Cooking with CQL Q&As

Session 68 - Thursday, October 27, 2022

Measure Logic in CQL

**Q:** When using the operation for unit conversion in Clinical Quality Language, in the example, does the operational logic perform the math as well as change the unit?

|  |  |
| --- | --- |
| // Explicit Conversions: https://cql.hl7.org/03-developersguide.html#explicit-conversion | |
|  | define ConvertValues: convert '1' to Integer | |
|  | define ConvertUnits: convert 5 'mg' to 'g' | |
|  | define InvalidConvert: convert 'Foo' to Integer // Results in null | |

**A:** The 'convert' operation performs the math and changes the unit. You must follow the Unified Code for Units of Measure conversion rules if a conversion is performed.

Measure Logic in CQL

**Q:** When casting with choice statements in Clinical Quality Language, in the if-then statement, the if statement value is set to a quantity, why is there a redundant statement to express the quantity as an integer function?

|  |  |
| --- | --- |
|  | |
| /\* | |
|  | @description: Returns the value, interpreted as an Integer | |
|  | @comments: If the input is a Quantity, the result is the same as invoking `ToInteger(value as Quantity)`. | |
|  | If the input is an Integer, the result is the same as invoking `value as Integer` | |
|  | \*/ | |
|  | define function ToInteger(value Choice<Integer, Quantity>): | |
|  | if value is Quantity then | |
|  | ToInteger(value as Quantity) | |
|  | else | |
|  | value as Integer | |
|  | |  |

**A:** In this CQL snippet, “ChoiceToInteger” accepts data value with a choice of integer or quantity. If not a quantity, it accepts the input attribute as an integer. If a quantity, it calls the QuantityToInteger function, which tests the value of quantity for appropriateness of conversion to integer and the unit as ‘1,’ which is the default Unified Code for Units of Measure. If both are suitable, it converts the quantity.value to an integer.

|  |  |
| --- | --- |
| /\*@description: Returns the value of the given Quantity, interpreted as an Integer if it is safe to do so. | |
|  | @comments: A Quantity value can be safely interpreted as an integer if it has no decimal component (i.e. zeros after the decimal), | |
|  | and it has the default UCUM unit of '1'\*/ | |
|  |  | |
|  | define function "QuantityToInteger"(value Quantity): | |
|  | case | |
|  | when Abs(value.value - Truncate(value.value))> 0.00000001 then Message(null, true, 'ToInteger.InvalidArgument', ErrorSeverity, 'The quantity has a non-zero decimal component and cannot be safely interpreted as an Integer') | |
|  | when value.unit != '1' then Message(null, true, 'ToInteger.InvalidUnit', ErrorSeverity, 'The quantity has non-default units specified and cannot be safely interpreted as an Integer') | |
|  | else Truncate(value.value) | |
|  | end | |
|  |  | |
|  | /\*@description: Returns the value, interpreted as an Integer | |
|  | @comments: If the input is a Quantity, the result is the same as invoking `ToInteger(value as Quantity)`. | |
|  | If the input is an Integer, the result is the same as invoking `value as Integer`\*/ | |
|  |  | |
|  | define function "ChoiceToInteger"(value Choice<Integer, Quantity>): | |
|  | if value is Quantity then "QuantityToInteger"(value as Quantity) | |
|  | else value as Integer | |
|  |  | |

Measure Logic in CQL

**Q:** When using choice types in Clinical Quality Language, why is a unit of '1' used instead of a Unified Code for Units of Measure (UCUM) annotation?

|  |  |
| --- | --- |
| /\* | |
|  | @description: Returns the value of the given Quantity, interpreted as an Integer if it is safe to do so. |
|  | @comments: A Quantity value can be safely interpreted as an integer if it has no decimal component (i.e. zeros after the decimal), |
|  | and it has the default UCUM unit of '1' |
|  | \*/ |
|  | define function ToInteger(value Quantity): |
|  | case |
|  | when Abs(value.value - Truncate(value.value)) > 0.00000001 then |
|  | Message(null, true, 'ToInteger.InvalidArgument', ErrorSeverity, 'The quantity has a non-zero decimal component and cannot be safely interpreted as an Integer') |
|  | when value.unit != '1' then |
|  | Message(null, true, 'ToInteger.InvalidUnit', ErrorSeverity, 'The quantity has non-default units specified and cannot be safely interpreted as an Integer') |
|  | else |
|  | Truncate(value.value) |
|  | end |
|  |  |
|  | /\* |

**A:** The number ‘1’ is the default UCUM unit as the identity unit in UCUM math. An annotation cannot be used at the current time, though that would be a good enhancement to the function.