Cooking with CQL Q&As

Session 54 - Thursday, May 27, 2021

# Using Fast Healthcare Interoperability Resources (FHIR®)

**Q:** When calculating Cumulative Medication Duration in Fast Healthcare Interoperability Resources (FHIR®), are refills considered in a medication dispense for individual prescription refills?

**A:** Prescription refills are disregarded in a medication dispense period because there is an expectation that there are multiple dispense records: a dispense record for the initial fill and a separate dispense record for each refill.

|  |  |
| --- | --- |
| define function MedicationDispensePeriod(Dispense "MedicationDispense"): | |
|  | Dispense D |
|  | let |
|  | dosage: singleton from D.dosageInstruction, |
|  | doseAndRate: singleton from dosage.doseAndRate, |
|  | doseRange: doseAndRate.dose as Range, |
|  | doseQuantity: doseAndRate.dose as SimpleQuantity, |
|  | dose: Coalesce(end of doseRange, doseQuantity), |
|  | timing: dosage.timing, |
|  | frequency: Coalesce(timing.repeat.frequencyMax, timing.repeat.frequency), |
|  | period: System.Quantity { value: timing.repeat.period, unit: timing.repeat.periodUnit.value }, |
|  | dosesPerDay: Coalesce(ToDaily(frequency, period), Count(timing.repeat.timeOfDay), 1.0), |
|  | startDate: |
|  | Coalesce( |
|  | D.whenHandedOver, |
|  | D.whenPrepared |
|  | ) |
|  | return |
|  | Interval[startDate, startDate + Coalesce(D.daysSupply, D.quantity / (dose \* dosesPerDay))] |

**Q:** Regarding Cumulative Medication Duration in FHIR, what logic expression should the measure developer use for a prescription specifying a number of tablets and uses a semantic clinical drug (SCD) concept that specifies the tablet strength (e.g., lisinopril 10 mg tablet, administer two tablets)?

**A:** The MedicationRequestPeriod logic expression applies the doseQuantity function to calculate the prescription amount, depending on the preparation of the medication (e.g., tablets, liquid, patches). The code for the particular medication (e.g., lisinopril) provides the dosage strength.

|  |  |
| --- | --- |
| define function MedicationRequestPeriod(Request "MedicationRequest"): | |
|  | Request R |
|  | let |
|  | dosage: singleton from R.dosageInstruction, |
|  | doseAndRate: singleton from dosage.doseAndRate, |
|  | doseRange: doseAndRate.dose as Range, |
|  | doseQuantity: doseAndRate.dose as SimpleQuantity, |
|  | dose: Coalesce(end of doseRange, doseQuantity), |
|  | timing: dosage.timing, |
|  | frequency: Coalesce(timing.repeat.frequencyMax, timing.repeat.frequency), |
|  | period: System.Quantity { value: timing.repeat.period, unit: timing.repeat.periodUnit.value }, |
|  | dosesPerDay: Coalesce(ToDaily(frequency, period), Count(timing.repeat.timeOfDay), 1.0), |
|  | boundsPeriod: timing.repeat.bounds as Period, |
|  | daysSupply: R.dispenseRequest.expectedSupplyDuration, |
|  | quantity: R.dispenseRequest.quantity, |
|  | refills: Coalesce(R.dispenseRequest.numberOfRepeatsAllowed, 0), |
|  | startDate: |
|  | Coalesce( |
|  | start of boundsPeriod, |
|  | start of R.dispenseRequest.validityPeriod, |
|  | R.authoredOn |
|  | ) |
|  | return |
|  | if HasEnd(boundsPeriod) then |
|  | Interval[startDate, end of boundsPeriod] |
|  | else |
|  | Interval[startDate, startDate + Coalesce(daysSupply, quantity / (dose \* dosesPerDay)) \* (1 + refills)] |

**Q:** Regarding Cumulative Medication Duration in Fast Healthcare Interoperability Resources (FHIR®), within the define function of the MedicationDispensePeriod logic expression, when would a measure developer use the start of boundsPeriod compared to D.whenPrepared? The structure of the Coalesce line of code is written in a way that suggests start of boundsPeriod is the preferential timing attribute for the MedicationDispensePeriod logic expression in all cases.

|  |  |
| --- | --- |
| define function MedicationDispensePeriod(Dispense "MedicationDispense"): | |
|  | Dispense D |
|  | let |
|  | dosage: singleton from D.dosageInstruction, |
|  | doseAndRate: singleton from dosage.doseAndRate, |
|  | doseRange: doseAndRate.dose as Range, |
|  | doseQuantity: doseAndRate.dose as SimpleQuantity, |
|  | dose: Coalesce(end of doseRange, doseQuantity), |
|  | timing: dosage.timing, |
|  | frequency: Coalesce(timing.repeat.frequencyMax, timing.repeat.frequency), |
|  | period: System.Quantity { value: timing.repeat.period, unit: timing.repeat.periodUnit.value }, |
|  | dosesPerDay: Coalesce(ToDaily(frequency, period), Count(timing.repeat.timeOfDay), 1.0), |
|  | boundsPeriod: timing.repeat.bounds as Period, |
|  | startDate: |
|  | Coalesce( |
|  | start of boundsPeriod, |
|  | D.whenHandedOver, |
|  | D.whenPrepared |
|  | ) |
|  | return |
|  | if HasEnd(boundsPeriod) then |
|  | Interval[startDate, end of boundsPeriod] |
|  | else |
|  | Interval[startDate, startDate + Coalesce(D.daysSupply, D.quantity / (dose \* dosesPerDay))] |
|  |

**A:**  The CQI/Clinical Decision Support (CDS) Work Group discussed this topic at the meeting with the Pharmacy Work Group on May 28, 2021, where the Work Groups agreed that the start of boundsPeriod should not be used to determine when the medication becomes available to the patient. The Work Groups determined the preference is whenHandedOver, but if unavailable, then whenPrepared is a reliable boundary. Therefore, the Work Group removed the start of boundsPeriod from the MedicationDispensePeriod definition in the updated version.

|  |  |
| --- | --- |
| define function MedicationDispensePeriod(Dispense "MedicationDispense"): | |
|  | Dispense D |
|  | let |
|  | dosage: singleton from D.dosageInstruction, |
|  | doseAndRate: singleton from dosage.doseAndRate, |
|  | doseRange: doseAndRate.dose as Range, |
|  | doseQuantity: doseAndRate.dose as SimpleQuantity, |
|  | dose: Coalesce(end of doseRange, doseQuantity), |
|  | timing: dosage.timing, |
|  | frequency: Coalesce(timing.repeat.frequencyMax, timing.repeat.frequency), |
|  | period: System.Quantity { value: timing.repeat.period, unit: timing.repeat.periodUnit.value }, |
|  | dosesPerDay: Coalesce(ToDaily(frequency, period), Count(timing.repeat.timeOfDay), 1.0), |
|  | startDate: |
|  | Coalesce( |
|  | D.whenHandedOver, |
|  | D.whenPrepared |
|  | ) |
|  | return |
|  | Interval[startDate, startDate + Coalesce(D.daysSupply, D.quantity / (dose \* dosesPerDay))] |

**Q:** Within the Cumulative Medication Duration in Fast Healthcare Interoperability Resources (FHIR®), is there a calculation that works well with adjacent intervals and the collapse function?

**A:** The most effective calculation involves the CumulativeDuration function for intervals and collapse. This logic provides the ability to look at the boundaries of the intervals for an adjacent collapse on a per day perspective supplying the necessary granularity to generate a successful collapse.

|  |  |
| --- | --- |
| define function CumulativeDuration(Intervals List<Interval<DateTime>>): | |
|  | Sum((collapse Intervals per day) X return all difference in days between start of X and end of X) |

Next, define a function that rolls out intervals.

|  |  |
| --- | --- |
| define function RolloutIntervals(intervals List<Interval<DateTime>>): | |
|  | intervals I |
|  | aggregate R starting (null as List<Interval<DateTime>>): |
|  | R union ({ |
|  | I X |
|  | let |
|  | S: Max({ end of Last(R) + 1 day, start of X }), |
|  | E: S + duration in days of X |
|  | return Interval[S, E] |
|  | }) |

**Q**: In Cumulative Medication Duration using the Fast Healthcare Interoperability Resources (FHIR®) standard, what logic expression should the measure developer use to accommodate an early refill on a medication dispense?

**A:** The RolloutIntervals function is the appropriate logic expression, which bumps out the intervals without altering the original dispense medication date. The current medication dispense date is set in the system (e.g., every 30 days). When there is a request for an early refill of the medication (e.g., a week early), the patient receives an add on supply of the medication equal to the time period of the current medication supply (e.g., 30 days). The roll out interval algorithm dispenses the second interval of medications (e.g., next 30 days) when the first medication interval has completed the original medication time period.

|  |  |
| --- | --- |
| define function CumulativeMedicationDuration( | |
|  | Medications List<Choice<"MedicationRequest", |
|  | "MedicationDispense", |
|  | "MedicationAdministration" |
|  | >>): |
|  | CumulativeDuration(( |
|  | Medications M |
|  | where M is MedicationRequest |
|  | return MedicationPeriod(M) |
|  | ) |
|  | union ( |
|  | RolloutIntervals( |
|  | Medications M |
|  | where M is MedicationDispense or M is MedicationAdministration |
|  | return MedicationPeriod(M) |
|  | ) |
|  | ) |
|  | ) |

# Measure Logic in CQL

**Q:** The expectation is for a stratifier to be a complete and disjoint partitioning of the population. For implementations that perform case-at-a-time processing, the calculation and collection of the stratifiers may be part of processing each time, whereas for implementations that evaluate at the population level, there may be independent calculation of the stratifiers. If the logic applies stratifiers to each set of population criteria by default, what is the advantage of applying a stratifier only within a numerator? What is a use case for which that scenario would come up?

define "Initial Population for Stratifier 1":

AgeInYearsAt(start of "Measurement Period") >= 6

and AgeInYearsAt(start of "Measurement Period") < 20

and exists ( "Qualifying Encounters" )

// AND the Stratifier 1 criteria

and AgeInYearsAt(start of "Measurement Period") >= 12

and AgeInYearsAt(start of "Measurement Period") < 20

define "Denominator":

"Initial Population"

// AND the Stratifier 1 criteria

and AgeInYearsAt(start of "Measurement Period") >= 12

and AgeInYearsAt(start of "Measurement Period") < 20

define "Denominator Exclusions":

Hospice."Has Hospice"

// AND the Stratifier 1 criteria

and AgeInYearsAt(start of "Measurement Period") >= 12

and AgeInYearsAt(start of "Measurement Period") < 20

define "Numerator":

exists ( ["Procedure": "Fluoride Varnish Application for Children"] FluorideApplication

where FluorideApplication.performed during "Measurement Period"

and FluorideApplication.status = 'completed'

)

// AND the Stratifier 1 criteria

and AgeInYearsAt(start of "Measurement Period") >= 12

and AgeInYearsAt(start of "Measurement Period") < 20

**A:** Generally, a stratifier reports the stratification for all measure criteria. From the code in the question, the initial population, including the numerator and denominator, and each stratum with the associated score can be seen. To calculate the score for the stratifier, one should partition each stratifier into a mini measure with the same criteria. The application of the stratifiers is to the criteria to establish that stratum score. The reason the measure developer might only apply the stratifier to the numerator is if they want the actual counts for the numerator and do not care what the counts are for the other components of the measure. In this case, the numerator is the reported result, but the measure developer still needs to calculate the full stratification in order to get the numerator value.

One example use case where one might only apply a stratifier to the numerator is CMS 111, the Emergency Department (ED) Measure, where the measure stratifies the measure population by mental health diagnoses, but does not stratify for the other populations.

**Q:** With the stratifications being somewhat independent of the measure logic itself, it would be interesting to put the stratifiers in a stratifier library, but if you do, how do you reference outside that measure to the library in Fast Healthcare Interoperability Resources® (FHIR)? This would effectively have a library referencing another library resource.

**A:** The use of a library, in this case a stratifier library, is a specific conformance requirement put on measures, and the Quality Measure Implementation Guide references specifically, so that all the expressions in a given measure are referencing the expressions in a single library. From an implementation perspective, that means the engine does not have to resolve library references outside of the engine, it only has to resolve references within that boundary.