$project.name

$DocumentTitle

Author: $Author

Revision: $Revisions.lastChild.name

|  |  |
| --- | --- |
| $CompanyName |  |
| $CompanyAddress | $date.get(“MMMMM dd, yyyy”) |

**Approval**

|  |
| --- |
| The original of this document is approved and signed by:  Name:  Surname:  Title:  Date:  Signature: |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Description** | **Author** |
| #forrow($rev in $Revisions)  $rev.name | $rev.date | $rev.description | $rev.author #endrow |
|  |  |  |  |

| Table of Contents |
| --- |
| Microsoft Word users please click here and press F9 to create Table of Contents.  OpenOffice.org users please remove this text and select Insert Table of Content from menu. |

| Table of Figures |
| --- |
| Microsoft Word users please click here and press F9 to create Table of Contents.  OpenOffice.org users please remove this text and select Insert Table of Content from menu. |

# **Introduction**

# **Introduction**

## **Purpose**

This CIMI Reference Model was modeled using the Archetype Modeling Language (AML) in MagicDraw, a standard UML modeling tool from No Magic, Inc. This documentation is auto-generated from three AML models (CIMI RM Core, CIMI RM Foundation, CIMI RM Clinical). The corresponding three Basic Metamodel (BMM) files also included in this distribution are also auto-generated from these models. BMM files are the formal OpenEHR serialization of the CIMI Reference Model. Thus, the AML representation of the CIMI Reference Model is the formal specification of CIMI reference model classes.

## **Model Specification**

The CIMI RM provides three layers of classes.   
  
The first layer contains the datatypes and a small number of classes necessary for the construction of more complex types.

The second layer defines the fundamental reference model patterns that align with ISO 13606 and the OpenEHR reference model.

The third layer contains the domain classes, representing all of the properties to be used in CIMI archetypes.

The archetypes work by constraining the properties defined in the reference model: *they do not add new properties*. Note that the archetypes are hand-created and not generated from the AML source *at this time*.  
  
Note that many CIMI properties have "model bindings" to SNOMED CT concept model attributes. The intent is to assert explicit semantics that can be used in description logics inference. For instance, a record of a pain with finding site of chest can be classified as equivalent to the concept of chest pain, which may have implications for treatment that are recognized by clinical decision support.  
  
In the future, model bindings will be included in the AML model. At this time, though, they are introduced in the accompanying ADL archetypes.

Please refer to the ***Architecture, Methodology and Style Guide*** for more information about the model itself.

## **Purpose**

$Purpose

## **Scope**

$Scope

## **Model Specification**

$Overview

#macro (removeLineBreak $s)

#set ($displayedName = “”)

#foreach ($c in $s.toCharArray())

#if($c.getType($c)==15)

#set($displayedName = $displayedName.concat(“ “))

#else

#set($displayedName = $displayedName.concat($c.toString()))

#end

#end

#end

#set($packages = $array.createArray())

#set($void = $array.addCollection($packages, $Package))

#set($void = $array.addCollection($packages, $SmartPackage))

#foreach ($package in $sorter.sort($packages, “qualifiedName”))

#set($qualifiedName = $package.qualifiedName)

# **Package $qualifiedName**

$package.documentation

#if ($package.hasImage())

$package.image

1. $package.name

#end

|  |  |
| --- | --- |
| **Name** | $package.name |
| **Qualified Name** | $qualifiedName |

#set ($classList = $array.createArray())

#set ($interfaceList = $array.createArray())

#set ($enumerationList = $array.createArray())

#foreach ($element in $report.getOwnedElementsIncludingAdditional($package, true))

#if ($element.elementType == “class”)

#set ($void = $classList.add($element))

#elseif ($element.elementType == “interface”)

#set ($void = $interfaceList.add($element))

#elseif ($element.elementType == “enumeration”)

#set ($void = $enumerationList.add($element))

#end

#end

#foreach ($class in $sorter.sort($classList))

## **Class $bookmark.create($class.ID, $class.name)**

$class.documentation

#if ($class.hasImage())

$class.image

1. $class.name

#end

|  |  |
| --- | --- |
| **Name** | $class.name |
| **Qualified Name** | $class.qualifiedName |
| **Visibility** | $class.visibility |
| **Abstract** | $class.isAbstract |
| **Base Classifier** | #if ($class.hasGeneralization())  #set($genList=$array.createArray())  #foreach ($general in $class.generalization)  #set($tmp=$genList.add($general.general))  #end  #foreach($gen in $sorter.sort($genList))   * $bookmark.open($gen.ID, $gen.name)   #end  #end |
| **Realized Interface** | #if ($class.hasInterfaceRealization())  #set($realList=$array.createArray())  #foreach ($realize in $class.interfaceRealization)  #set($tmp=$realList.add($realize.contract))  #end  #foreach($real in $sorter.sort($realList))   * $bookmark.open($real.ID, $real.name)   #end  #end |

#if ($class.hasOwnedAttribute())

### **Attribute Detail**

#foreach ($attr in $sorter.sort($class.ownedAttribute))

#### $attr.name

$attr.documentation

|  |  |
| --- | --- |
| **Type** | $attr.type.name |
| **Default Value** | #if($attr.defaultValue && $attr.defaultValue.hasText())  $attr.defaultValue.text  #end |
| **Visibility** | $attr.visibility |
| **Multiplicity** | $attr.multiplicity |

#end

#end

#if ($class.hasOwnedOperation())

### **Operation Detail**

#foreach ($ope in $sorter.sort($class.ownedOperation))

#### $ope.name

$ope.documentation

|  |  |
| --- | --- |
| **Type** | $ope.type.name |
| **Visibility** | $ope.visibility |
| **Is Abstract** | $ope.isAbstract |
| **Parameter** | #set ($paraList = $array.createArray())  #foreach($para in $sorter.sort($ope.ownedParameter))  #if($para.direction!=”return”)  #set ($void = $paraList.add($para))  #end  #end  #foreach($para in $sorter.sort($paraList))  #set($paraName = “$para.direction $para.name : $para.type.name”)  #if($para.multiplicity!=””)  #set($paraName = “$paraName [$para.multiplicity]”)  #end  #if($para.defaultValue && $para.defaultValue.hasText())  #set($paraName = $para.defaultValue.text)  #end   * $paraName   #end |

#end

#end

#set($relationList = $report.getRelationship($class))

$group.init()

#foreach ($relation in $relationList)

#set($void = $group.put($relation.humanType, $relation))

#end

#if (!$relationList.isEmpty())

### **Relation Detail**

#foreach ($key in $sorter.sort($group.groupNames()))

#### $key

#foreach($rel in $sorter.sort($group.get($key)))

|  |  |
| --- | --- |
| **Name** | #if($rel.hasName())  $rel.name#end |
| **Related Element** | #foreach ($re in $sorter.sort($rel.relatedElement))  #if($re != $class)   * $re.name   #end  #end |

#end

#end

#end

#end

#foreach ($interface in $sorter.sort($interfaceList))

## **Interface $bookmark.create($interface.ID, $interface.name)**

$interface.documentation

#if ($interface.hasImage())

$interface.image

1. $interface.name

#end

|  |  |
| --- | --- |
| **Name** | $interface.name |
| **Qualified Name** | $interface.qualifiedName |
| **Visibility** | $interface.visibility |
| **Base Classifier** | #if ($interface.hasGeneralization())  #set($genList=$array.createArray())  #foreach ($general in $interface.generalization)  #set($tmp=$genList.add($general.general))  #end  #foreach($gen in $sorter.sort($genList))   * $bookmark.open($gen.ID, $gen.name)   #end  #end |

#if ($interface.hasOwnedAttribute())

### **Attribute Detail**

#foreach ($attr in $sorter.sort($interface.ownedAttribute))

#### $attr.name

$attr.documentation

|  |  |
| --- | --- |
| **Type** | $attr.type.name |
| **Default Value** | #if($attr.defaultValue && $attr.defaultValue.hasText())  $attr.defaultValue.text  #end |
| **Visibility** | $attr.visibility |
| **Multiplicity** | $attr.multiplicity |

#end

#end

#if ($interface.hasOwnedOperation())

### **Operation Detail**

#foreach ($ope in $sorter.sort($interface.ownedOperation))

#### $ope.name

**Description**

$ope.documentation

|  |  |
| --- | --- |
| **Type** | $ope.type.name |
| **Visibility** | $ope.visibility |
| **Is Abstract** | $ope.isAbstract |
| **Parameter** | #set ($paraList = $array.createArray())  #foreach($para in $sorter.sort($ope.ownedParameter))  #if($para.direction!=”return”)  #set ($void = $paraList.add($para))  #end  #end  #foreach($para in $sorter.sort($paraList))  #set($paraName = “$para.direction $para.name : $para.type.name”)  #if($para.multiplicity!=””)  #set($paraName = “$paraName [$para.multiplicity]”)  #end  #if($para.defaultValue && $para.defaultValue.hasText())  #set($paraName = $paraName.concat(“=”).concat($para.defaultValue.text))  #end   * $paraName   #end |

#end

#end

#set($relationList = $report.getRelationship($interface))

$group.init()

#foreach ($relation in $relationList)$group.put($relation.humanType, $relation)#end

#if (!$relationList.isEmpty())

### **Relation Detail**

#foreach ($key in $sorter.sort($group.groupNames()))

#### $key

#foreach($rel in $sorter.sort($group.get($key)))

|  |  |
| --- | --- |
| **Name** | #if($rel.hasName())  $rel.name#end |
| **Related Element** | #foreach ($re in $sorter.sort($rel.relatedElement))  #if($re != $interface)   * $re.name   #end  #end |

#end

#end

#end

#end

#foreach ($enum in $sorter.sort($enumerationList))

## **Enumeration $bookmark.create($enum.ID, $enum.name)**

$enum.documentation

#if ($enum.hasImage())

$enum.image

1. $enum.name

#end

|  |  |
| --- | --- |
| **Name** | $enum.name |
| **Qualified Name** | $enum.qualifiedName |
| **Visibility** | $enum.visibility |
| **Base Classifier** | #foreach ($classifier in $sorter.sort($enum.baseClassifier))   * $bookmark.open($classifier.ID, $classifier.name)   #end |

#set($relationList = $report.getRelationship($enum))

$group.init()

#foreach ($relation in $relationList)$group.put($relation.humanType, $relation)#end

#if (!$relationList.isEmpty())

### **Relation Detail**

#foreach ($key in $sorter.sort($group.groupNames()))

#### $key

#foreach($rel in $sorter.sort($group.get($key)))

|  |  |
| --- | --- |
| **Name** | #if($rel.hasName())  $rel.name#end |
| **Related Element** | #foreach ($re in $sorter.sort($rel.relatedElement))  #if($re != $enum)   * $re.name   #end  #end |

#end

#end

#end

#end

#end

# **Appendix A: Diagram**

#foreach ($diagram in $sorter.sort($Diagram))

#if ($diagram.diagramType == “Class Diagram”)

#removeLineBreak($diagram.name)

## **$displayedName**

$diagram.documentation

$diagram.image

1. $diagram.name

#end

#end