Clinical Quality Language (CQL) 301: Training for Measure Implementers

Shanna Hartman
Centers for Medicare & Medicaid Services

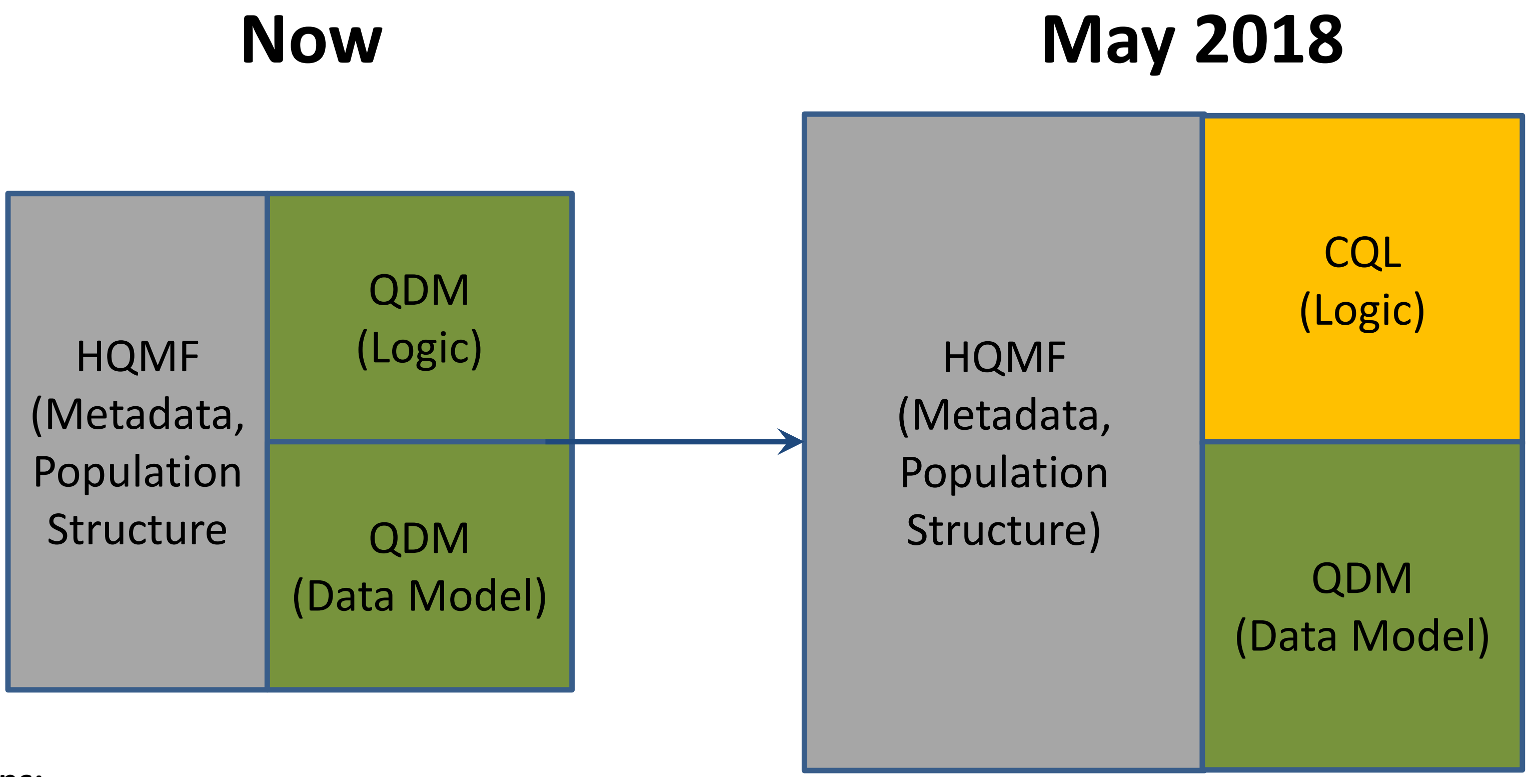
Deborah Krauss
Centers for Medicare & Medicaid Services (CMS)

Bryn Rhodes
ESAC, Inc.

CQL Training for Measure Implementers: Agenda

- Welcome and Background
- Implementation of CQL

CQL Training for Measure Implementers: Evolving eCQM Standards


Now
May 2018
HQMF (Metadata, Population Structure
QDM (Logic)
QDM (Data Model)
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

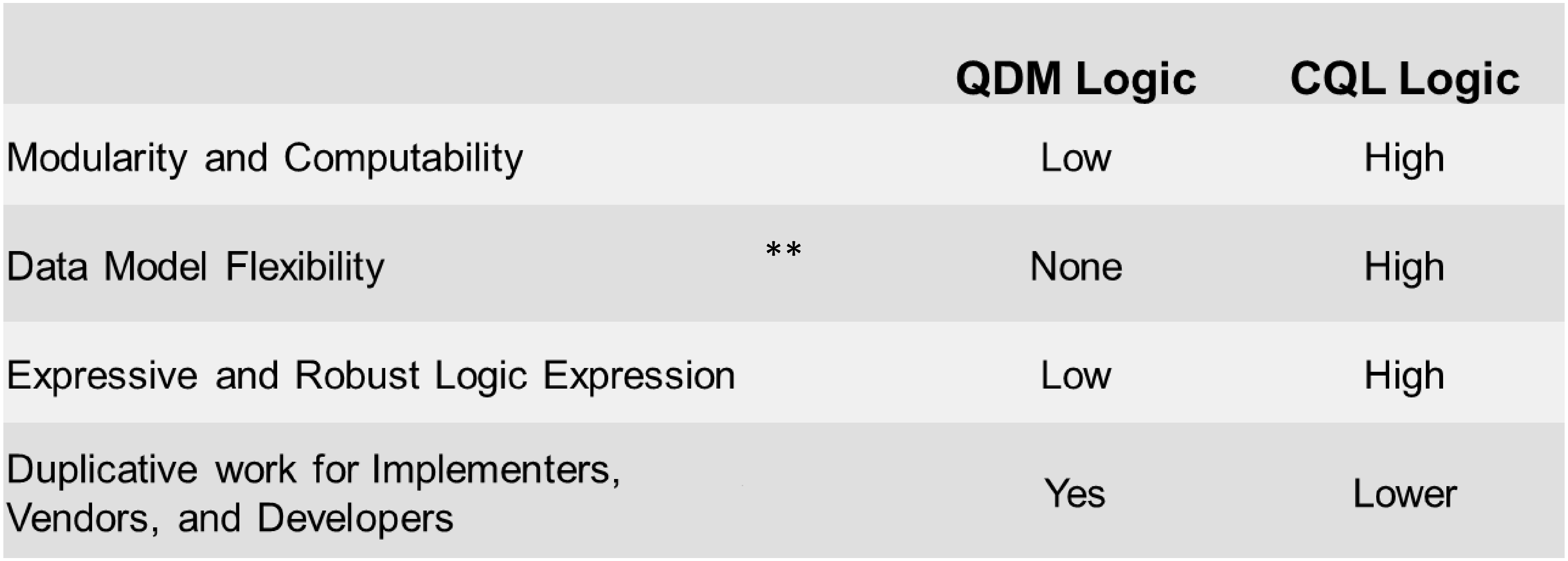
Updated 12/8/2017

CQL Training for Measure Implementers: Differences Between QDM Now and With CQL

QDM Now	QDM with CQL
• Data Model and Logic are both in the QDM	• 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

Updated 12/8/2017

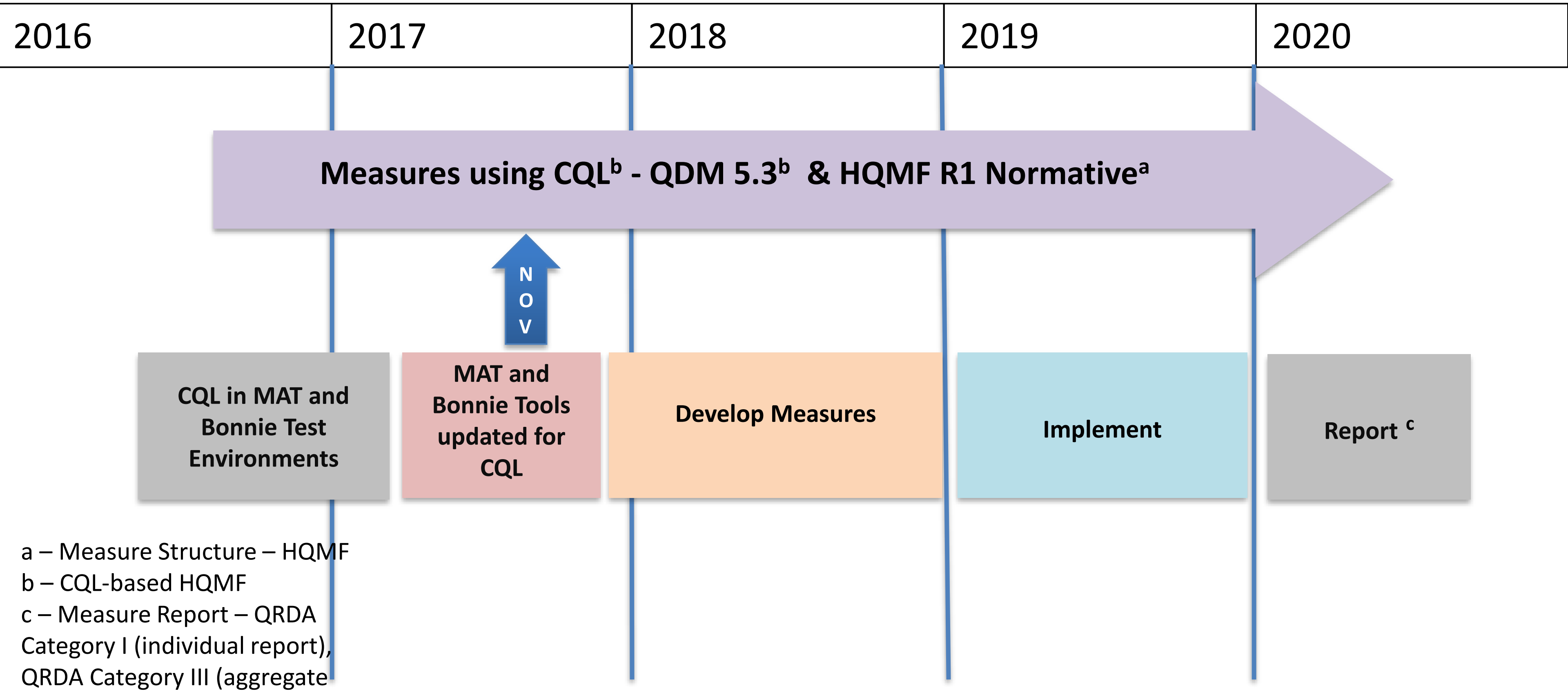
CQL Training for Measure Implementers: Benefits of CQL



	QDM Logic	CQL Logic
Modularity and Computability	Low	High
Data Model Flexibility **	None	High
Expressive and Robust Logic Expression	Low	High
Duplicative work for Implementers, Vendors, and Developers	Yes	Lower

CQL Training for Measure Implementers: Proposed Timeline For Updating Standards

Measure Development – Expected Timelines


2016
2017
2018
2019
2020
Measures using CQL[b] - QDM 5.3[b] & HQMF R1 Normative[a]
N O V
CQL in MAT and Bonnie Test Environments
MAT and Bonnie Tools updated for CQL
Develop Measures
Implement
Report [c]

a – Measure Structure – HQMF
b – CQL-based HQMF
c – Measure Report – QRDA Category I (individual report), QRDA Category III (aggregate report)
Updated 12/8/2017

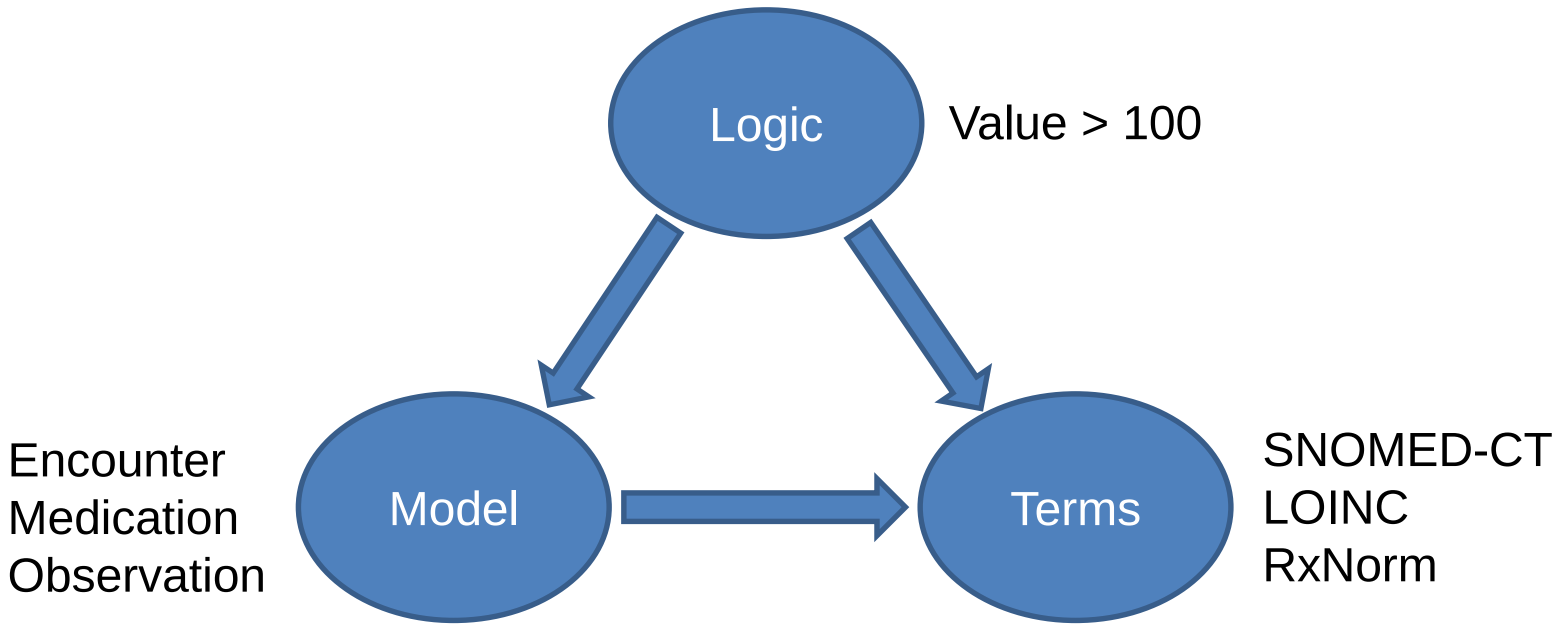
CQL Training for Measure Implementers: Presentation Goals

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

CQL Training for Measure Implementers: Assumptions

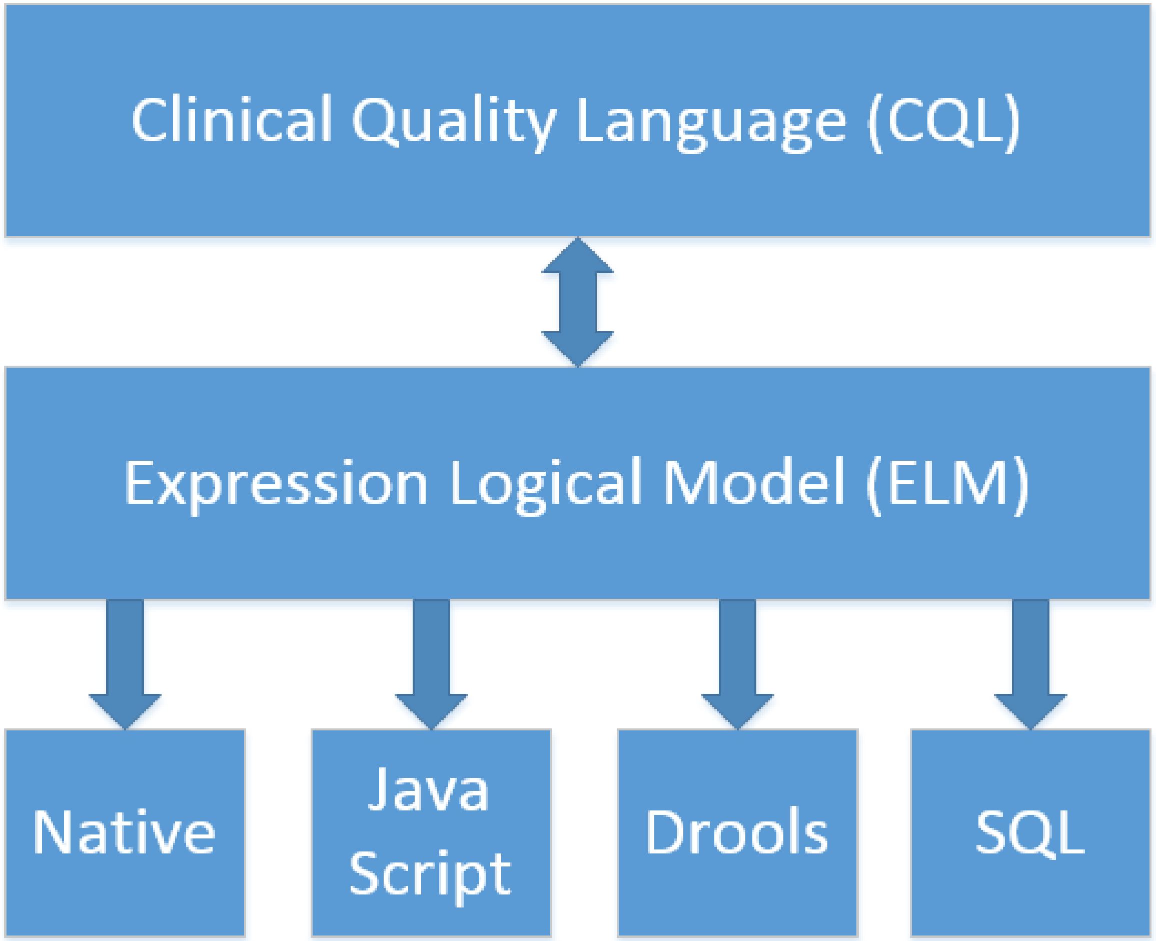
- Familiar with CQL
- Background in language processing
  - Language translation and/or evaluation
- Familiar with Clinical Data Representation
  - Clinical Data Models
  - Terminology

CQL Training for Measure Implementers: Components of Sharing Logic



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

CQL Training for Measure Implementers: 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.

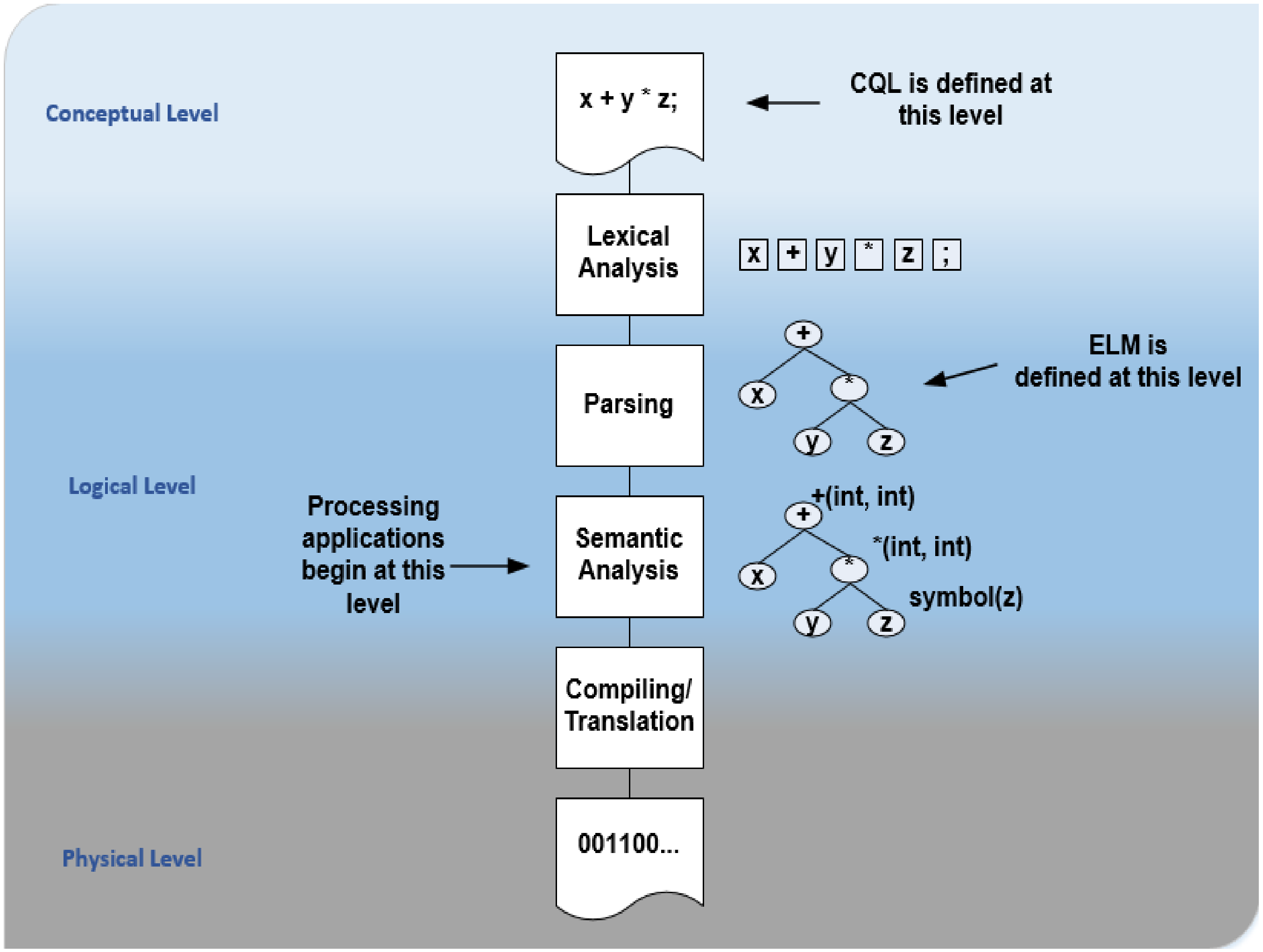
ELM XML documents contain machine-friendly rendering of the CQL logic. This is the intended mechanism for distribution of libraries.

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

Updated 12/8/2017

Definitions:
SQL – Structured query language

CQL Training for Measure Implementers: CQL-to-ELM Translation



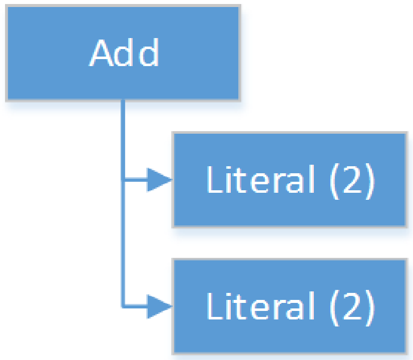
Updated 12/8/2017

CQL Training for Measure Implementers: 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

CQL Training for Measure Implementers: 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:



CQL Training for Measure Implementers: ELM Representation (cont'd)

- 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

CQL Training for Measure Implementers: Type Categories

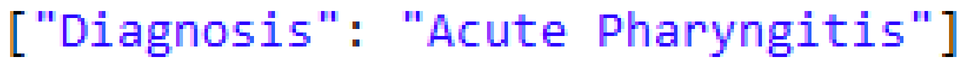
- Primitive types
  - Boolean
  - String
  - Integer
  - Decimal
  - DateTime
  - Time
- Collection types
  - List<T>
- Structured types
  - Class types (defined by a data model)
  - Tuple (anonymous class types)
- Interval types
  - Interval<T> (must be an ordered type)

CQL Training for Measure Implementers: 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)

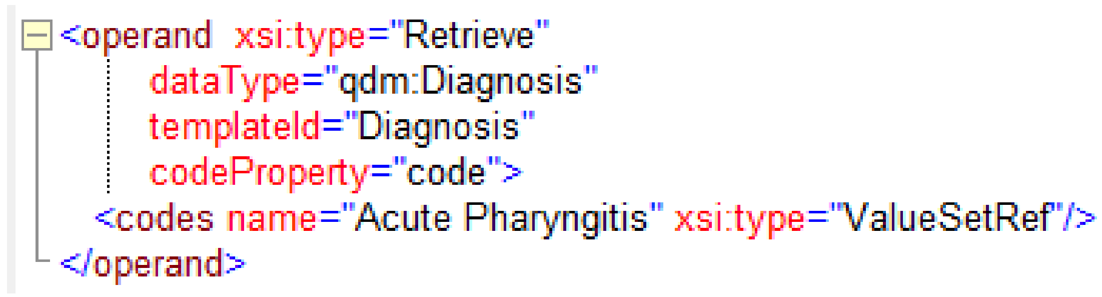
CQL Training for Measure Implementers: Simple Retrieve

- Pharyngitis Diagnoses:



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

- ELM Retrieve:

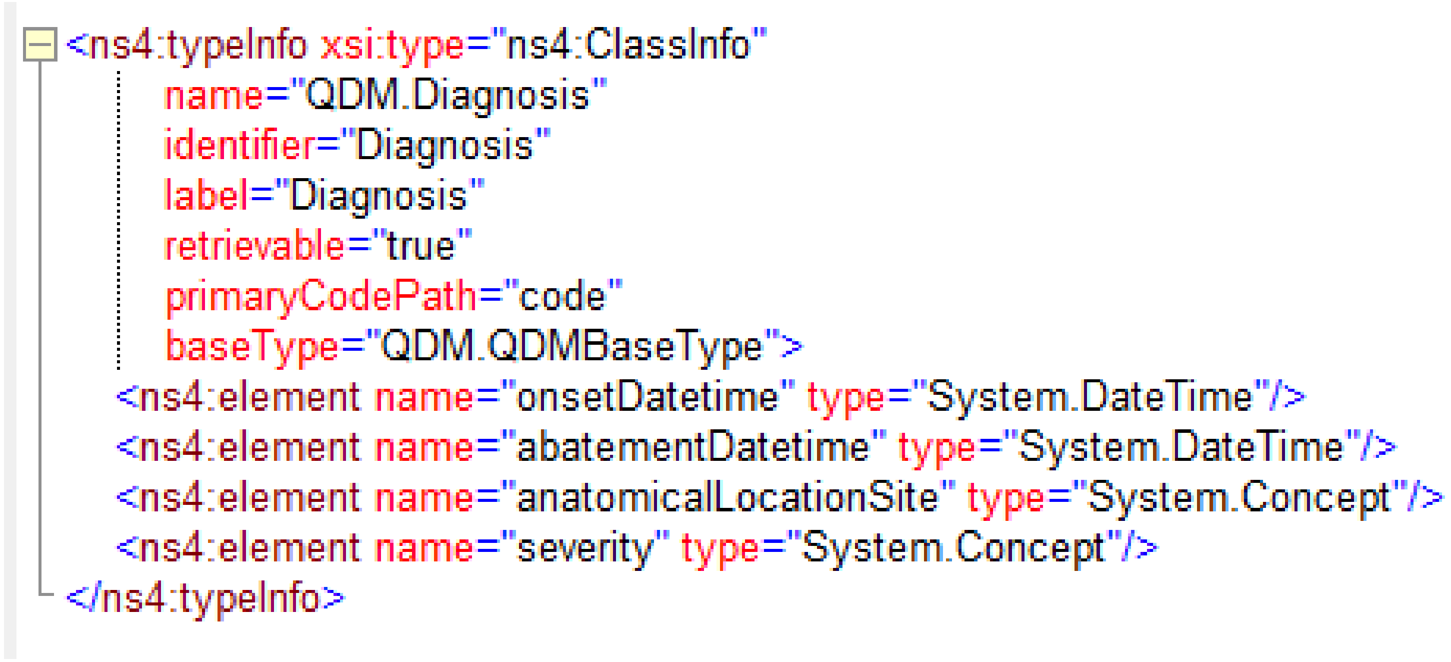


```
<operand  xsi:type="Retrieve"
        dataType="qdm:Diagnosis"
        templateId="Diagnosis"
        codeProperty="code">
    <codes name="Acute Pharyngitis" xsi:type="ValueSetRef"/>
</operand>
```

CQL Training for Measure Implementers: 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*

CQL Training for Measure Implementers: Model Info Example



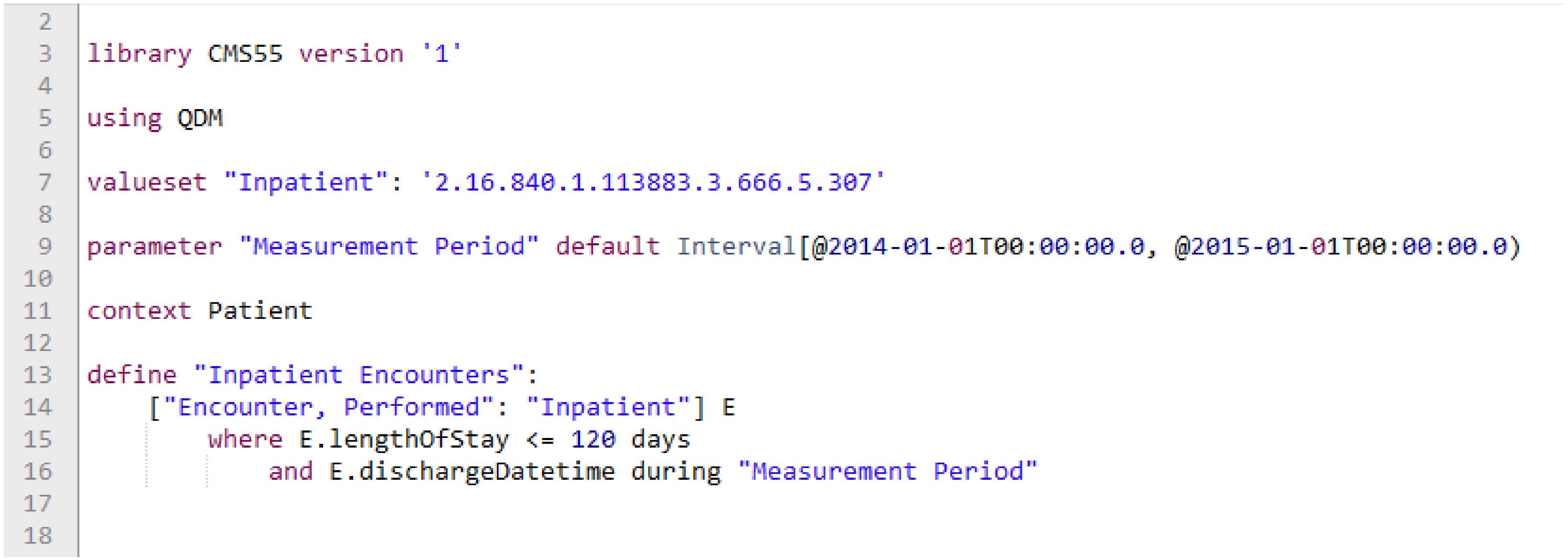
```
<ns4:typeInfo xsi:type="ns4:ClassInfo"
        name="QDM.Diagnosis"
        identifier="Diagnosis"
        label="Diagnosis"
        retrievable="true"
        primaryCodePath="code"
        baseType="QDM.QDMBaseType">
    <ns4:element name="onsetDatetime" type="System.DateTime"/>
    <ns4:element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

CQL Training for Measure Implementers: 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

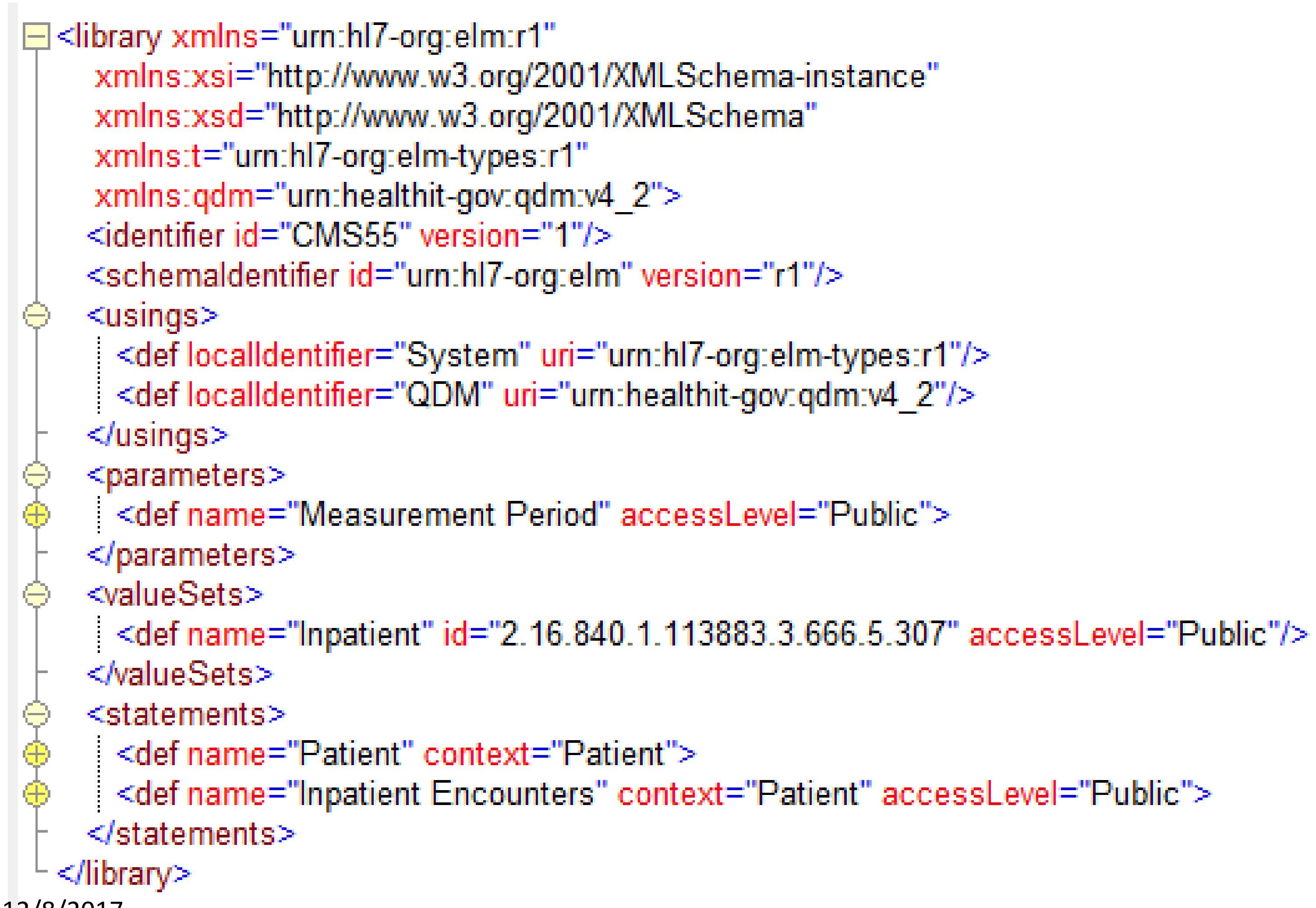
CQL Training for Measure Implementers: CQL Library

- Named, versioned groupings of CQL components



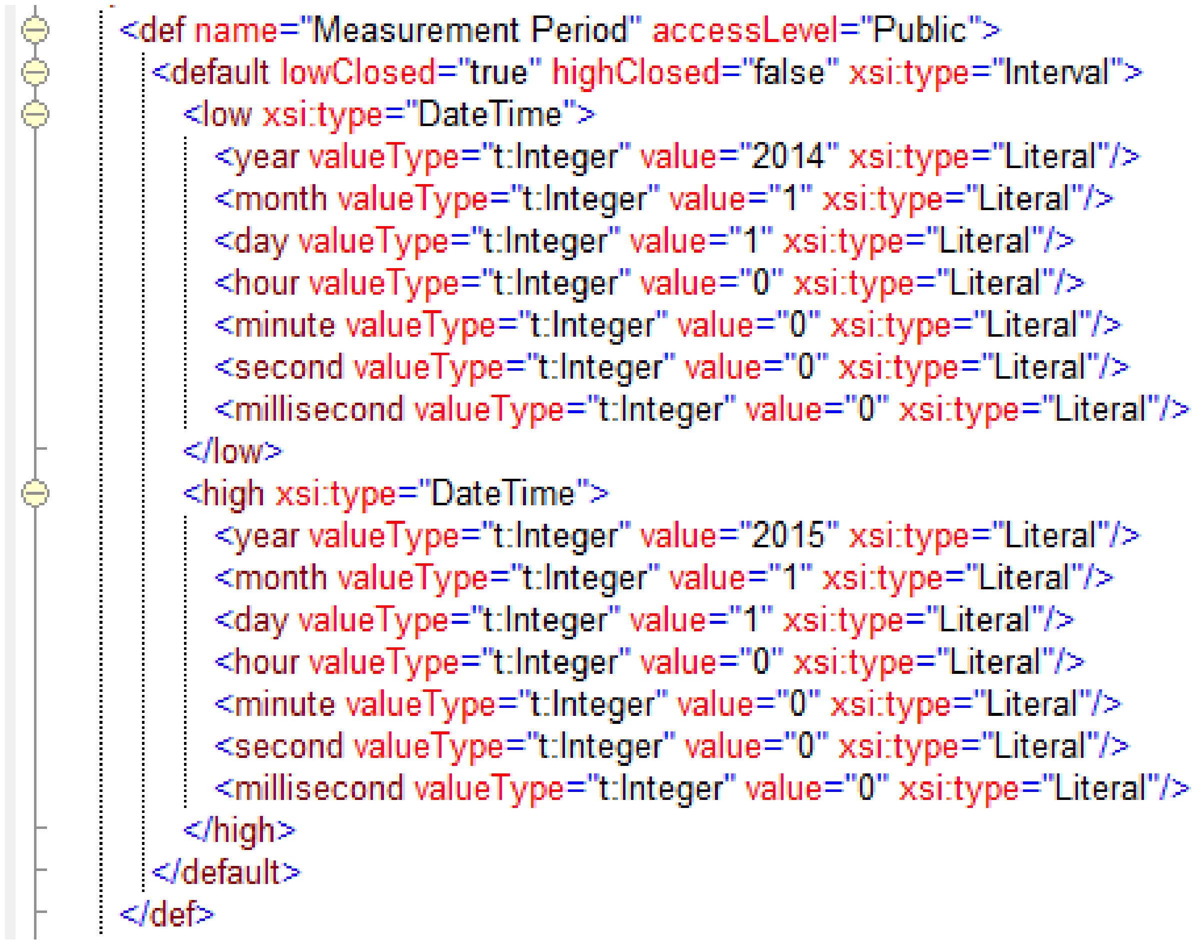
```

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"


```

CQL Training for Measure Implementers: Library Example



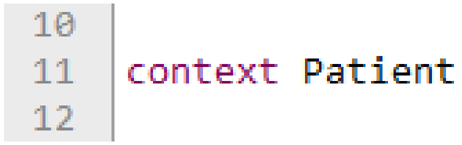
```
<library xmlns="urn:hl7-org:elm:r1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:t="urn:hl7-org:elm-types:r1"
    xmlns:qdm="urn:healthit-gov:qdm:v4_2">
   <identifier id="CMS55" version="1"/>
   <schemaIdentifier id="urn:hl7-org:elm" 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>
```

CQL Training for Measure Implementers: Parameter Definition

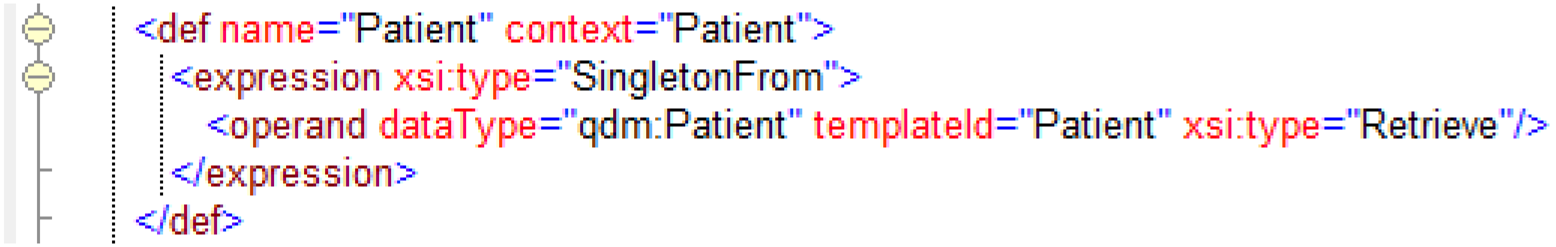


```
<def name="Measurement Period" accessLevel="Public">
  <default lowClosed="true" highClosed="false" xsi:type="Interval">
    <low xsi:type="DateTime">
      <year valueType="t:Integer" value="2014" xsi:type="Literal"/>
      <month valueType="t:Integer" value="1" xsi:type="Literal"/>
      <day valueType="t:Integer"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

CQL Training for Measure Implementers: Patient Context

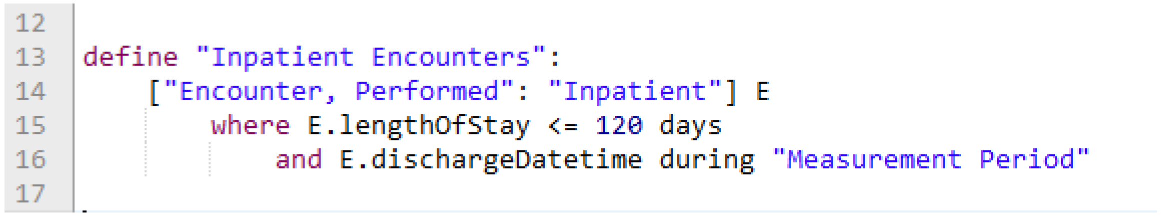


```
context Patient
```

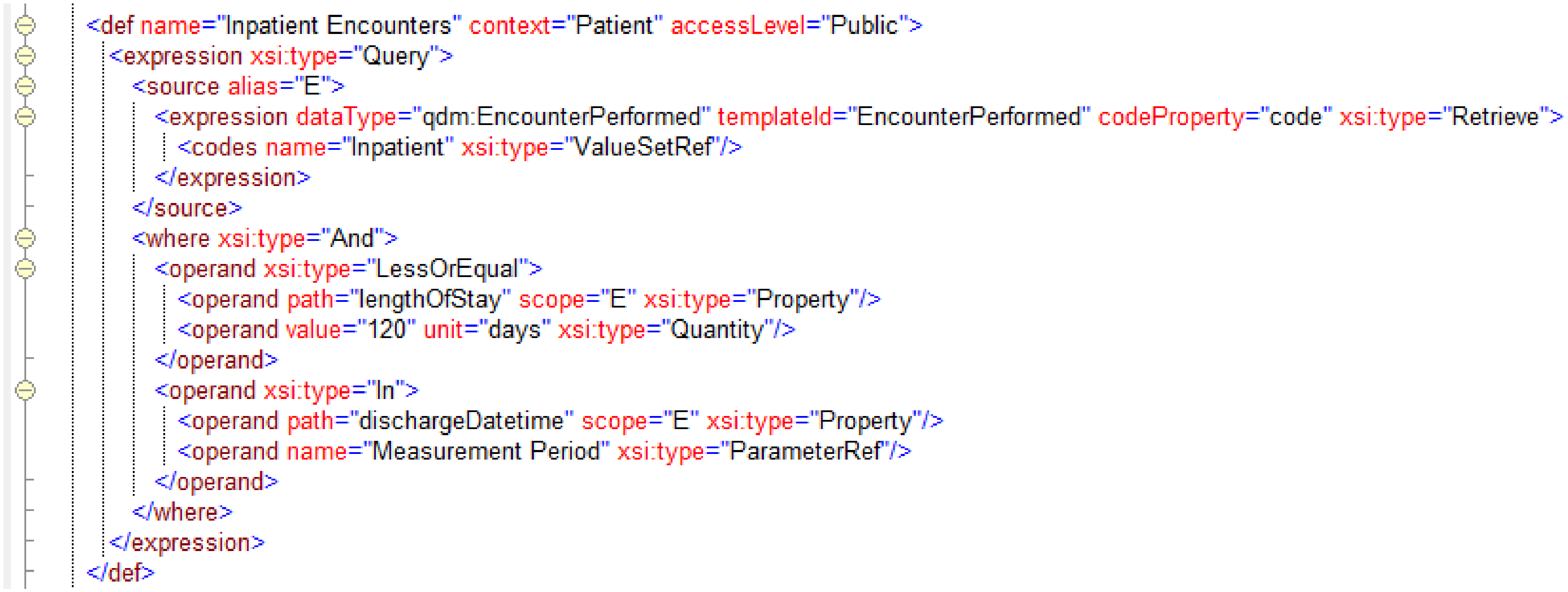


```
<def name="Patient" context="Patient">
  <expression xsi:type="SingletonFrom">
    <operand dataType="qdm:Patient" templateId="Patient" xsi:type="Retrieve"/>
  </expression>
</def>
```

CQL Training for Measure Implementers: Expression Example

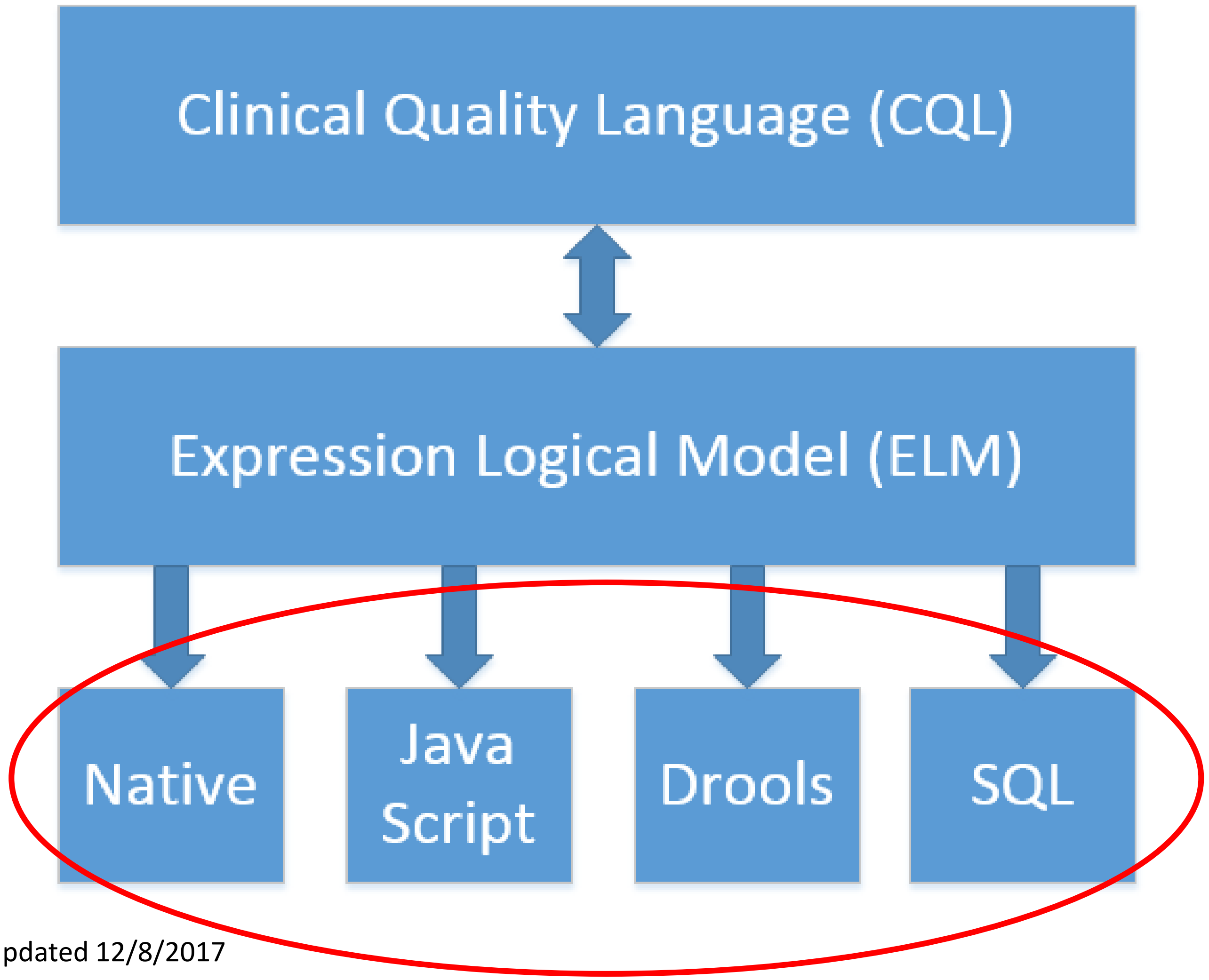


```
define "Inpatient Encounters":
    ["Encounter, Performed": "Inpatient"] E
        where E.lengthOfStay <= 120 days
            and E.dischargeDatetime during "Measurement Period"
```



```xml
<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>
```

CQL Training for Measure Implementers: 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.

Updated 12/8/2017

CQL Training for Measure Implementers: 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 Training for Measure Implementers: 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

CQL Training for Measure Implementers: 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

CQL Training for Measure Implementers: HeD Schema Framework

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

CQL Training for Measure Implementers: Resources

- CQL Specification - CQL Release 1, Standard for Trial Use (STU) 2
  - http://www.hl7.org/implement/standards/product_brief.cfm?product_id=400
- eCQI Resource Center
  - CQL Space, including the QDM v5.3 and v5.3 Annotated
    - https://ecqi.healthit.gov/cql
  - Check the eCQI Resource Center Events page and CQL Educational Resources page for more information
    - https://ecqi.healthit.gov/ecqi/ecqi-events
    - https://ecqi.healthit.gov/cql/cql-educational-resources

CQL Training for Measure Implementers: Resources (cont'd)

- CQL Formatting and Usage Wiki
  - https://github.com/esacinc/CQL-Formatting-and-Usage-Wiki/wiki
- CQL GitHub Tools Repository
  - https://github.com/cqframework/clinical_quality_language
- Measure Authoring Tool
  - https://www.emeasuretool.cms.gov/
- Bonnie Testing Tool
  - https://bonnie.healthit.gov/
- To submit an issues ticket for CQL, please visit the ONC JIRA site
  - https://oncprojectracking.healthit.gov/support/projects/CQLIT