The Centers for Medicare & Medicaid Services (CMS) contracted with Mathematica and its partners, including The Joint Commission, to develop this style guide to support the electronic specification and maintenance of electronic clinical quality measures that eligible professionals, eligible clinicians, eligible hospitals, and critical access hospitals can potentially use for reporting using certified electronic health records (EHRs) for CMS quality programs. Mathematica and its partners developed this document under two Measure and Instrument Development and Support indefinite delivery, indefinite quantity master contract vehicle task order contracts: Electronic Clinical Quality Measures Development and Maintenance for Eligible Clinicians (CMS Contract #75FCMC18D0032, Task Order #75FCMC19F0004) and Behavioral Health Measures Development and Inpatient and Outpatient Measure Maintenance (CMS Contract #75FCMC18D0032, Task Order #75FCMC19F0003).
## CONTENTS

1. PREREQUISITE .............................................................................................................................. 1
2. INTRODUCTION .............................................................................................................................. 2
3. STANDARDS – LIBRARIES ............................................................................................................ 3
   A. Best practices for using CQL libraries ....................................................................................... 4
   B. Best practices for naming CQL libraries .................................................................................... 4
4. STANDARDS – DEFINITIONS ........................................................................................................ 5
   A. Best practices for writing CQL definitions .................................................................................. 5
   B. Best practices for naming CQL definitions ................................................................................ 5
   C. Standards for naming definitions across measures ................................................................. 8
5. STANDARDS – ALIASES OR ARGUMENT NAMES ...................................................................... 9
   A. Best practices for using CQL aliases and argument names ..................................................... 9
   B. Standards for naming aliases across measures ..................................................................... 12
6. FUNCTIONS .................................................................................................................................. 14
   A. Best practices and standards for naming new CQL functions ................................................ 14
   B. Selecting functions .................................................................................................................. 14
7. OTHER CQL BEST PRACTICES .................................................................................................. 18
   A. Population criteria .................................................................................................................... 18
   B. Additional timing phrases ........................................................................................................ 18
   C. Operator precedence ............................................................................................................... 19
   D. Direct Reference Codes .......................................................................................................... 20

Version history ............................................................................................................................................ 21

APPENDIX A: THE ORDER OF OPERATOR PRECEDENCE IN CQL FROM HIGHEST TO LOWEST .................. A-1

APPENDIX B: CASE TYPE DEFINITIONS ............................................................................................... B-1
1. PREREQUISITE

Clinical Quality Language (CQL)\(^1\) is an HL7 standard developed as part of the Clinical Quality Framework (CQF) initiative. CQL is intended to promote standardization and harmonization across the CQF standards; it is also intended to be clinically focused, author friendly, and human readable.\(^2\)

Users of this style guide are assumed to have a baseline knowledge of CQL. Please refer to the following links for more information on CQL:

- eCQI Resource Center
- CQL Formatting and Usage Wiki
- Benefits of CQL
- CQL for Implementers

The Measure and Instrument Development and Support (MIDS) project, in coordination with the Centers for Medicare & Medicaid Services (CMS), created this document using guidelines from the CQL Formatting and Usage Wiki. The purpose of the guide is to promote consistency and reusability of the specifications in measures included in federal reporting programs. Measure developers and stakeholders contributed to the development of the standards in this guide.

The Measure Authoring Tool (MAT) supports the authoring of CQL and sharing of CQL libraries. Please see the MAT User Guide for additional MAT-specific information.

\(^1\) The guidance in this document is based on CQL STU 1.4, located at: https://cql.hl7.org/01-introduction.html.

\(^2\) Raw CQL files are human readable, but there is also an HTML version in the eCQM package exported from the MAT. HTML human readable provides a view consistent with the style provided with earlier electronic clinical quality measures based on Quality Data Model logic.
2. INTRODUCTION

The twofold purpose of this style guide is to (1) standardize expression-of-measure concepts across electronic clinical quality measures (eCQMs) and (2) define a uniform “look and feel” for eCQM logic using CQL. The guide focuses on common best practices that have been implemented across CQL-based eCQMs in CMS reporting programs. It also promotes the use of consistent language within the framework of CQL, including libraries, aliases, definitions, and functions, and provides guidance on other conventions, such as operator precedence. Measure stewards or developers who are developing or specifying eCQMs for potential inclusion in CMS reporting programs should follow these best practices.
3. STANDARDS: LIBRARIES

Libraries, the basic units of sharing CQL, consist of a foundation of CQL statements used within an eCQM. Every eCQM has at least one CQL library. eCQMs can use CQL expressions stored in shared libraries. Shared libraries enable similar logic to be used across multiple measures, improving standardization and harmonization across eCQMs. Shared libraries can exist at the local or global level in the MAT.

- **Local library**: CQL libraries can be shared among several eCQMs or a specific group of eCQMs within the MAT.

  The local CQL library ‘Hospice’ is used in a CQL definition statement below. The ‘Hospice’ library contains one CQL definition named “Has Hospice”.

  **Example CQL library: Hospice**

  Denominator Exclusions
  Hospice."Has Hospice"
  or exists "Malignant Neoplasm"
  or exists "Total Colectomy Performed"
  or FrailtyLTI."Advanced Illness and Frailty Exclusion Not Including Over Age 80"
  or ( exists ["Patient Characteristic Birthdate": "Birth date"] BirthDate
  where ( Global."CalendarAgeInYearsAt" ( BirthDate.birthDatetime, start of "Measurement Period" ) >= 65 )
  and FrailtyLTI."Long Term Care Periods Longer Than 90 Consecutive Days"
  )

- **Global library**: A global library is a shared CQL library created in the MAT that contains CQL expressions for all measure developers to use when specifying an eCQM. The common library, MATGlobalCommonFunctions, is accessible to all MAT users and is maintained and updated by Mathematica. The common library with each published update to the Quality Data Model (QDM) and CQL standards incorporated into the MAT. Using this common library reduces duplication and maintains consistency across measure specifications. The version of the common library to be used with each measure update is specified on the eCQI Resource Center site.

  The CQL library ‘Global’ is used in the definition below. Note: The common library definition “Inpatient Encounter” is recommended for use in hospital measures.

  **Add CQL Library with Alias:**
  include MATGlobalCommonFunctions version '6.2.000' called Global

  **Example CQL Library Name: Global**

  Global."Inpatient Encounter"
  ["Encounter, Performed": "Encounter Inpatient"]EncounterInpatient
  where "LengthInDays"(EncounterInpatient.relevantPeriod)\(\leq 120\)
and EncounterInpatient.relevantPeriod ends during day of "Measurement Period"

A. **Best practices for using CQL libraries**

   A local library should be used when similar functions or definitions are used across measures; it is required if five or more measures are affected.

B. **Best practices for naming CQL libraries**

   The measure name and measure library name do not have to be the same. When naming a new measure CQL library, adhere to the following standards:

   - **DO** use PascalCase.³
     
     **Example CQL library name:** AdultOutpatientEncounters

   - **DO** use names that are short, descriptive, and easy to read and that accurately reflect the contents of the library.
     
     **Example CQL library name:** Hospice

   - **DO NOT** start the library name with a special character,⁴ number, or underscore.
     
     **Example:** *AdultEncounters

   - **DO NOT** use spaces or special characters⁴ in the library name.
     
     **Example:** Adult+PediatricEncounter

³ See Appendix B for a complete list of case type definitions.

⁴ Special characters are symbols such as +, -, *, and /.
4. STANDARDS—DEFINITIONS

Definitions are concise logical CQL expressions that explain the meaning of measure concepts. Definitions are also referenced in the measure population logic. They should be reused and referenced in other CQL expressions, whenever appropriate.

A. Best practices for writing CQL definitions

When writing definitions, use the following best practices:

- **DO** use a ‘with’ or ‘without’ statement with a ‘such that’ statement when comparing two different data types or data sources.
  
  *Example of CQL definition using ‘with’:

  Encounte*er with Antibiotic Ordered within Three Days

  "Qualifying Encounter" EDOrAmbulatoryVisit
  with ["Medication, Order": "Antibiotic Medications for Pharygitis"] AntibioticOrdered
  such that ( EDOrAmbulatoryVisit.relevantPeriod starts 3 days or less on or before EDOrAmbulatoryVisit.authorDatetime )

- **DO** use a ‘from’ statement when comparing more than two sources of information.
  
  *Example of CQL definition using ‘from’:

  Most Recent Adult Depression Screening Positive and Follow Up Provided

  from
  "Most Recent Adult Depression Screening" LastAdultScreen,
  "Follow Up for Positive Adult Depression Screening" FollowUpPositiveAdultScreen,
  "Qualifying Encounter During Measurement Period" QualifyingEncounter
  where LastAdultScreen.relevantDatetime 14 days or less on or before day of start of QualifyingEncounter.relevantPeriod
  and LastAdultScreen.result in "Positive Depression Screening"
  and ( FollowUpPositiveAdultScreen.authorDatetime same day as end of QualifyingEncounter.relevantPeriod
  or FollowUpPositiveAdultScreen.relevantDatetime same day as end of QualifyingEncounter.relevantPeriod
)

B. Best practices for naming CQL definitions

When naming definitions, use the following best practices:
• **DO** use Title Case.\(^5\)

  **Example CQL definition name:**
  
  Baseline DEXA Scan Two Years Prior to the Start of or Less than Three Months After the Start of ADT

• **DO** create names that are easy to read, describe the contents of the logical expression, and provide context as to what makes the statement unique and clinically meaningful.

  **Example CQL definition names:**
  
  End Stage Renal Disease Encounter
  Estimated Gestational Age Captured on the Day of Delivery
  Initial Congestive Heart Failure Functional Assessment

• **DO** use only abbreviations or acronyms that are consistent with terminology used in the measure’s narrative.

  **Example CQL definition names:**
  
  Encounter with Prior or Present Diagnosis of Atrial Fibrillation or VTE
  ASCVD Procedure before End of Measurement Period

• **DO** create definition names that are clear and indicate the return.\(^6\) For example, a yes/no return should be named like a question, using the words “Is” or “Has”. A list of encounters should be named “Encounters…”, while a list of procedures should be named “Procedures…”

  **Example CQL definition names:**
  
  Has Initial Major Depression Diagnosis
  Has Complete Hearing Screening
  Has Continuation of Treatments
  Has Liver Disease
  Qualifying Encounters
  Procedures Identifying Sexual Activity
  Ischemic Stroke Encounters with Discharge Status

• **DO NOT** give a definition the same name as a value set.\(^7\)

  **Example:**
  
  Cognitive Assessment

---

5 See Appendix B for a complete list of case type definitions.

6 The MAT will display the “Return Type” in the definition form (when there are no CQL errors).

7 In CQL, a definition name and value set name cannot share the same identifier.
"Intervention, Performed": "Cognitive Assessment"

- DO NOT give a definition the same name as a CQL operator.  
- DO NOT use special characters in definition names.

Example:

% Patients with Test Result

Use Table 1 as a guide for naming definitions. The left column provides examples of definition names, and the right column shows alternatives that offer improved description and readability.

### Table 1. Making good definition names better (more clear and concise)

<table>
<thead>
<tr>
<th>Good definition name</th>
<th>Better definition name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anticoagulant Not Given at Discharge</td>
<td>Reason for Not Giving Anticoagulant at Discharge</td>
</tr>
<tr>
<td>In Demographic</td>
<td>Single Live Birth Encounter with Gestational Age 37 Weeks or More</td>
</tr>
<tr>
<td>Lab Test with Result</td>
<td>Most Recent Elevated HbA1c with Result</td>
</tr>
</tbody>
</table>

Below are more examples of CQL definitions that use descriptive and clinically meaningful text, title case, and appropriate spacing.

**Example CQL definition:**

**Newborn Fed Breast Milk Only Since Birth**

"Single Live Birth Encounter With Gestational Age 37 Weeks or More" QualifyingEncounter with ["Substance, Administered": "Breast Milk"] BreastMilkFeeding such that BreastMilkFeeding.relevantPeriod starts during QualifyingEncounter.relevantPeriod without ["Substance, Administered": "Dietary Intake Other than Breast Milk"] OtherFeeding such that OtherFeeding.relevantPeriod starts during QualifyingEncounter.relevantPeriod

**Example CQL definition:**

**Has Most Recent HbA1c**

Last(["Laboratory Test, Performed": "HbA1c Laboratory Test"] HbA1cNoResult where HbA1cNoResult.relevantDatetime during "Measurement Period" sort by resultDatetime)

---

8 Please see the CQL specification for a full list of operators: [https://cql.hl7.org/STU4/02-authorsguide.html#operations](https://cql.hl7.org/STU4/02-authorsguide.html#operations)
C. Standards for naming definitions across measures

Use Table 2 as a guide for naming definitions that use common concepts across measures. The left column presents the concept, and the right column shows the recommended standard naming convention.

Table 2. Standard definition names for use across measures

<table>
<thead>
<tr>
<th>Concept</th>
<th>Recommended definition name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hospice Exclusions</td>
<td>Has Hospice</td>
</tr>
<tr>
<td>Exclusions for Hospice</td>
<td></td>
</tr>
<tr>
<td>Encounters or</td>
<td>Qualifying Encounters</td>
</tr>
<tr>
<td>Eligible Encounters or</td>
<td></td>
</tr>
<tr>
<td>Valid Encounters</td>
<td></td>
</tr>
</tbody>
</table>
5. STANDARDS—ALIASES OR ARGUMENT NAMES

Aliases, or argument names, are identifiers that refer to individual CQL expressions or libraries. Aliases should correlate clearly to their source and can be reused to avoid restating key expressions. This allows for a more fluid, concise, and standardized CQL expression. Alias names should maintain their meaning and uniformity within and across measures. Authors can develop aliases for libraries, functions, and definitions.

A. Best practices for using CQL aliases and argument names

When naming aliases and argument names, use the following standards:

- **DO** use PascalCase.9

  Example CQL aliases:

  
  ```cql
  HighRiskMedicationOrder
  Referral
  ```

- **DO** use names that are short, descriptive, and easy to read and that accurately reflect the identified concept.

  Example CQL aliases:

  ```cql
  EggAllergy
  ColorectalCancer
  FluVaccination
  ```

- **DO** create alias names that are clinically focused.

  Example CQL aliases:

  ```cql
  DiastolicBP
  BetaBlockerOrdered
  InitiatedTreatment
  ```

---

9 See Appendix B for a complete list of case-type definitions.
• **DO** use only abbreviations or acronyms that are consistent with terminology used in the measure’s narrative sections.

  Example CQL aliases (note: in each of these examples, the acronym is defined in the narrative):

  ```cql
  VR12PhysicalScore
  NoVTEMedication
  ```

• **DO NOT** reuse aliases, unless a single alias carries the same content and scope as its original definition when reused in a measure.

  See below for an example to avoid. The alias `HeartRate` is reused in two different definitions in the same measure, each with a different scope.

  Example:

  ```cql
  First(["Encounter, Performed": "Heart Rate Visit"] HeartRate
       with ["Diagnosis": "Essential Hypertension"] Hypertension
       such that HeartRate.relevantPeriod overlapsHypertension.prevalencePeriod)
  Last(["Physical Exam, Performed": "Heart Rate Exam"] HeartRate
       with "Initial Blood Pressure Visit" InitialEncounter
       such that HeartRate.relevantDatetime during InitialEncounter.relevantPeriod
       where HeartRate.result is not null
       sort by start of relevantDatetime)
  ```

• **DO NOT** give an alias the same name as the definition name.

  Example:

  ```cql
  LowerBackProcedure
  ```

• **DO NOT** use an alias if the definition statement does not require additional logic.

  Example:

  ```cql
  BloodTransfusion
  ```
Use Table 3 as a guide for naming aliases. The left column lists examples of alias names that measure developers should avoid. The alternatives in the right column offer improved descriptions and readability.

**Table 3. Making aliases easier to read and more clinically focused**

<table>
<thead>
<tr>
<th>Alias names to avoid</th>
<th>Better alias names</th>
</tr>
</thead>
<tbody>
<tr>
<td>D or Dx or Diagnosis</td>
<td>HeartFailure</td>
</tr>
<tr>
<td></td>
<td>Pregnancy</td>
</tr>
<tr>
<td></td>
<td>Asthma</td>
</tr>
<tr>
<td></td>
<td>Bradycardia</td>
</tr>
<tr>
<td>Med Medication</td>
<td>BetaBlocker</td>
</tr>
<tr>
<td></td>
<td>Antidepressant</td>
</tr>
<tr>
<td>P or Proc or Procedure</td>
<td>CardiacSurgery</td>
</tr>
<tr>
<td></td>
<td>Dialysis</td>
</tr>
<tr>
<td>Lab or LabTest</td>
<td>HepBAntigenTest</td>
</tr>
<tr>
<td></td>
<td>MumpsTiter</td>
</tr>
<tr>
<td></td>
<td>PregnancyTest</td>
</tr>
<tr>
<td>E or Enc</td>
<td>Encounter* (use with caution if referring to several</td>
</tr>
<tr>
<td></td>
<td>types of encounters in measure)</td>
</tr>
<tr>
<td></td>
<td>InpatientEncounter</td>
</tr>
<tr>
<td></td>
<td>HeartFailureEncounter</td>
</tr>
<tr>
<td></td>
<td>Psychotherapy</td>
</tr>
<tr>
<td>{[&quot;Physical Exam, Performed&quot;: &quot;Heart Rate&quot;] Exam}</td>
<td>HeartRate</td>
</tr>
<tr>
<td>{[&quot;Diagnostic Study, Performed&quot;: &quot;Ejection Fraction&quot;] Study}</td>
<td>EjectionFraction</td>
</tr>
</tbody>
</table>

Use Table 4 as a guide for improving alias names even further. The left column lists examples of aliases. The right column includes alternatives that offer improved descriptions and clarity.

**Table 4. Making good alias names more descriptive**

<table>
<thead>
<tr>
<th>Good alias name</th>
<th>Better alias name</th>
</tr>
</thead>
<tbody>
<tr>
<td>HeartRate</td>
<td>FirstHeartRate</td>
</tr>
<tr>
<td>AntithromboticNotGiven</td>
<td>NoAntithrombotic</td>
</tr>
<tr>
<td>VisualExam</td>
<td>VisualFootExam</td>
</tr>
<tr>
<td>Fracture</td>
<td>LowerBodyFracture</td>
</tr>
<tr>
<td>THAProcedure</td>
<td>TotalHip</td>
</tr>
<tr>
<td>HeightExam</td>
<td>Height</td>
</tr>
</tbody>
</table>

Use Table 5 as a guide for creating distinctions between two aliases with similar characteristics within a measure by adding specificity.
Table 5. Differentiating between aliases with similar concepts by adding specificity

<table>
<thead>
<tr>
<th>Similar concepts</th>
<th>Similar alias names with specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart failure encounter and Heart failure diagnosis</td>
<td>HeartFailureEncounter and HeartFailureDiagnosis</td>
</tr>
</tbody>
</table>

B. Standards for naming aliases across measures

Use Table 6 as a guide for naming aliases that use common concepts across measures. The left column presents the alias concept, and the right column presents the recommended standard alias naming convention.

Table 6. Standard alias names recommended for use across measures

<table>
<thead>
<tr>
<th>Concept</th>
<th>Standardized alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient date of birth</td>
<td>BirthDate</td>
</tr>
<tr>
<td>Hospice discharge</td>
<td>DischargeHospice</td>
</tr>
<tr>
<td>Hospice care order</td>
<td>HospiceOrder</td>
</tr>
<tr>
<td>Hospice intervention performed</td>
<td>HospicePerformed</td>
</tr>
</tbody>
</table>

Also see the following examples of standard alias names.

- The alias BirthDate is used consistently in different CQL definition statements, and in multiple places within and across measures.

- **Example CQL alias: BirthDate**
  ```cql
  exists ( ["Patient Characteristic Birthdate": "Birth date"]
  BirthDate
  where Global."CalendarAgeInYearsAt"(BirthDate.birthDatetime, start of "Measurement Period")>= 18
  )
  ```

- **Example CQL alias: DischargeHospice, HospiceOrder, HospicePerformed**
  ```cql
  exists ( ["Encounter, Performed": "Encounter Inpatient"]
  DischargeHospice
  where ( DischargeHospice.dischargeDisposition ~ "Discharge to home for hospice care (procedure)"
  or DischargeHospice.dischargeDisposition ~ "Discharge to healthcare facility for hospice care (procedure)"
  )
  and DischargeHospice.relevantPeriod ends during "Measurement Period"
  )
  or exists ( ["Intervention, Order": "Hospice care ambulatory"]
  HospiceOrder
  where HospiceOrder.authorDatetime during "Measurement Period"
  )
  ```
or exists ( ["Intervention, Performed": "Hospice care ambulatory"] HospicePerformed where HospicePerformed.relevantPeriod overlaps "Measurement Period"
)

Note: Use ~ equivalence operator for code comparison, but use in() function to compare to Value Set Authority Center object identifiers (VSAC OIDs).
6. FUNCTIONS

A function is a named CQL expression that can perform any variety of calculations. Before creating new functions, measure developers should review and—to the extent possible and applicable—use the predefined functions available in the MAT or in the shared “Global” common library. Functions act on the input arguments passed to them, whereas definitions operate only on the expressions in the definition.

A. Best practices and standards for naming new CQL functions

New function names should be short, descriptive, and easy to read and should provide an expression that accurately represents the identified concept. When naming functions, use the following standards:

- **DO** use PascalCase.\(^{10,11}\)
  
  **Example CQL function name:**
  
  "EmergencyDepartmentArrivalTime"(Encounter "Encounter, Performed")

- **DO** use spaces after commas to separate arguments, always.
  
  **Example CQL function:**
  
  Global."CalendarAgeInYearsAt"(BirthDateTime DateTime, AsOf DateTime)

- **DO NOT** give a function the same name as a MAT predefined function.
  
  **Process step:** Please review the predefined operators available in the MAT\(^{12}\) to ensure that the name of your function is not the same.

B. Selecting functions

To differentiate similar functions, choose from the predefined list in the MAT or from the “Global” common library. Select the function that is most appropriate to meet the measure’s intent. See the following examples of preferred functions from the “Global” common library.

- ‘Normalize Interval’ function:
  
  **Example:** “Global” CQL function:
  
  Global."NormalizeInterval"(pointInTime DateTime, period Interval<DateTime>):

---

\(^{10}\) See Appendix B for complete definitions of case types.

\(^{11}\) The use of double quotes in invoking is not required and should be discouraged.

\(^{12}\) See the MAT user guide for a list of operators: [https://www.emeasuretool.cms.gov/training-resources/user-guide](https://www.emeasuretool.cms.gov/training-resources/user-guide).
if pointInTime is not null then Interval[pointInTime, pointInTime]
else if period is not null then period
else null as Interval<DateTime>

- Use the ‘Normalize Interval’ function for QDM datatypes that have use cases for both a relevantDateTime and a relevantPeriod to reduce implementation burden associated with variable use of timing attributes across measures.

- Use the ‘Normalize Interval’ function for the following QDM datatypes:
  - Assessment, Performed
  - Device, Applied
  - Diagnostic Study, Performed
  - Intervention, Performed
  - Laboratory Test, Performed
  - Medication, Administered
  - Medication, Dispensed
  - Physical Exam, Performed
  - Procedure, Performed
  - Substance, Administered

- **Age functions:**

  Example CQL function:

  ```cql
  Global."CalendarAgeInYearsAt"( BirthDateTime DateTime, AsOf DateTime)
  ```

  - This function calculates age using **only birth date**, not time.
  - Use this function for measures pertaining to adult patients, for example to determine all adult patients aged 65 or older at the beginning of the Measurement Period, or as applicable.

  Example CQL function (used in logic):

  ```cql
  exists ( ["Patient Characteristic Birthdate": "Birth date"]
  BirthDate
  where Global."CalendarAgeInYearsAt"(BirthDate.birthDatetime, start of "Measurement Period")>= 65
  ```

- **Length-of-stay functions** (generally used for hospital measures):

  Example CQL function: `Global."LengthInDays"()`
LengthInDays() calculates the difference in calendar days between the start and end of the given interval. Timing intervals should always be noted in chronological order as [start, finish] to avoid negative time intervals.

Global."LengthInDays"(Value Interval<DateTime>)

difference in days between start of Value and end of Value

This function can be used to calculate the length of a hospital stay for an inpatient encounter from admission to discharge. Also see the examples below.

Example CQL function (used in logic):

"Encounter, Performed": "Encounter Inpatient"
EncounterInpatient
where LengthInDays(EncounterInpatient.relevantPeriod) <= 120
and EncounterInpatient.relevantPeriod ends during day of "Measurement Period"

Example CQL function: Global."HospitalizationLengthOfStay"()

Returns the length of stay in days (i.e., the number of days between admission and discharge) for the given encounter, or from the admission of any immediately prior emergency department visit to the discharge of the encounter.

Example CQL function (used in logic):

TJC."Ischemic Stroke Encounter" IschemicStrokeEncounter
where Global."HospitalizationLengthOfStay"(IschemicStrokeEncounter) <= 2

Example CQL function: Global."Hospitalization"()

This function returns the total interval for admission to discharge for the given encounter, or for the admission of any immediately prior emergency department visit to the discharge of the given encounter.

Example CQL function:

Definition name: Encounter Visit
let EDVisit: Last("Encounter, Performed": "Emergency Department Visit") LastED
where LastED.relevantPeriod ends 1 hour or less on or before start of Visit.relevantPeriod
sort by
end of relevantPeriod
return Interval[Coalesce(start of EDVisit.relevantPeriod, 
start of Visit.relevantPeriod), 
end of Visit.relevantPeriod]

Example CQL function (used in logic):
"Ischemic Stroke Encounter" IschemicStrokeEncounter 
with "Intervention Comfort Measures" ComfortMeasure 
such that Coalesce(start of 
ComfortMeasure.relevantPeriod, 
ComfortMeasure.authorDatetime)during 
Global."Hospitalization"(IschemicStrokeEncounter)
7. OTHER CQL BEST PRACTICES

A. Population criteria

When using population criteria, be descriptive and specific, making sure names are easy to read. Below is an example of how to improve the naming of population criteria.

<table>
<thead>
<tr>
<th>Current population criteria</th>
<th>Improved population criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Population 'In Demographic'</td>
<td>Initial Population 'Single Live Birth Encounter with Gestational Age 37 Weeks or More'</td>
</tr>
</tbody>
</table>

When the denominator population criteria are equivalent to the initial population criteria, state “Initial Population” for the Denominator, as shown below:

Example CQL: Initial Population = Denominator

Initial Population

exists ( ["Patient Characteristic Birthdate": "Birth date"]
BirthDate
where
Global."CalendarAgeInYearsAt"(BirthDate.birthDatetime,
start of "Measurement Period") >= 65
) and exists "Qualifying Encounters"

Denominator
"Initial Population"

- **DO** use closed brackets, [ and ], instead of open brackets, ( and ), for definitions containing age intervals to avoid inconsistencies and to harmonize logic across measures. In the example below, the definition uses Interval[45, 65] to include patients ages 45 to 65 but exclude patients ages 44 and younger and 66 and older at the start of the measurement period.

Example in CQL of Age Interval:

exists ( ["Patient Characteristic Birthdate": "Birth date"]
BirthDate
where Global."CalendarAgeInYearsAt"(BirthDate.birthDatetime,
start of "Measurement Period") in Interval[45, 65]
)

B. Additional timing phrases

CQL supports precision-based date/time comparisons. Be sure to consider whether day or time should be considered in timing phrases. Additional timing phrases, besides **day of** expressions, may be needed when making a statement such as A starts/ends before/after or concurrent with start of B. However, when anchoring a timing to the parameter “Measurement Period,” it is not necessary to use **day of**.
Example CQL timing phrase: **ends 1 day after day of start of**

A supplementary timing constraint is added to ensure the timing of the relevant period for the AnesthesiaProcedure ends 1 day after the start of relevant period of the QualifyingEncounter.

```
from
"Encounter with Age Range and without VTE Diagnosis or Obstetrical Conditions" QualifyingEncounter,
["Procedure, Performed": "General or Neuraxial Anesthesia"]
AnesthesiaProcedure,
"No VTE Prophylaxis Medication Administered or Ordered"
NoVTEMedication
where NoVTEMedication.negationRationale in "Medical Reason"
    and AnesthesiaProcedure.relevantPeriod ends 1 day after
day of start of QualifyingEncounter.relevantPeriod
    and NoVTEMedication.authorDatetime during
TJC."CalendarDayOfOrDayAfter"(end of
AnesthesiaProcedure.relevantPeriod)
return QualifyingEncounter
```

### C. Operator precedence

Precedence in CQL expressions is determined by the order of appearance in the expression, left to right. To ensure consistent and predictable behavior in the order of operations within CQL expressions, use parentheses around a grouping to enforce higher precedence. See the table in Appendix A for more details.

In the example below, parentheses are used to promote operator precedence around **exists** ‘Left Mastectomy Diagnosis’ and **exists** ‘Left Mastectomy’ and to make the groupings clear.

**Example CQL operator precedence:**

```
Denominator Exclusions
Hospice."Has Hospice"
    or ( ( exists ( "Right Mastectomy Diagnosis" )
        or exists ( "Right Mastectomy Procedure" )
    )
        and ( exists ( "Left Mastectomy Diagnosis" )
            or exists ( "Left Mastectomy Procedure" )
        )
    )
    or exists "Bilateral Mastectomy Diagnosis"
    or exists "Bilateral Mastectomy Procedure"
    or FrailtyLTI."Advanced Illness and Frailty Exclusion Not Including Over Age 80"
    or ( exists ["Patient Characteristic Birthdate": "Birth date"]
BirthDate
        where ( Global."CalendarAgeInYearsAt"(BirthDate.birthDatetimetime, start of "Measurement Period") >= 65 )
```

---

19
D. Direct Reference Codes

Measure developers maintain and publish value sets on the Value Set Authority Center web site. Value sets and codes are listed in the terminology section of each measure specification. Direct reference codes (DRCs) are single terminology codes that can be referenced directly within CQL logic, instead of creating a single code value set. DRCs are recommended for all single-use LOINC codes and may be used for other single-use terminology codes. When included in a definition, a DRC is incorporated in the CQL syntax through use of the code descriptor. The DRC’s specific code and corresponding descriptor will always be called out in the Terminology section of the human readable; it may also be referenced in the Data Criteria section if used as part of a QDM element, not just as an attribute of a previously defined QDM element.

- **DO NOT** include the version of the DRC directly in the CQL. This causes downstream issues in MAT and Bonnie.

**Example of DRC without a version number:**

```
o code "Birth date": '21112-8' from "LOINC" display 'Birth date'
```

**Examples of DRCs used in CQL logic:**

**Denominator Exclusions**

```
"Initial Population" LiveBirthEncounter
where LiveBirthEncounter.dischargeDisposition ~ "Patient deceased during stay (discharge status = dead) (finding)"
and not exists "Has Complete Hearing Screening"
```

**Rotavirus numerator inclusion conditions**

```
( ( ["Diagnosis": "Anaphylaxis due to rotavirus vaccine (disorder)" ]
  union ["Diagnosis": "Severe Combined Immunodeficiency"]
  union ["Diagnosis": "Intussusception"] ) RotavirusConditions
  with ["Patient Characteristic Birthdate":"Birth date"]
BirthDate
  such that
Global."CalendarAgeInDaysAt"(BirthDate.birthDatetime, start of RotavirusConditions.prevalencePeriod) <= 730)
```
## VERSION HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description of change</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>August 21, 2018</td>
<td>Initial publication</td>
</tr>
<tr>
<td>3.0</td>
<td>May 2019</td>
<td>Removed references related to retired QDM logic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated examples of logic to reflect most recent use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added additional examples of logic in each section to provide context</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added clarifying language to content in each section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated language and content to align with standards changes QDM 5.4 and CQL 1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed reference to Keyword-Distinct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated examples using birthdate to reflect the addition of birth date, a direct reference code</td>
</tr>
<tr>
<td>4.0</td>
<td>May 2020</td>
<td>Updated language and content to align with standards changes QDM 5.5 and CQL 1.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated examples of logic to reflect most recent use</td>
</tr>
<tr>
<td>5.0</td>
<td>May 2021</td>
<td>Removed references related to retired QDM logic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated examples of logic to reflect most recent use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added section on best practices for writing definitions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added guidance on creating age intervals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added ‘Normalize Interval’ function to the ‘Selecting functions’ section</td>
</tr>
</tbody>
</table>
APPENDIX A:

THE ORDER OF OPERATOR PRECEDENCE IN CQL
Table A.1. Order of operator precedence in CQL (highest to lowest)

<table>
<thead>
<tr>
<th>Category</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>. [] ()</td>
</tr>
<tr>
<td>Conversion Phrase</td>
<td>convert..to</td>
</tr>
<tr>
<td>Unary Arithmetic</td>
<td>unary +/−</td>
</tr>
<tr>
<td>Extractor</td>
<td>start end difference duration width successor predecessor of component singleton from</td>
</tr>
<tr>
<td>Exponentiation</td>
<td>^</td>
</tr>
<tr>
<td>Multiplicative</td>
<td>* / div mod</td>
</tr>
<tr>
<td>Additive</td>
<td>+ - &amp;</td>
</tr>
<tr>
<td>Conditional</td>
<td>if..then..else case..else..end</td>
</tr>
<tr>
<td>Unary List</td>
<td>distinct collapse flatten expand</td>
</tr>
<tr>
<td>Unary Test</td>
<td>is null true false</td>
</tr>
<tr>
<td>Type Operators</td>
<td>is as cast..as</td>
</tr>
<tr>
<td>unary Logical</td>
<td>not exists</td>
</tr>
<tr>
<td>Between</td>
<td>between precision between duration in precision between difference in precision between</td>
</tr>
<tr>
<td>Comparison</td>
<td>&lt;= &lt; &gt; &gt;=</td>
</tr>
<tr>
<td>Timing Phrase</td>
<td>same as includes during before/after within</td>
</tr>
<tr>
<td>Interval Operators</td>
<td>meets overlaps starts ends</td>
</tr>
<tr>
<td>Equality</td>
<td>= != ~ !~</td>
</tr>
<tr>
<td>Membership</td>
<td>in contains</td>
</tr>
<tr>
<td>Conjunction</td>
<td>and</td>
</tr>
<tr>
<td>Disjunction</td>
<td>or xor</td>
</tr>
<tr>
<td>Implication</td>
<td>implies</td>
</tr>
<tr>
<td>Binary List</td>
<td>union intersect except</td>
</tr>
</tbody>
</table>

Source: https://cql.hl7.org/03-developersguide.html#operator-precedence
APPENDIX B:

CASE-TYPE DEFINITIONS
Case-Type Definitions

(Note: CQL is a case-sensitive language)

- **lowercase** – All letters are lowercase
- **camelCase** – First letters of words are capitalized except for the first word, with no whitespace characters allowed (used for QDM attributes)
- **PascalCase** – First letters of words are capitalized, including words not capitalized in Title Case such as “and” and “of,” with no whitespace characters allowed
- **Title Case** – Standard title casing including spaces and tabs, but no other whitespace characters allowed