# Notes on OpenCDS Internal Data Structure

## Purpose

The OpenCDS internal data structure is roughly based on the HL7 September 2011 ballot logical model for the Virtual Medical Record (vMR). This logical model defines all the data elements in rigorous detail, and those element definitions are carefully adhered to in the internal data structure..

The internal data structure is more closely aligned with the implementable schema that is provided as an informative addendum to the logical model. This implementable schema provides two methods to submit data relationships:

1. nested (as is typical in XML structures), and
2. listed with related ids (as is typical in relational database structures).

It also moves the location where nested structures are attached to the parent class out to the leaf nodes, so that nested data in a clinical statement comes at the end of the enclosing data element. Listed data relationships are placed at the end of all other data for an “Evaluated Person”.

The internal data structure is not an exact implementation of that implementable external schema, either, because it tries to provide a single list structure for data that makes it easy for rule writers to use. This means that all nested data elements are mapped into the lists and merged with any data that was provided in list format.

Note that it is up to the user of OpenCDS not to duplicate data between nested an list structure. Both structures may be used, but an individual element must only be provided in one structure or the other.

## The Internal Structure

### Datatypes

#### Internal datatypes start as a clone

Internal datatypes start as a clone of the datatypes used in the logical model and the implementable schema. Some changes are made to all of the datatype classes to turn them into Java Beans useable in Drools and OpenCDS:

All of the generated comments from JaxB are removed (except for the description of the class), along with the imported JaxB classes, and all methods of the generated code.

Using Eclipse, we add the following:

* Getters and setters
* Hash code and equals methods
* toString() method (including the super class, where one exists)
  + (NOTE that this may require modifications to the generated toString() method. For example:

return "AdverseEventBase [adverseEventCode=" + adverseEventCode

+ ", adverseEventAgent=" + adverseEventAgent

+ ", adverseEventTime=" + adverseEventTime

+ ", documentationTime=" + documentationTime + "]";

* + needs to become:

return super.toString()

+ ", AdverseEventBase [adverseEventCode=" + adverseEventCode

+ ", adverseEventAgent=" + adverseEventAgent

+ ", adverseEventTime=" + adverseEventTime

+ ", documentationTime=" + documentationTime + "]";

* Apache License as comments at the top of the class.

#### Some internal datatypes are modified

Some internal datatypes are modified or optimized to suit the way we want to use them. Specifically, the following are changed:

* CD datatype structure is changed to change the nested CD element displayName to an attribute of of the CD. This is done for ease of use and debugging.
* II datatype structure is changed from the external root and extension structure to a single string, expected to be of the format “root^extension” which simplifies its use in rules as a unique identifier. Note that instances of II that do not have an extension are implemented as a string just consisting of “root”.
* There may be other datatypes which need similar changes.

#### Mapping Utility

There is a mappingUtility which performs mapping between internal and external datatypes. Names of mapping methods are structured as “DT2InternalDT” or “InternalDT2DT”.

There is also a JUnit class to validate this mapping.

### vMR Classes

All VMR classes defined in the external schema are expected to be present in the internal structure except for the following:

* RelatedEntity
* RelatedClinicalStatement
* RelatedEntityInRole

Data in the above three classes is normalized and moved from the nested source structure into the following corresponding internal classes in a list structure:

* EntityRelationship
* ClinicalStatementRelationship
* ClinicalStatementEntityInRoleRelationship

NOTE: The vMR classes all get the same treatment as the datatypes, but most of them will also eventually get pullIn() and pushOut() methods. See the alpha release for examples.

NOTE: All repeating elements in the JaxB generated classes (e.g., List<II> templateId ) need to have separate classes created. See the alpha release for examples, such as ClinicalStatementTemplateId for the repeating templateId values in a ClinicalStatement..

There will be a few additional Internal VMR classes defined for special purposes.

## Mapping from External Structure

### Interfaces: org.opencds.vmr.v1\_0.mappings.in

#### IMappingInbound.java

The implementation of this interface accepts the payload from the DSS message as a base64 string, and converts it into a structured JaxB element for the CDSInput. The output of this class can then be further processed by an implementation of the IBuildFactLists interface to produce the internal VMR structure for processing by an inferencing engine.

#### IBuildFactLists.java

The implementation of this interface accepts the JaxB CDSInput element as produced by the implementation of the IMappingInbound interface, and parses it into the internal VMR structure as lists of facts. This is where the heavy lifting is done to normalize the submitted data to be processed by an inferencing engine.

## Mapping to External Structure

### Interfaces: org.opencds.vmr.v1\_0.mappings.out

#### IBuildResultSet.java

The implementation of this interface accepts the processed fact lists from the inferencing engine, and produces a structured output in the CDSOutput structure. This structure is then further processed by the implementation of the IMappingOutbound interface to produce the response payload.

#### IMappingOutbound.java

The implementation of this interface accepts the CDSOutput structure produced by the implementation of the IBuildResultSet interface, and produces a base64 string for return as a response by the DSS service.