Arden Mapping

This section provides detail on mapping a HeDS ECA artifact to an ArdenML 2.8 rule definition. The approach taken is to provide each relevant element in the HeDS schema and the corresponding element or elements in the ArdenML syntax.

|  |  |
| --- | --- |
| HeDS Element | ArdenML Element |
| metadata | ArdenLibrary |
| externalData\parameter | Argument |
| externalData\def | ReadAs |
| expressions\def | LogicStatementType |
| trigger | EvokeStatementType |
| condition | LogicStatementType |
| behavior | ? |
| group | ActionStatementType |

## Expressions

|  |  |
| --- | --- |
| Expression Type | ArdenML Type |
| ExpressionDef | Assignment |
| ExpressionRef | Identifier |
| ParameterDef | Argument |
| ParameterRef | Identifier |
| Literal | Value (w/o unit) |
| PropertyExpression | New |
| ObjectExpression | New |
| ObjectRedefine | Clone + Dot Operator for Assignment |
| Interval | Duration (only applies to Time) |
| List | List |
| Property | Identifier |

## Temporal Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| DateAdd | Add |
| DateDiff | Subtract |
| DatePart | ExtractYear, ExtractMonth, ExtractDay ExtractHour, ExtractMinute, ExtractSecond |
| Today | Midnight |
| Now | Now |
| Date | Construct as a string, use Arden’s implicit conversion |

## Logical Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| And | And (expand nary to binary) |
| Or | Or (expand nary to binary) |
| Not | Not |
| Conditional | If Statement w/ local temporary variable |
| Case | If Statement w/ Else and local temp |
| IsNull | IsNull |
| IfNull | If Statement w/ local temporary variable |
| Coalesce | If Statement w/ Else and local temp |

## Comparison Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| Equal | EQ |
| NotEqual | Not EQ |
| Less | LT |
| Greater | GT |
| LessOrEqual | LE |
| GreaterOrEqual | GE |

## String Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| Concat | Concat (nested) |
| Combine | String |
| Split | ExtractCharacters |
| Upper | Uppercase |
| Lower | Lowercase |
| Length | Length |
| Pos | FindString |
| Indexer | SubstringCharactersFrom |
| Substring | SubstringCharactersFrom |

## Arithmetic Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| Add | Add |
| Subtract | Subtract |
| Multiply | Multiply |
| Divide | Divide |
| Div | Write Equivalent |
| Mod | Write Equivalent |
| Ceiling | Ceiling |
| Floor | Floor |
| Abs | Abs |
| Negate | Minus |
| Round | Round |
| Ln | Log |
| Log | Write Equivalent using Log10 |
| Power | Power |
| Succ | +1 (Depends on type) |
| Pred | -1 (Depends on type) |

## Interval Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| Equal |  |
| NotEqual |  |
| Contains |  |
| In |  |
| ProperContains |  |
| ProperIn |  |
| Length | Length |
| Before |  |
| After |  |
| Meets |  |
| Overlaps |  |
| Union |  |
| Intersect |  |
| Difference |  |
| Begin |  |
| End |  |

If the Interval is of type date, then the Duration operations and other time-based functions in Arden can be used, otherwise, the operation would need to be mapped to its primitive expressions.

SeqTo – Generates a sequence of values within a given interval, this could be useful for performing some of these mappings. (Integer only)

## List Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| Equal | EQ |
| NotEqual |  |
| IsEmpty | No? |
| IsNotEmpty | Exist |
| Filter | Where |
| Contains |  |
| In |  |
| ProperContains |  |
| ProperIn |  |
| IndexOf | Element? |
| Sort | Sort, SortUsing |
| ForEach |  |
| Current |  |
| First | First |
| Last | Last |
| Indexer | Element? |
| Union | Merge |
| Intersect |  |
| Difference |  |

## Aggregate Operators

|  |  |
| --- | --- |
| HeDS Operator | ArdenML Operator |
| Count | Count |
| Sum | Sum |
| Min | Minimum |
| Max | Maximum |
| Avg | Average |
| AllTrue | All |
| AnyTrue | Any |

## Data Type Expressions

|  |  |
| --- | --- |
| HeDS Type | ArdenML Type |
| Address |  |
| Boolean |  |
| CodeLiteral |  |
| SimpleCode |  |
| EntityName |  |
| Identifier |  |
| Integer |  |
| IntegerInterval |  |
| PhsyicalQuantityInterval |  |
| QuantityInterval |  |
| RatioInterval |  |
| RealInterval |  |
| TimestampInterval |  |
| PhysicalQuantity | Value (w/ Unit) |
| Real |  |
| Ratio |  |
| String |  |
| Url |  |
| Timestamp |  |
| Period |  |

## Clinical Expressions

|  |  |
| --- | --- |
| HeDS Type | ArdenML Type |
| ClinicalRequest | ReadAs |
| ValueSet | Interface? |
| InValueSet | Interface? |

The following example illustrates a HeDS format clinical request for blood glucose observations in terms of the VMR model:

<def name="PatientBloodGlucoseObservations">

<expression xsi:type="ClinicalRequest"

cardinality="Multiple"

dataType="vmr:ObservationResult"

codeProperty="observationFocus.code"

dateProperty="observationEventTime"

>

<description>Retrieve patient's blood glucose level from the vMR</description>

<codes xsi:type="List">

<element xsi:type="CodeLiteral"

code="790.29"

codeSystem="2.16.840.1.113883.6.29"

codeSystemName="CPT II"

displayName="Other abnormal glucose"

/>

</codes>

<dateRange xsi:type="Interval">

<begin xsi:type="DateAdd">

<date xsi:type="Today"/>

<granularity xsi:type="Literal"

valueType="DateGranularity"

value="Month"

/>

<numberOfPeriods xsi:type="Literal"

valueType="xs:int"

value="-12"

/>

</begin>

<end xsi:type="Today"/>

</dateRange>

</expression>

</def>

The following example provides the equivalent request in ArdenML syntax:

<Object>

<ObjectIdentifier var="CD"/>

<Defined>

<Attribute var="codeSystem" otype="string"/>

<Attribute var="code" otype="string"/>

<Attribute var="codeSystemName" otype="string"/>

<Attribute var="displayName" otype="string"/>

</Defined>

</Object>

<Object>

<ObjectIdentifier var="BodySite"/>

<Defined>

<Attribute var="bodySiteCode" otype="CD"/>

<Attribute var="laterality" otype="CD"/>

</Defined>

</Object>

<Object>

<ObjectIdentifier var="ObservationResult"/>

<Defined>

<Attribute var="templateId" otype="string"/>

<Attribute var="id" otype="string"/>

<Attribute var="dataSourceType" otype="CD"/>

<Attribute var="observationFocus" otype="CD"/>

<Attribute var="observationMethod" otype="CD"/>

<Attribute var="targetBodySite" otype="BodySite"/>

<Attribute var="observationEventTime" otype="time"/>

<Attribute var="observationValue" otype="any-type"/>

<Attribute var="interpretation" otype="CD"/>

</Defined>

</Object>

<ReadAs>

<Identifier var="PatientBloodGlucoseObservations" otype="ObservationResult"/>

<Assigned>

<Mapping>

<Contents>

select \*

from ObservationResult

where observationFocus.codeSystem = ''

and observationFocus.code = ''

and observationEventTime between

DateAdd(month, -12, GetDate()) and GetDate()

</Contents>

</Mapping>

</Assigned>

</ReadAs>