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

HQMF (Metadata, Population Structure)
QDM (Data Model)
QDM (Logic)

Near Term

HQMF (Metadata, Population Structure)
CQL (Logic)
QDM (Data Model)

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

<table>
<thead>
<tr>
<th>QDM Now</th>
<th>QDM with CQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Data Model and Logic</td>
<td>• The Data Model will continue to exist as the QDM</td>
</tr>
<tr>
<td>are both in the QDM</td>
<td>• CQL will provide the logic expressions and will replace that function currently in the QDM</td>
</tr>
</tbody>
</table>
## Benefits of CQL

<table>
<thead>
<tr>
<th>Feature</th>
<th>QDM Logic</th>
<th>CQL Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modularity and Computability</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Data Model Flexibility</td>
<td><strong>None</strong></td>
<td>High</td>
</tr>
<tr>
<td>Expressive and Robust Logic Expression</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Duplicative work for Implementers, Vendors, and Developers</td>
<td>Yes</td>
<td>Lower</td>
</tr>
</tbody>
</table>
Proposed Timeline For Updating Standards

Work Effort: 2016 through Fall 2017

Measures using QDM v4.2 & HQMF 2.1

Testing CQL – QDM – HQMF 2.1

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

Fall 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

Logic: Value > 100

Model: Encounter, Medication, Observation

Terms: SNOMED-CT, LOINC, RxNorm

Definitions:
SNOMED CT – Systematized Nomenclature of Medicine – Clinical Terms
LOINC – Logical Observation Identifiers Names and Codes
CQL Architecture

Clinical Quality Language (CQL)

Expression Logical Model (ELM)

Native
Java Script
Drools
SQL

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 Training for Measure Implementers

CQL-to-ELM Translation

1. Conceptual Level
   - CQL is defined at this level
   - `x + y * z;`

2. Logical Level
   - Processing applications begin at this level
   - Parsing
     - `x + y * z;`
   - Semantic Analysis
     - `+(int, int)`
     - `*(int, int)`
     - `symbol(z)`

3. Physical Level
   - Compiling/Translation
     - `001100...`
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:

```
+---+---+
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```text
Add
```

```
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Literal (2)
```
In general, operations and functions in CQL have an equivalent ELM representation.

<table>
<thead>
<tr>
<th>CQL Operator or Function</th>
<th>ELM Node Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal</td>
</tr>
<tr>
<td>and</td>
<td>And</td>
</tr>
<tr>
<td>+</td>
<td>Add</td>
</tr>
<tr>
<td>Ceiling()</td>
<td>Ceiling</td>
</tr>
</tbody>
</table>

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:
  
  ```json
  "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

```xml
<ns4:typelnfo 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:typelnfo>
```
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

```cql
library CMS55 version '1'
using QDM
valueset "Inpatient": '2.16.840.1.113883.3.666.5.307'
parameter "Measurement Period" default Interval[@2014-01-01T00:00:00.0, @2015-01-01T00:00:00.0]
context Patient
define "Inpatient Encounters":
    ["Encounter, Performed": "Inpatient"] E
    where E.lengthOfStay <= 120 days
    and E.dischargeDatetime during "Measurement Period"
```
Library Example

```xml
<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"/>
   <schemalldentifier 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

```xml
<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>
  </default>
  <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>
</def>
```
Patient Context

```xml
<def name="Patient" context="Patient">
  <expression xsi:type="SingletonFrom">
    <operand dataType="qdm:Patient" templateld="Patient" xsi:type="Retrieve"/>
  </expression>
</def>
```
Expression Example

define "Inpatient Encounters":
    ["Encounter, Performed": "Inpatient"] E
    where E.lengthOfStay <= 120 days
    and E.dischargeDatetime during "Measurement Period"

<def name="Inpatient Encounters" context="Patient" accessLevel="Public">
    <expression xsi:type="Query">
        <source alias="E">
            <expression dataType="qdm:EncounterPerformed" templateld="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)

Expression Logical Model (ELM)

Native | Java Script | Drools | SQL

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.
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

- HL7 CDS Workgroup Project Homepage:

- GitHub Tools Repository:
  - [https://github.com/cqframework/clinical_quality_language](https://github.com/cqframework/clinical_quality_language)
Questions?
eCQI Resource Center

• CQL Space
  • https://ecqi.healthit.gov/cql
CQL Training for Measure Implementers

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 02 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 eCQV Value Set Annual Update
  VSAC Authoring: The Centers for Medicare and Medicaid Services (CMS) Clinical Quality Measure (eCQV) 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, 2016
- 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=m8a94b76e1eb76fbd1a09c8d6eb3b60
  Meeting number: 733 101 720
  Meeting password: qdm1
  JOIN BY PHONE
  +1-415-655-0002 US Toll
CQL Training for Measure Implementers

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 (eCQM) 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.


Comments or Questions?

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

https://jira.oncprojecttracking.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.