostgreSQL image from Docker Hub. (Docker Hub provides a central repository of Docker images.)

```
docker pull postgres
```

Then run the postgres image using the following command.

```
docker run --name omop_v6 -p 5432:5432 -e POSTGRES_PASSWORD=password -d
postgres
```

This gives our container a name, specifies the port mapping from your localhost to the container, and passes the required password setting as an environment variable. (The -d flag runs this in detached mode. This prevents the container logs from taking over your terminal window. If you wish to have the logs, omit this tag and open a second terminal window to work from. Note these logs are also available through Docker Desktop, so for most users this would be unneeded.)

You can modify the value of the password if desired, though you will need to adjust the following steps and OMOPonFHIR configuration appropriately to reflect the password you select. There is also a chance you may need to change the ports because of a conflict. If so, modify the first number following the -p tag appropriately. It is recommended that you attempt this in increments of one. For example, using "-p 5433:5432". You should not change the second value, as this is the port of the localhost ***inside*** the

container and will not conflict with your host environment's ports. **If you do need to change this value, you will need to adjust all following steps whenever a port is referenced, including in the OMOPonFHIR environment variables.**

Once your container is up and running, you'll need to create the database and results schema. The following command will run a series of psql commands inside the container as the postgres user. Each -c parameter's argument is a separate command given in order. This will first create a database called "omop", switch to that database, and then create a schema called "results". (The results schema will be used for some additional tables beyond what will go into the default public schema.)

```
docker exec -it omop_v6 psql -U postgres -c "CREATE DATABASE omop;" -c "\c
omop" -c "CREATE SCHEMA results;"
```

While the terminal should indicate the success of each step, if you wish to confirm this executed properly it is recommended you use a tool such as PG Admin (https://www.pgadmin.org/). You can connect to your database at localhost:5432 (or whatever port you specified previously), with the username "postgres" and a password of "password" (or whatever you changed it to).

## OMOPv6 DDLs

For this step you will setup the OMOP CDM v6 Database. For the official repository and original DDL scripts you can visit https://github.com/OHDSI/CommonDataModel, though for this guide we will be using the scripts provided as part of the supporting repository cloned previously.

To begin, navigate into the directory you cloned the repository into, and then move into the "OHDSI PostgreSQL DDLs - Fixed" subdirectory. Here you will find slightly modified versions of the OHDSI scripts to work out of the box with this deployment guide around Docker.

The first script you should run is the OMOP CDM postgresql DDL to setup the tables. The following command will allow you to connect directly to your running PostgreSQL container from the host environment command line. The password will be whatever you defined previously, which if you are using the default values in this guide is simply "password".

```
psql -U postgres -h localhost -p 5432 -d omop -f "OMOP CDM postgresql ddl.txt"
```

While this guide does not include loading any existing data into your OMOP CDM database, if you have some you wish to use this is the point you would do so. For example, it is common to preload a OMOP database with data from the CMS Synpuf dataset (https://www.cms.gov/Research-Statistics-Data-and-Systems/Downloadable-Public-Use-Files/SynPUFs/DE_Syn_PUF) or generated by Synthea, a synthetic records generator (https://github.com/synthetichealth/synthea). (Note: Much available data is not naturally provided in OMOP v6, so your data may require some level of ETL. Synthea offers a guide for this for their generated data at https://github.com/OHDSI/ETL-Synthea.)

As per the OHDSI primary repository, loading data prior to indexing is recommended for the sake of speed (https://github.com/OHDSI/CommonDataModel/blob/master/PostgreSQL/README.md).

Once you have your data loaded or have decided to proceed without doing so, you may execute the next script from the preconfigured OHDSI files to provide indexing as below:

```
psql -U postgres -h localhost -p 5432 -d omop -f "OMOP CDM postgresql pk
indexes.txt"
```

You may follow the same procedure with the constraint script, but this is not required. For development work with OMOPonFHIR, it may even be undesirable.

At this point, your database itself is mostly ready to go. However, to support portions of the OHDSI stack and future OMOPonFHIR development it is also recommended to setup the Results schema that you created in a previous step.

To do so, execute the following command to run the Results script:

```
psql -U postgres -h localhost -p 5432 -d omop -f "OMOP CDM Results postgresql
ddl.txt"
```

## Load the Vocabularies

Now navigate into the Vocabulary directory in the repository. If you did not in the CPT4 step, make sure you have your vocabulary CSVs moved into this folder, then run the following command:

```
psql -U postgres -h localhost -p 5432 -d omop -f "OMOP CDM vocabulary load -
PostgreSQL.sql"
```

Be aware this step will take some time, potentially as long as a few hours. You may need to increase the available memory or disk image for your containers if you run into any issues. (For Docker Desktop users, you may do so in the Configuration menu under “Resources” > “Advanced”.)

## Create OMOPonFHIR F_Person Table and F_Observation_View

Because it is focused on de-identified population level research, OMOP CDM is lacking some core fields expected in FHIR elements for patients. To address this, OMOPonFHIR utilizes a custom F_Person table. For preloaded de-identified data, there is also a script that will build randomized names.

It also includes an F_Observation view to handle discrepancies in how observation style data is handled between FHIR and OMOP.

To set these up, return to the main repository directory and then navigate to the f_tables sub directory. Run the following two commands:

```
psql -U postgres -h localhost -p 5432 -d omop -f
"omoponfhir_f_person_table_ddl.txt"

psql -U postgres -h localhost -p 5432 -d omop -f
"omoponfhir_v6.0_f_observation_view_ddl.txt"
```

Now it is time to setup the potential randomly selected names. This step will require working within the container for the sake of simplicity, as it can be troublesome to manage Postgres dump files within a container from the host environment.

First, copy the names.dmp file into the container’s root directory:

```
docker cp names.dmp omop_v6:/
```

Now you will run a series of commands to load the command line within your container, switch to the postgres user, and execute the dump.

```
docker exec -it omop_v6 /bin/bash
```

```
su postgres

psql omop < names.dmp
```

Once executed, you should see a series of insertions into the database. This step may take a few minutes.

To provide some explanation for the final command, we are using the psql tool to import a previously created database of possible names that was output into the names.dmp file. Here “omop” is a reference to our previously created database. The < is an infile indicator for the names.dmp file.

Now, if you need to, you may run the SQL script to build names for your existing data as follows:

```
psql -U postgres -h localhost -p 5432 -d omop -f "insert_names_to_f_person.sql"
```

Your OMOPonFHIR ready OMOP CDM v6 database should now be complete!

# 3. Deploying OMOPonFHIR with Docker

## Cloning the OMOPonFHIR Repository

In a different location from the previous repository, clone the OMOPonFHIR main repo including submodules with the command:

```
git clone --recurse-submodules https://github.com/omoponfhir/omoponfhir-
main.git
```

## Creating a Docker Network

For this deployment we will use a docker network to allow our database and OMOPonFHIR to connect readily. Set this up with the following command:

```
docker network create omop
```

Then add the existing OMOP v6 container to this network with:

```
docker network connect omop omop_v6
```

## Configuring OMOPonFHIR

To configure OMOPonFHIR’s environment variables to match your deployment, you will need to set them in the Dockerfile. The Dockerfile can be found in the root of the OMOPonFHIR directory. Open it in a plain text capable editor of your choice to avoid potential formatting mishaps (Notepad++, Atom, VS Code, etc.).

In the Dockerfile line 9 will have a comment noting where to add environment variables, with line 10 providing a sample JDBC connection for the OMOP database. Uncomment out the JDBC line and change it to the following:

```
ENV JDBC_URL=jdbc:postgresql://omop_v6:5432/omop JDBC_USERNAME=postgres
JDBC_PASSWORD=password
```

Note that each part of this setting will depend on your configuration thus far. The “omop_v6” is a reference to your OMOP container name which is us here as part of the Docker network. (In a non-containerized deployment, this will more likely be something along the lines of “localhost”.) :5432 is

whatever port you configured. The username will typically be static and simply remain “postgres” in almost any case, but if you modified your postgres password from the default in this guide of “password”, change it here.

You will also need to add the lines below, setting the AUTH_BASIC to your desired combination of username:password for the OMOPonFHIR server itself (which will be passed when you make a request for the FHIR resources so may be public facing). FHIR_READONLY controls whether or not you can write to the omop database through the OMOPonFHIR server, and defaults to “True” (in that the database **IS** Read-Only.) For the purposes of this guide, we will assume you wish to write to your database and set this value to “False”. You are free to omit the FHIR_READONLY line if you wish to keep the server set to read only, such as if you are only interested in working with existing data sets in an OMOP to FHIR flow and not concerned with converting your own FHIR resources to OMOP.

```
ENV AUTH_BASIC="client:secret"

ENV FHIR_READONLY="False"
```

## Building and Running the Container

Once you have your environment variables configured, it is time to build and run the container. Start by executing the following docker command to create your OMOPonFHIR image. Do not neglect the “.” at the end of the line, as this is giving the current directory as the context for the location of the Dockerfile.

```
sudo docker build -t omoponfhir .
```

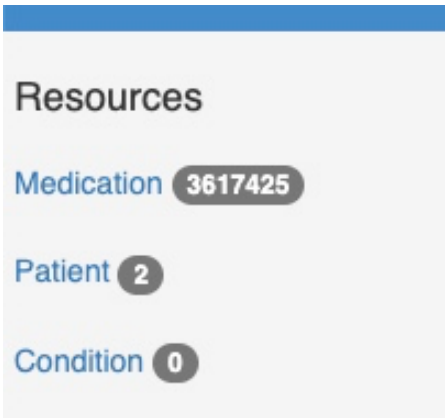
Once this is built, it is time to run the container. Note that instead of adding this container to the omop docker network later, it is being attached to the network on run.

```
sudo docker run --name omoponfhir --network=omop -p 8080:8080 -d
omoponfhir:latest
```

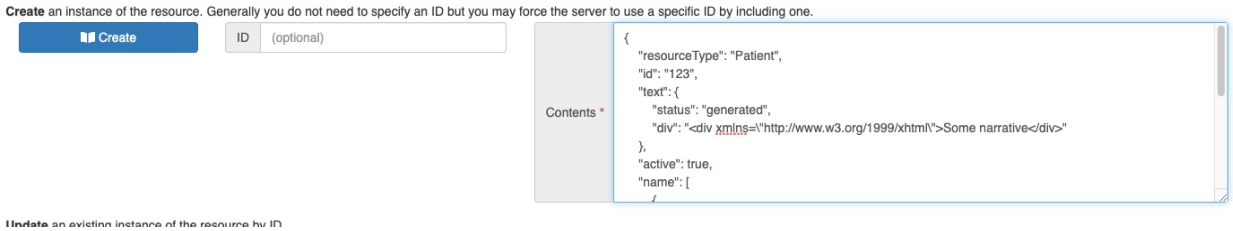
If everything was setup properly, you should now have a fully functional OMOPonFHIR deployment!

## Using and Testing your OMOPonFHIR Deployment

You may connect to the OMOPonFHIR UI at [http://localhost:8080/omoponfhir4/](http://localhost:8080/omoponfhir4/). If you preloaded any data, you should see the related resources enumerated in the list on the left by resource type. Even if you did not, you should still have a large number of Medication resources from loading vocabularies.



You can test the write capability of the server with any simple resource through the UI. A small patient resource from the FHIR Specification site examples is given in Appendix B. In the UI, select “Patient” in the left hand column pictured above. Select the “CRUD Operations” tab. In the “Create an instance” section, paste the resource from the appendix into the contents text box. When ready, press the “Create” button.




If you are successful, the server will return an HTTP 201 response in the UI. You can then access your new resource via the link at the top or by navigating directly to http://localhost:8080/omoponfhir4/fhir/Patient. Note that this provides a bundle response of all patients, not just your test patient.

## Appendix A – Recommended Athena Vocabularies

1. SNOMED
2. ICD9CM
3. ICD9Proc
4. CPT4
5. HCPCS
6. LOINC
7. RxNorm
8. NDC
9. Gender (OMOP Gender)
10. Race (Race and Ethnicity Code Set (USBC))
11. CMS Place of Service (Place of Service Codes for Professional Claims (CMS))
12. ATC
13. Revenue Code (UB04/CMS1450 Revenue Codes(CMS))
14. Ethnicity (OMOP Ethnicity)
15. NUCC (National Uniform Claim Committee Health Care Provider Taxonomy Code Set (NUCC))
16. Medicare Specialty (Medicare provider/supplier specialty code (CMS))
17. SPL (Structured Product Labeling (FDA))
18. Currency (International Classification of Diseases, Tenth Revision, Clinical Modification (NCHS))
19. ICD10CM
20. ABMS (Provider Specialty (American Board of Medical Specialties)
21. RxNorm Extension (RxNorm Extension OHDSI)
22. OMOP Extension (OMOP Extension (OHDSI))

## Appendix B - Simple Patient JSON

```
{
```

```
    "resourceType": "Patient",
    "id": "123",
    "text": {
        "status": "generated",
        "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Some
narrative</div>"
    },
    "active": true,
    "name": [
        {
            "use": "official",
            "family": "Chalmers",
            "given": [
                "Peter",
                "James"
            ]
        }
    ],
    "gender": "male",
    "birthDate": "1974-12-25"
}
```