

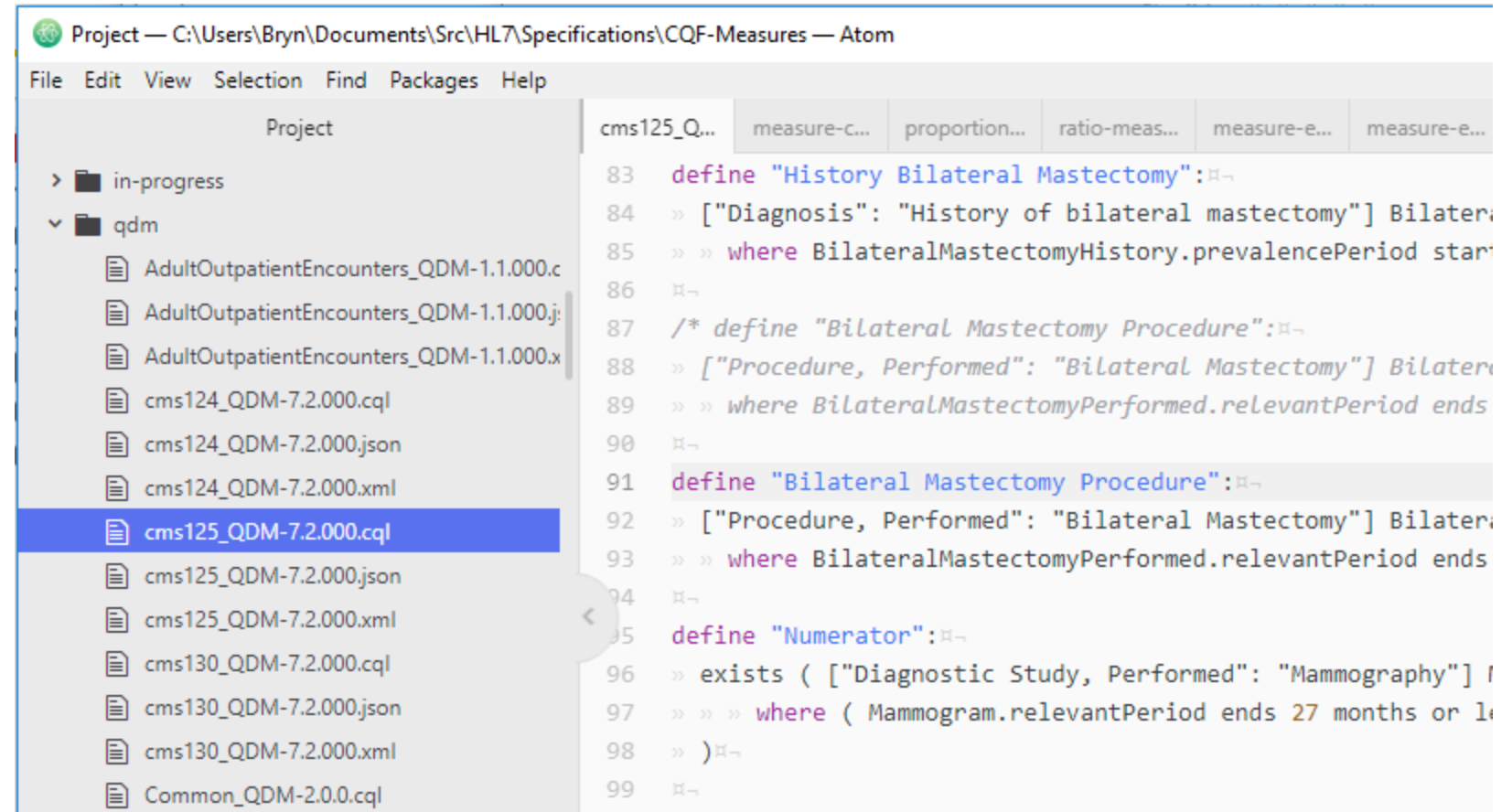
Getting Started with Clinical Quality Language (CQL)

Topics

- Out-of-the-box Tools
- Resources
- Implementation Tools

Atom – Best editor ever

- <https://atom.io/>
- Simple text editor
- CQL highlighter



Project — C:\Users\Bryn\Documents\Src\HL7\Specifications\CQF-Measures — Atom

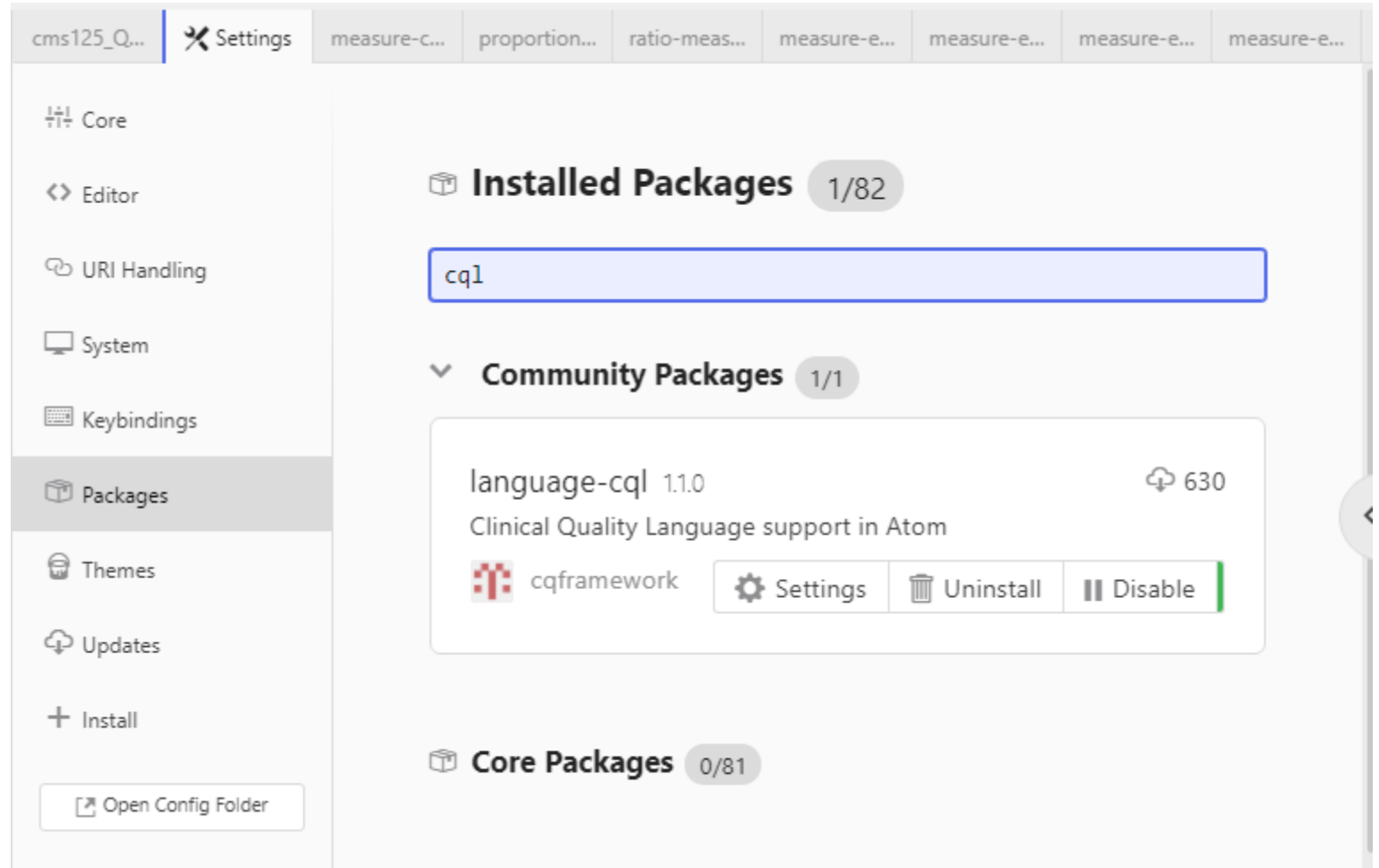
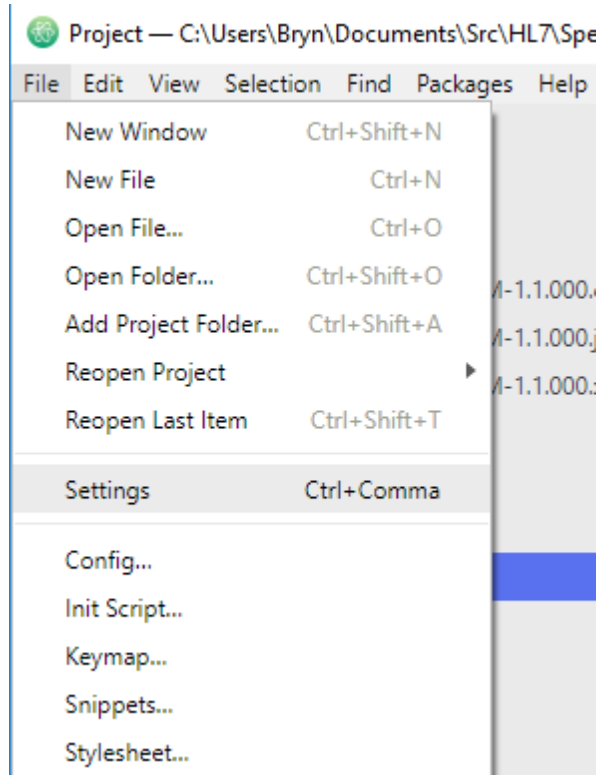
File Edit View Selection Find Packages Help

Project

- > in-progress
- ▼ qdm
 - AdultOutpatientEncounters_QDM-1.1.000.c
 - AdultOutpatientEncounters_QDM-1.1.000.j
 - AdultOutpatientEncounters_QDM-1.1.000.x
 - cms124_QDM-7.2.000.cql
 - cms124_QDM-7.2.000.json
 - cms124_QDM-7.2.000.xml
 - cms125_QDM-7.2.000.cql**
 - cms125_QDM-7.2.000.json
 - cms125_QDM-7.2.000.xml
 - cms130_QDM-7.2.000.cql
 - cms130_QDM-7.2.000.json
 - cms130_QDM-7.2.000.xml
 - Common_QDM-2.0.0.cql

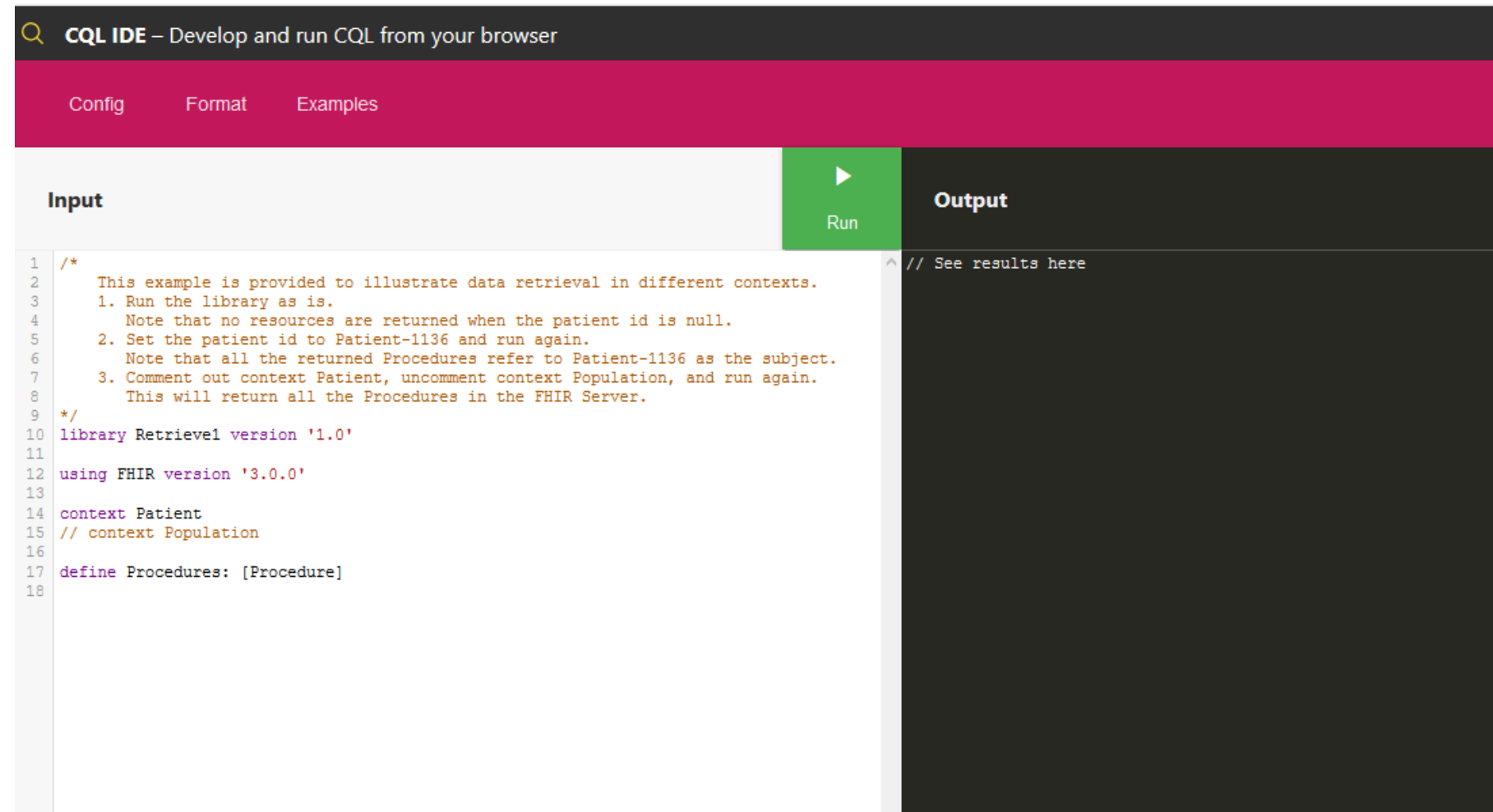
```
83 define "History Bilateral Mastectomy":#-
84   » ["Diagnosis": "History of bilateral mastectomy"] Bilater
85   » » where BilateralMastectomyHistory.prevalencePeriod start
86   #-
87   /* define "Bilateral Mastectomy Procedure":#-
88   » ["Procedure, Performed": "Bilateral Mastectomy"] Bilater
89   » » where BilateralMastectomyPerformed.relevantPeriod ends
90   #-
91   define "Bilateral Mastectomy Procedure":#-
92   » ["Procedure, Performed": "Bilateral Mastectomy"] Bilater
93   » » where BilateralMastectomyPerformed.relevantPeriod ends
94   #-
95   define "Numerator":#-
96   » exists ( ["Diagnostic Study, Performed": "Mammography"] /
97   » » » where ( Mammogram.relevantPeriod ends 27 months or 1
98   » » )#-
99   #-
```

Atom – Package settings



CQL-Runner – Ad-hoc CQL execution

- <http://cql-runner.dataphoria.org>

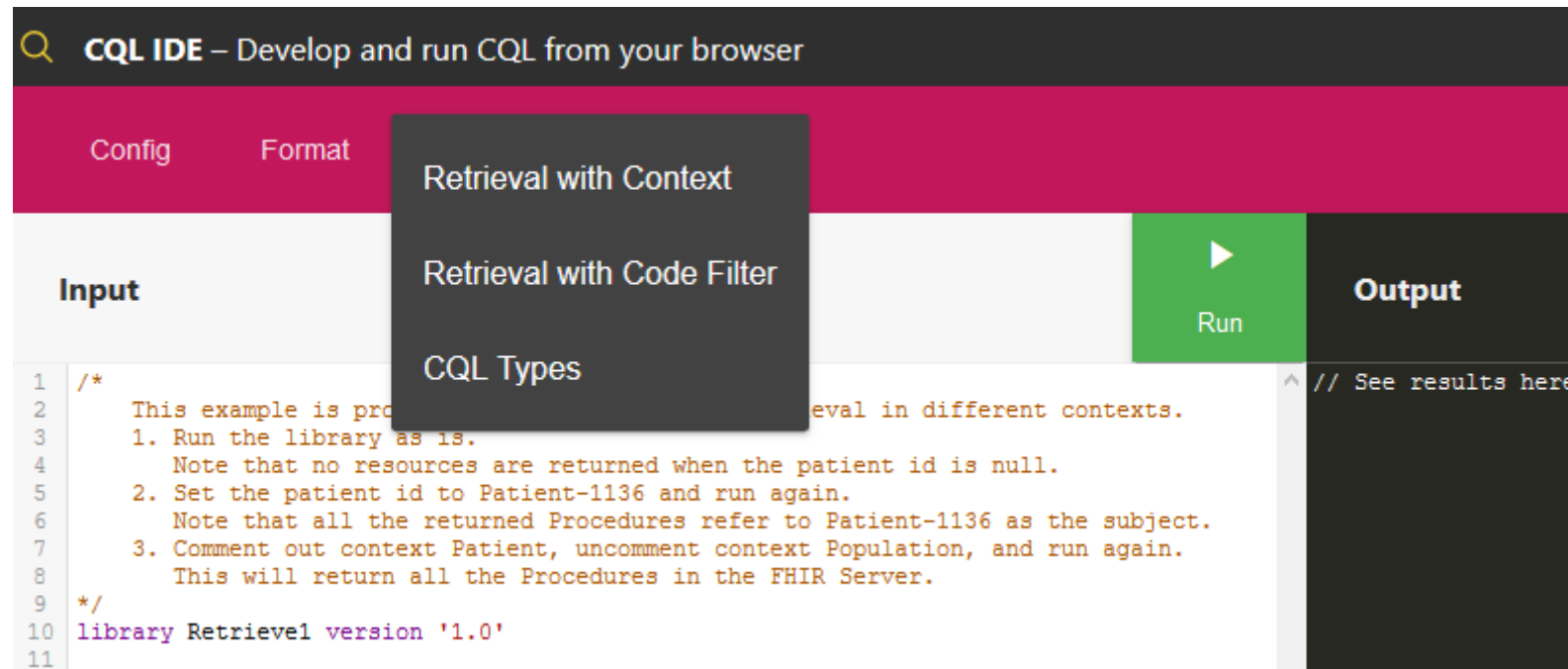


The screenshot shows the CQL IDE web application interface. At the top, there is a search bar and the text "CQL IDE – Develop and run CQL from your browser". Below this, there are three navigation links: "Config", "Format", and "Examples". The main interface is divided into two sections: "Input" and "Output". The "Input" section contains a code editor with the following CQL code:

```
1  /*
2   This example is provided to illustrate data retrieval in different contexts.
3   1. Run the library as is.
4   Note that no resources are returned when the patient id is null.
5   2. Set the patient id to Patient-1136 and run again.
6   Note that all the returned Procedures refer to Patient-1136 as the subject.
7   3. Comment out context Patient, uncomment context Population, and run again.
8   This will return all the Procedures in the FHIR Server.
9  */
10 library Retrieval version '1.0'
11
12 using FHIR version '3.0.0'
13
14 context Patient
15 // context Population
16
17 define Procedures: [Procedure]
18
```

The "Output" section is currently empty, with a placeholder text "// See results here". A green "Run" button is located between the "Input" and "Output" sections.

CQL-Runner – Examples



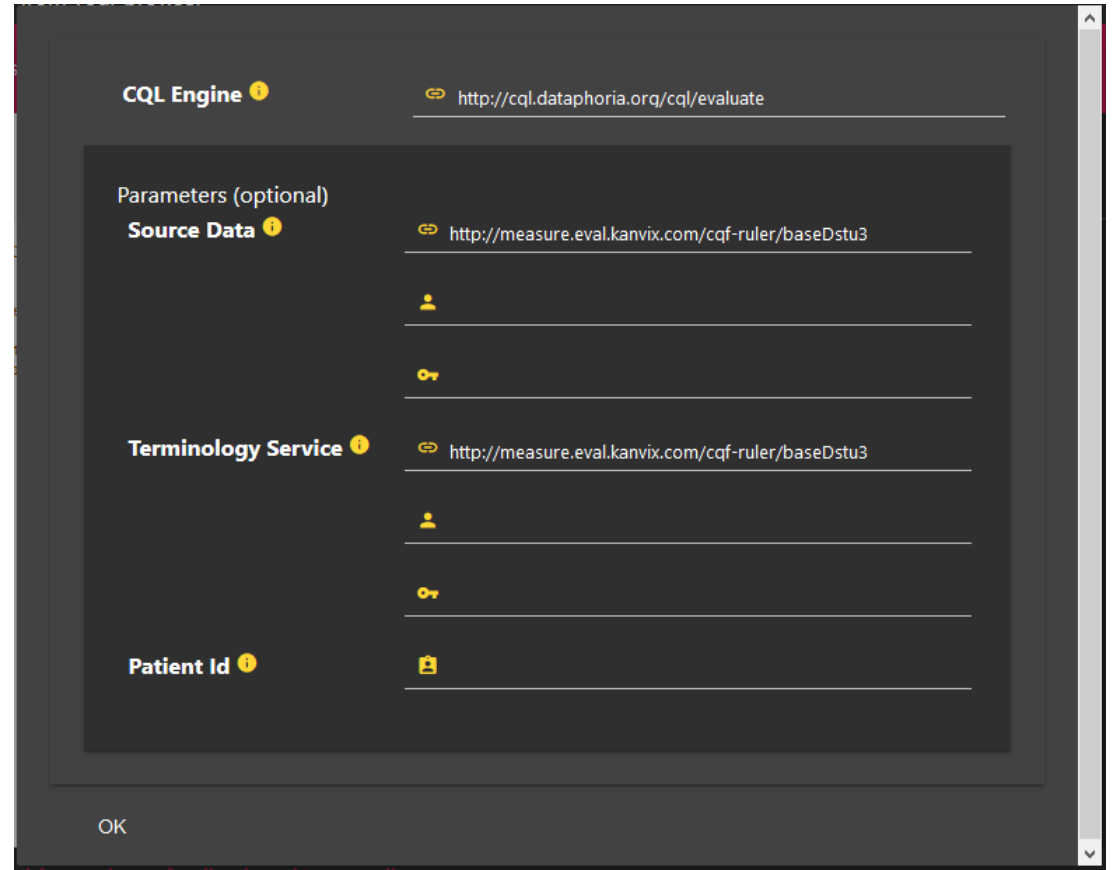
The screenshot displays the CQL IDE web interface. At the top, a search icon and the text "CQL IDE – Develop and run CQL from your browser" are visible. Below this is a navigation bar with "Config" and "Format" buttons. A dropdown menu is open, showing three options: "Retrieval with Context", "Retrieval with Code Filter", and "CQL Types". To the right of the dropdown is a green "Run" button with a play icon. The main area is split into "Input" and "Output" sections. The "Input" section contains a code editor with the following text:

```
1 /*
2   This example is provided to demonstrate how CQL can be eval in different contexts.
3   1. Run the library as is.
4   Note that no resources are returned when the patient id is null.
5   2. Set the patient id to Patient-1136 and run again.
6   Note that all the returned Procedures refer to Patient-1136 as the subject.
7   3. Comment out context Patient, uncomment context Population, and run again.
8   This will return all the Procedures in the FHIR Server.
9 */
10 library Retrieval version '1.0'
11
```

The "Output" section is currently empty, with a placeholder comment "// See results here" at the top.

CQL-Runner – Config

- CQL Engine – leave it
- Source Data – What FHIR Server?
- Terminology – Usually the same
- Supports basic auth
- Patient Id – Which patient?



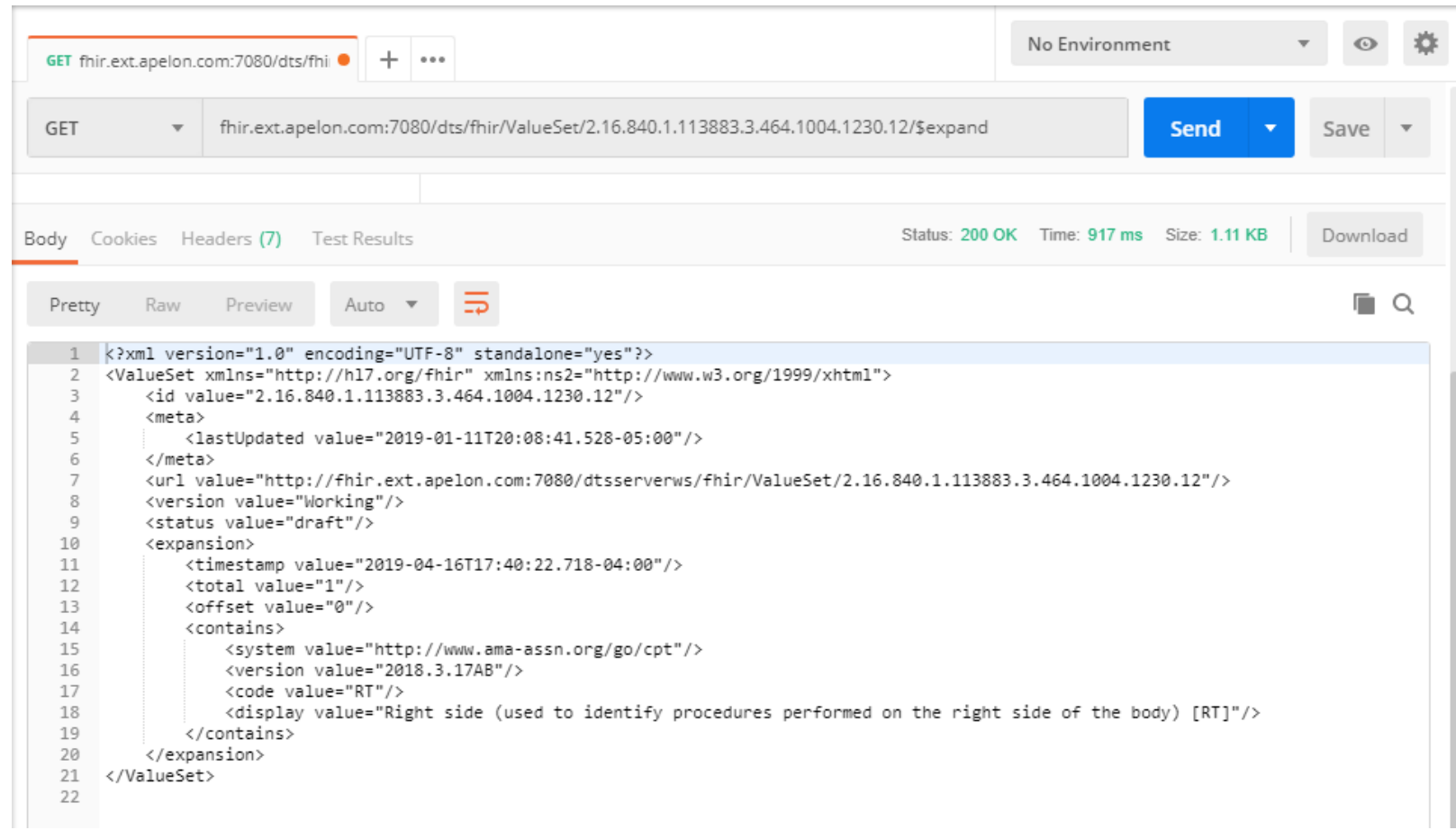
The screenshot shows a configuration dialog box for CQL-Runner. It has a dark background with white text and icons. The dialog is titled "CQL Engine" with a yellow information icon. Below the title, there are four sections, each with a label and a yellow information icon:

- CQL Engine**: <http://cql.dataphoria.org/cql/evaluate>
- Parameters (optional)**
 - Source Data**: <http://measure.eval.kanvix.com/cqf-ruler/baseDstu3>
 - Username field (with person icon)
 - Password field (with key icon)
- Terminology Service**: <http://measure.eval.kanvix.com/cqf-ruler/baseDstu3>
- Username field (with person icon)
- Password field (with key icon)

At the bottom, there is a **Patient Id** field with a yellow information icon and a patient icon. An "OK" button is located at the bottom left of the dialog.

Postman – REST API Testing

- <https://www.getpostman.com/>



clinFHIR

- <http://clinfhir.com/>

clinFHIR Launcher ↶ ⚙

Main modules (open in new tab) **Experimental modules (open in new tab)**

Patient Viewer	Display resources for a specific patient, using a number of different views such as a list by resource type, json & tree views, encounters by condition, numeric Observation charting and graphical relationship views. There is also the option to add a new patient, and to create sample data for that patient.	Patient resources are stored on the Data Server. The server should support the Patient/\$everything operation.
Server Query	Supports ad hoc queries against any FHIR server. Includes a simple query builder. The response can be displayed as Json or a Tree view, and FHIRPath is supported.	Can access any compliant FHIR server (must expose a Capability Statement)
Scenario Builder	The Scenario Builder is used to join together the resources needed to represent a specific clinical scenario. It can use Core Resource types, Profiles and Logical models as it does this. The intention is to help people understand how resources can tell a clinical story, and to validate that the resource types available (including profiles) are sufficient. Note that the builder still has issues with more complex resource types - this is a work in progress	Patient information is on the Data Server. Profiles on the Conformance server. ValueSets on the Terminology server. Create a simple scenario Adding structured data to a scenario Create a Document
Logical Modeller	The Logical modeller allows the creation of a model that represents a particular interoperability requirement in a format that is easy to use. It uses FHIR datatypes, and can be based on an existing resource type or completely 'ad hoc'. It is intended to act as a 'bridge' between Modeller and User, and can act as the basis for the generation of the profiling components required by FHIR	Models are saved on the Conformance Server. Can reference ValueSets from the Terminology server. Create an Information Model Create a Resources Model
Implementation Guide Browser	Display the contents of an Implementation Guide, and the relationships between the contents of the Guide.	The Implementation guide, profiles and Extension Definitions are on the Conformance Server, the terminology resources (eg ValueSet) are on the

Current servers Edit

Data Server	Public HAPI STU3 server	?
Conformance Server	Public HAPI STU3 server	?
Terminology Server	Ontoserver (terminology)	?

[Add Server](#) [Set all the same as the Data Server](#)

FHIR Links (open in new tab)

STU-3 (R3) Specification	Hay on FHIR
STU-2 Specification	FHIR Chat
FHIR wiki	FHIR.org
	Clinicians Workshop

clinFHIR Videos (open in new tab)

[Scenario Builder](#)
[Adding structured data](#)
[Logical Modeller](#)
[Logical Modeller and Scenario Builder](#)
[RESTful query tool](#)

Note that some of these videos may describe earlier versions, so may not completely match the current functionality.

Other links

[SNOMED browser](#)
[SHRIMP \(Terminology browser\)](#)

Resources

- Clinical Quality Language
 - <http://cql.hl7.org>
- FHIR STU3
 - <http://hl7.org/fhir/STU3>
- QI-Core Profiles
 - <http://hl7.org/fhir/us/qicore>
- Quality Data Model (QDM) to QI Core Mapping
 - <http://hl7.org/fhir/us/qicore/qdm-to-qicore.html>
- QUICK
 - <http://hl7.org/fhir/us/qicore/quick/QUICK-index.html>

More Resources!

- Authoring Measures in CQL
 - <https://github.com/esacinc/CQL-Formatting-and-Usage-Wiki/wiki/Authoring-Measures-in-CQL>
- FHIR Measure Implementation Guide
 - <http://hl7.org/fhir/us/cqfmeasures/>
- Data Exchange for Quality Measures
 - <http://hl7.org/fhir/us/davinci%2Ddeqm/2019May/STU3/>
- HEDIS Implementation Guide
 - <http://build.fhir.org/ig/cqframework/hedis-ig>

Implementation Tools – (!faint_of_heart)

- CQL-to-ELM Translator
 - https://github.com/cqframework/clinical_quality_language/blob/master/Src/java/cql-to-elm/OVERVIEW.md
- JS CQL Execution Engine
 - <https://github.com/cqframework/cql-execution>
- Java CQL Execution
 - https://github.com/dbcg/cql_engine
- CQF Ruler
 - <https://github.com/DBCG/cqf-ruler>
- CQL-to-SQL Translation
 - <https://github.com/cqframework/healthdecisions>