



Clinical Quality Language (CQL): Training for Measure Implementers

June 22, 2016

4:00 PM EDT

Deborah Krauss

Centers for Medicare & Medicaid Services (CMS)

Bryn Rhodes

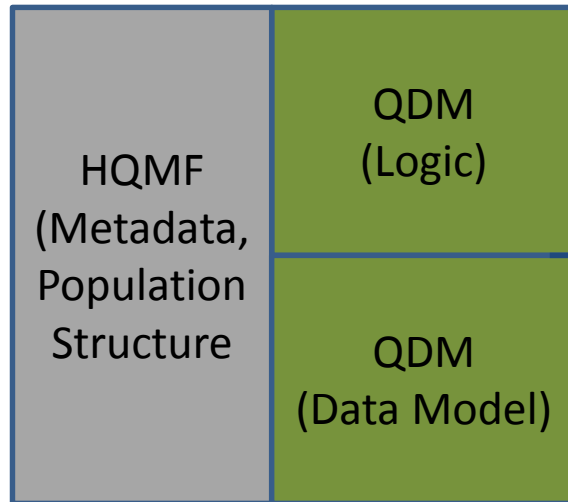
ESAC, Inc.

Agenda

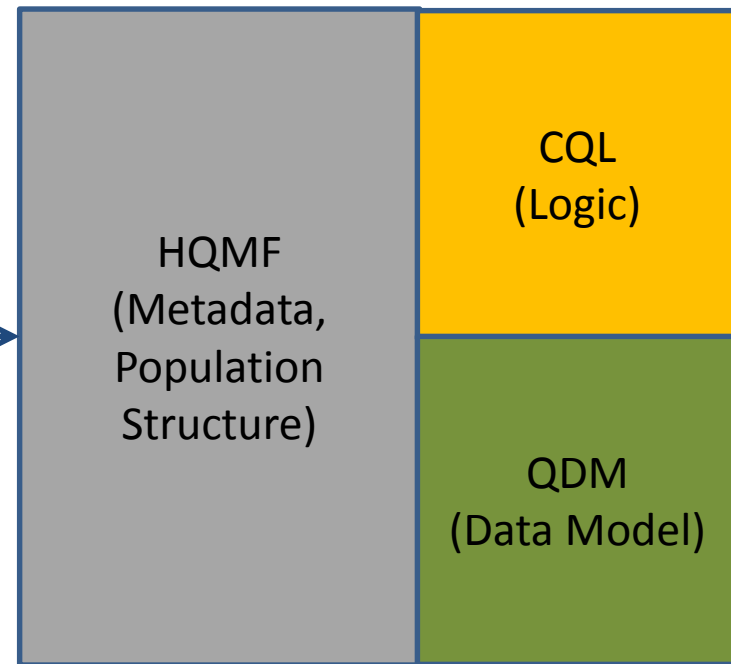
- Welcome and Background
- Implementation of CQL

Evolving eCQM Standards

Now



Near Term



Definitions:

eCQM – Electronic Clinical Quality Measure

HQMF – Health Quality Measure Format

CQL – Clinical Quality Language

QDM – Quality Data Model

Differences Between QDM Now and With CQL

QDM Now

- Data Model and Logic are both in the QDM

QDM with CQL

- The Data Model will continue to exist as the QDM
- CQL will provide the logic expressions and will replace that function currently in the QDM

Benefits of CQL

	QDM Logic	CQL Logic
Modularity and Computability	Low	High
Data Model Flexibility	**	High
Expressive and Robust Logic Expression	Low	High
Duplicative work for Implementers, Vendors, and Developers	Yes	Lower

Proposed Timeline For Updating Standards

Work Effort: 2016 through Fall 2017

Fall 2017 +

Measures using QDM v4.2 & HQMF 2.1

Testing CQL – QDM – HQMF 2.1

Measure Development

- 2015
- 2016

Testing and Development

- Measure Developers
- Implementers & Vendors
- CQL Training/Education
- Measure Authoring Tool
- Bonnie & Cypress
- Quality Data Model
- Integration Testing
- Feedback Loops

**Testing eCQM using CQL –
– QDM – HQMF 2.1**

**Measure Development and Testing
in a Simulated Environment**

- Starts 2017

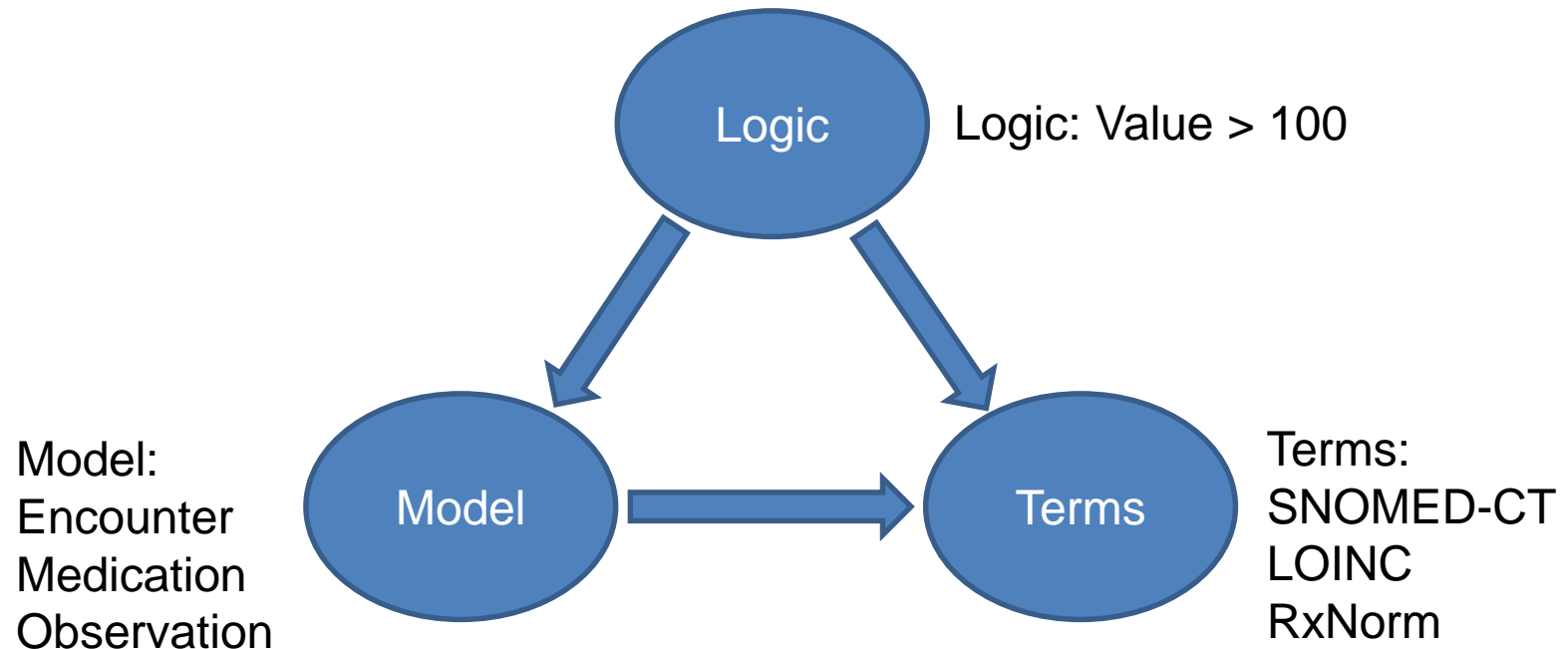
Presentation Goals

- Knowledge Sharing with CQL
- Language Runtime Semantics
- Clinical Data Representation in CQL
- Evaluation Approaches
- Overview of Existing Tooling

Assumptions

- Familiar with CQL
- Background in language processing
 - Language translation and/or evaluation
- Familiar with Clinical Data Representation
 - Clinical Data Models
 - Terminology

Components of Sharing Logic

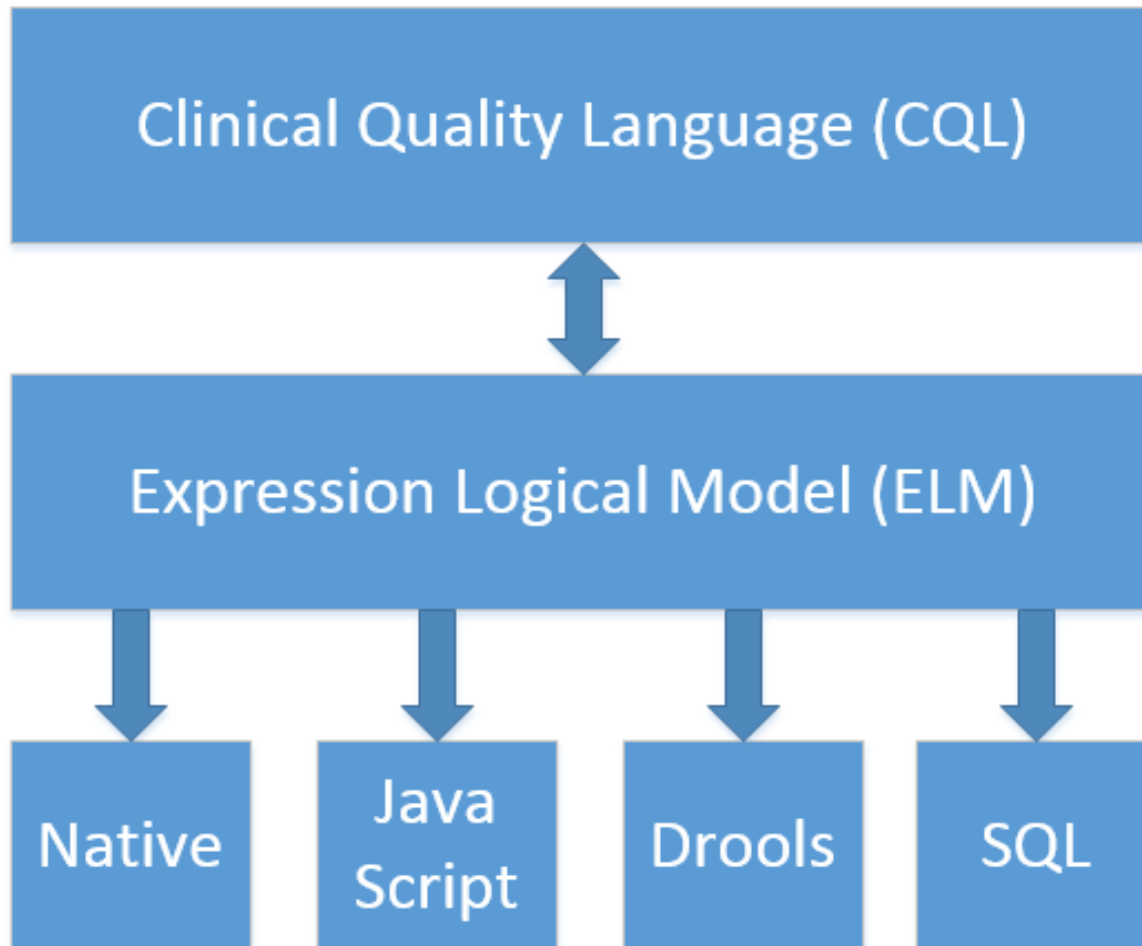


Definitions:

SNOMED CT – Systematized Nomenclature of Medicine – Clinical Terms

LOINC – Logical Observation Identifiers Names and Codes

CQL Architecture



Authors use CQL to produce libraries containing human-readable yet precise logic.

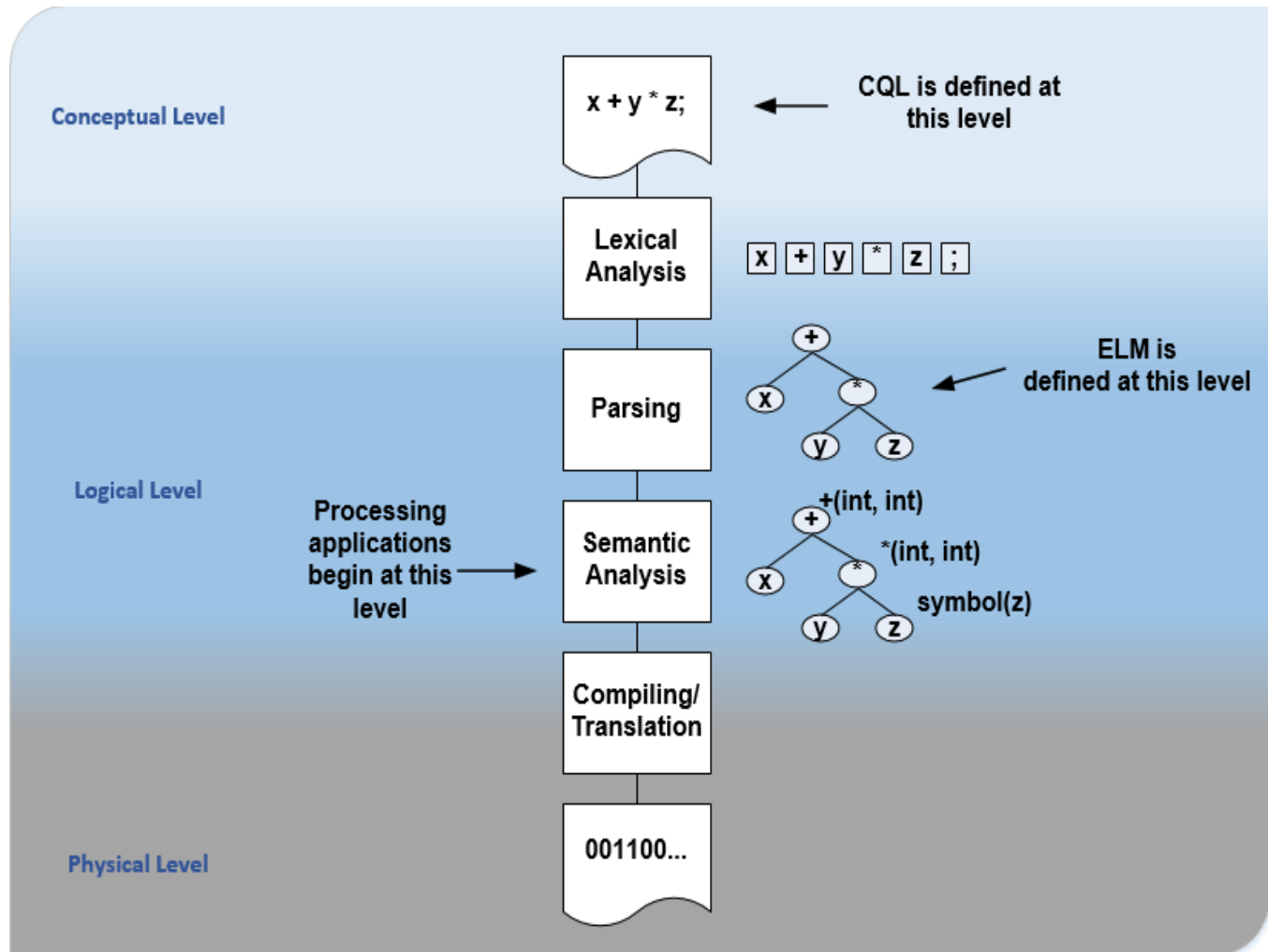
ELM XML documents contain machine-friendly rendering of the CQL logic. This is the intended mechanism for distribution of libraries.

Implementation environments will either directly execute the ELM, or perform translation from ELM to their target environment language.

Definitions:

SQL – Structured query language

CQL-to-ELM Translation

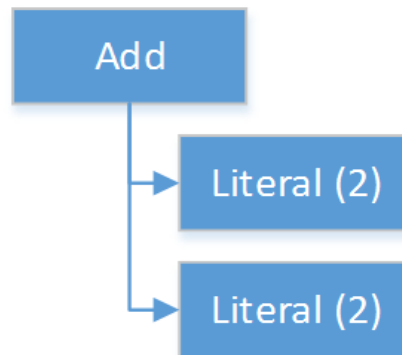


ELM

- A “byte-code” representation of CQL logic: carries sufficient semantics to enable execution independent of the CQL that produced it
- A “canonical” representation in terms of more primitive operations: focused on supporting implementation use cases such as evaluation and translation

ELM Representation

- ELM expressions are built as trees of nodes, where each kind of expression is represented by a different node type
- For example, $2 + 2$ is represented as:



ELM Representation (cont.)

- In general, operations and functions in CQL have an equivalent ELM representation

CQL Operator or Function	ELM Node Type
=	Equal
and	And
+	Add
Ceiling()	Ceiling

- Complete reference in the CQL specification

Type Categories

- Primitive types
 - Boolean
 - String
 - Integer
 - Decimal
 - DateTime
 - Time
- Collection types
 - List<T>
- Structured types
 - Class types (defined by a data model)
 - Tuple (anonymous class types)
- Interval types
 - Interval<T> (must be an ordered type)

Data Access

- All data access is done through Retrieve
 - Type information (data type and optional “template” identifier)
 - Code filter (a valueset or a set of codes)
 - Date filter (a date range)
 - Path information (id, code, date)

Simple Retrieve

- Pharyngitis Diagnoses:

```
[ "Diagnosis": "Acute Pharyngitis" ]
```

- ELM Retrieve:

```
<operand xsi:type="Retrieve"  
  dataType="qdm:Diagnosis"  
  templateId="Diagnosis"  
  codeProperty="code">  
  <codes name="Acute Pharyngitis" xsi:type="ValueSetRef"/>  
</operand>
```

Specifying Data Models

- Each data model is described with “model info”
- Describes the types available in the model
- Also defines “primary code path” for each retrievable type
- Specifies the “patient” type
- *NOTE: Model info is not required by ELM, it's only required to translate CQL to ELM*

Model Info Example

```
<ns4:TypeInfo xsi:type="ns4:ClassInfo"  
  name="QDM.Diagnosis"  
  identifier="Diagnosis"  
  label="Diagnosis"  
  retrievable="true"  
  primaryCodePath="code"  
  baseType="QDM.QDMBaseType">  
  <ns4:element name="onsetDatetime" type="System.DateTime"/>  
  <ns4:element name="abatementDatetime" type="System.DateTime"/>  
  <ns4:element name="anatomicalLocationSite" type="System.Concept"/>  
  <ns4:element name="severity" type="System.Concept"/>  
</ns4:TypeInfo>
```

System Model

- System.Any – Base type for all types
- System.Boolean
- System.Integer
- System.Decimal
- System.String
- System.DateTime
- System.Time
- System.Quantity – e.g., 3 'gm'
- System.Code – code, system, version, display
- System.Concept – codes, display

CQL Library

- Named, versioned groupings of CQL components

```
2
3 library CMS55 version '1'
4
5 using QDM
6
7 valueset "Inpatient": '2.16.840.1.113883.3.666.5.307'
8
9 parameter "Measurement Period" default Interval[@2014-01-01T00:00:00.0, @2015-01-01T00:00:00.0)
10
11 context Patient
12
13 define "Inpatient Encounters":
14     ["Encounter, Performed": "Inpatient"] E
15     where E.lengthOfStay <= 120 days
16     and E.dischargeDatetime during "Measurement Period"
17
18
```

Library Example

```

<library xmlns="urn:hl7-org:elm:r1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:t="urn:hl7-org:elm-types:r1"
  xmlns:qdm="urn:healthit-gov:qdm:v4_2">
  <identifier id="CMS55" version="1"/>
  <schemalocalIdentifier id="urn:hl7-org:elm" version="r1"/>
  <usings>
    .....
    <def localIdentifier="System" uri="urn:hl7-org:elm-types:r1"/>
    .....
    <def localIdentifier="QDM" uri="urn:healthit-gov:qdm:v4_2"/>
  </usings>
  <parameters>
    .....
    <def name="Measurement Period" accessLevel="Public">
  </parameters>
  <valueSets>
    .....
    <def name="Inpatient" id="2.16.840.1.113883.3.666.5.307" accessLevel="Public"/>
  </valueSets>
  <statements>
    .....
    <def name="Patient" context="Patient">
    .....
    <def name="Inpatient Encounters" context="Patient" accessLevel="Public">
  </statements>
</library>

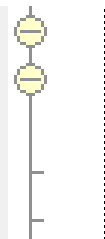
```

Parameter Definition

```
<def name="Measurement Period" accessLevel="Public">
  <default lowClosed="true" highClosed="false" xsi:type="Interval">
    <low xsi:type="DateTime">
      <year valueType="t:Integer" value="2014" xsi:type="Literal"/>
      <month valueType="t:Integer" value="1" xsi:type="Literal"/>
      <day valueType="t:Integer" value="1" xsi:type="Literal"/>
      <hour valueType="t:Integer" value="0" xsi:type="Literal"/>
      <minute valueType="t:Integer" value="0" xsi:type="Literal"/>
      <second valueType="t:Integer" value="0" xsi:type="Literal"/>
      <millisecond valueType="t:Integer" value="0" xsi:type="Literal"/>
    </low>
    <high xsi:type="DateTime">
      <year valueType="t:Integer" value="2015" xsi:type="Literal"/>
      <month valueType="t:Integer" value="1" xsi:type="Literal"/>
      <day valueType="t:Integer" value="1" xsi:type="Literal"/>
      <hour valueType="t:Integer" value="0" xsi:type="Literal"/>
      <minute valueType="t:Integer" value="0" xsi:type="Literal"/>
      <second valueType="t:Integer" value="0" xsi:type="Literal"/>
      <millisecond valueType="t:Integer" value="0" xsi:type="Literal"/>
    </high>
  </default>
</def>
```

Patient Context

```
10  
11 context Patient  
12
```



```
<def name="Patient" context="Patient">  
  <expression xsi:type="SingletonFrom">  
    <operand dataType="qdm:Patient" templateId="Patient" xsi:type="Retrieve"/>  
  </expression>  
</def>
```


Expression Example

```

12
13 define "Inpatient Encounters":
14     ["Encounter, Performed": "Inpatient"] E
15     where E.lengthOfStay <= 120 days
16     and E.dischargeDatetime during "Measurement Period"
17

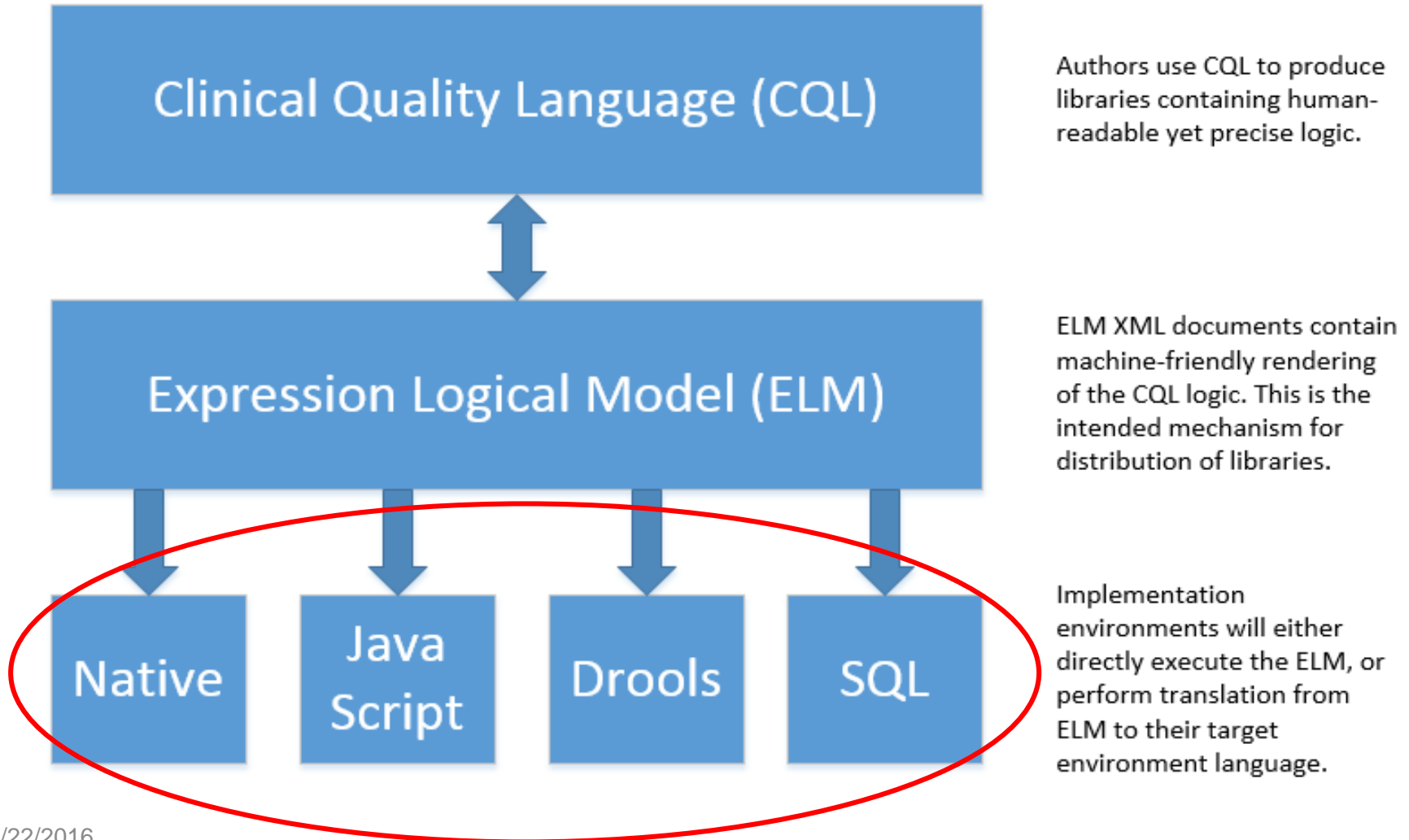
```

```

<def name="Inpatient Encounters" context="Patient" accessLevel="Public">
  <expression xsi:type="Query">
    <source alias="E">
      <expression dataType="qdm:EncounterPerformed" templateId="EncounterPerformed" codeProperty="code" xsi:type="Retrieve">
        <codes name="Inpatient" xsi:type="ValueSetRef"/>
      </expression>
    </source>
    <where xsi:type="And">
      <operand xsi:type="LessOrEqual">
        <operand path="lengthOfStay" scope="E" xsi:type="Property"/>
        <operand value="120" unit="days" xsi:type="Quantity"/>
      </operand>
      <operand xsi:type="In">
        <operand path="dischargeDatetime" scope="E" xsi:type="Property"/>
        <operand name="Measurement Period" xsi:type="ParameterRef"/>
      </operand>
    </where>
  </expression>
</def>

```

Evaluation Approaches



Clinical Quality Language (CQL)

Authors use CQL to produce libraries containing human-readable yet precise logic.

Expression Logical Model (ELM)

ELM XML documents contain machine-friendly rendering of the CQL logic. This is the intended mechanism for distribution of libraries.

Native

Java
Script

Drools

SQL

Implementation environments will either directly execute the ELM, or perform translation from ELM to their target environment language.

Evaluation Approaches

- Native Evaluation
 - Each node in the ELM is an *evaluator*
 - Provides a simple basis for an execution engine
- Interpreter
 - A simple *visitor* pattern can provide an interpreter
- Translation
 - ELM provides a simple and computable description of the logic, suitable for translation to other targets (e.g., Drools, SQL)

CQL-to-ELM Translator

- CQL-to-ELM Translator
 - Reference implementation of a translator that produces ELM from CQL input
 - Kept up to date as part of the specification
 - Used to produce and validate examples used in the specification
- Java-based
- Service packaging available

JavaScript Engine

- JavaScript ELM interpreter
 - Runs based on the JSON of an ELM library
 - Can be embedded in a browser or run via node.js
 - Kept up to date as part of the tooling for the specification

HeD Schema Framework

- .NET Based Framework for building ELM language processing applications
 - Part of the CDS Knowledge Artifact Specification (HeD) tooling
 - Used to validate CDS KAS examples
 - Also to translate HeD for pilots

CQL Resources

- HL7 Standard: Clinical Quality Language Specification, Release 1 DSTU
 - http://www.hl7.org/implement/standards/product_brief.cfm?product_id=400
- HL7 CDS Workgroup Project Homepage:
 - http://wiki.hl7.org/index.php?title=Clinical_Quality_Language
- GitHub Tools Repository:
 - https://github.com/cqframework/clinical_quality_language

Questions?

eCQI Resource Center

- CQL Space
 - <https://ecqi.healthit.gov/cql>

The one-stop shop for the most current resources to support **Electronic Clinical Quality Improvement.**

Learn about eCQI resources and connect with the community of professionals who are dedicated to clinical quality improvement for better health

Getting Started



A *gentle* introduction to understanding eCQI and this Resource Center

[More information](#)


eCQMs



The who, what, when, where, and why of eCQMs

[More information](#)

Education



A selection of educational materials and resources to broaden your eCQI knowledge

[More information](#)

Latest News

- Tue 03 May **NLM released VSAC update version 2.10.11 on April 20, 2016**
UPDATED: Code System Versions
 RxNorm 2016-02, 2016-03, 2016-04
 US Edition of SNOMED CT 2016-03
 See all VSAC-hosted code system versions in the VSAC Support Center. Select the Help button on any VSAC page and go to Code Systems and Tools
NEW: VSAC Authoring and VSAC Collaboration Support for CMS eCQM Value Set Annual Update
 VSAC Authoring: The Centers for Medicare and Medicaid Services (CMS) Clinical Quality Measure (eCQM) value sets are... Read more
- Tue 03 May **Soliciting Example Electronic Clinical Quality Measures for Upcoming Cooking with CQL Webinar Sessions**
 CMS and ESAC, Inc. are looking for examples of electronic clinical quality measure

Upcoming Events

- May **18**
QDM User Group Webinar
 NOTE: Participants are not required to register for this meeting.
 JOIN WEBEX MEETING
<https://esacinc2.webex.com/esacinc2/j.php?MTID=m9a94b76ea1eb76fd1ad9c3d66eb3b60>
 Meeting number: 733 101 720
 Meeting password: qdm1
- JOIN BY PHONE
 +1-415-655-0002 US Toll

CQL | eCQI Resource Center

← → ↻ <https://ecqi.ahrqdev.org/cql>

For quick access, place your bookmarks here on the bookmarks bar. [Import bookmarks now...](#)

eCQI Resource Center
The one-stop shop for the most current resources to support electronic clinical quality improvement.

CMS The Office of the National Coordinator for Health Information Technology

About FAQ Glossary of eCQI Terms eCQI Resource Center Contact Information

Search [] fi [] Login

Topic areas EH Measures EP Measures QDM HQMF QRDA eCQM Tools Kaizen Education

CQL

Clinical Quality Language (CQL) is an HL7 draft standard for trial use (DSTU). It is part of the effort to harmonize standards between electronic clinical quality measures (eQMs) and clinical decision support (CDS). CQL provides the ability to express logic that is human readable yet structured enough for processing a query electronically. In the future, CQL is to be used in all of the clinical quality measure HQMF electronic specifications. It will replace the logic expressions currently defined in the Quality Data Model (QDM) and QDM (v5.0) will include only the method for defining the data elements (the data model). More information about CQL is found at:

- HL7 Standard: Clinical Quality Language Specification, Release 1 DSTU
- HL7 CDS Workgroup Project Homepage
- GitHub Tools Repository

CQL is discussed in the HL7 CQF-on-FHIR forum and CQL STU comments are discussed during the HL7 Clinical Decision Support Work Group calls.

CQL Formatting and Usage Wiki

This wiki serves as a collaborative workspace for the development of CQL formatting conventions and usage patterns for the representation of logic within quality measures. All users have edit rights to be able to submit edits, add comments and concerns. Items on the Wiki are a work in progress and subject to change.

<https://github.com/esacinc/CQL-Formatting-and-Usage-Wiki/wiki>

Comments or Questions?

For issues, comments, and questions related to CQL, please use the CQL JIRA Issue Tracker.

<https://jira.oncprojectracking.org/browse/CQLIT>

CQL Events

For upcoming CQL Events, click the CQL Events link on the right navigation bar.

CQL Resources

For past CQL presentations, click the CQL Educational Resources link on the right navigation bar.

Public

[Request space membership](#)

There is no content in this space.