REFERENCE MODEL

The *open*EHR Support Information Model

Editors:{T Beale, S Heard}[1], {D Kalra, D Lloyd}[2]

Revision: 1.2

Pages: 49

[1] Ocean Informatics Australia

[2] Centre for Health Informatics and Multi-professional Education, University College London

Copyright Notice

Amendment Record

Issue	Details	Who	Completed
1.2	CR-000128. Update Support assumed types to ISO 11404:2003. CR-000107. Add support for exclusion and inclusion of Interval limits. CR-000116. Add PARTICIPATION.*function* vocabulary and invariant. CR-000122. Fix UML in Terminology_access classes in Support model. CR-000118. Make package names lower case. CR-000111. Move Identification Package to Support. CR-000064. Re-evaluate COMPOSITION.*is_persistent* attribute. Add “composition category” vocabulary. Re-ordered vocabularies alphabetically.	T Beale A Goodchild T Beale D Lloyd T Beale DSTC D Kalra	10 Feb 2005
1.1	CR-000047. Improve handling of codes for structural attributes. Populated Terminology and code_set codes.	S Heard	11 Mar 2004
1.0	CR-000091. Correct anomalies in use of CODE_PHRASE and DV_CODED_TEXT. Add simple terminology service interface. CR-000095. Remove *property* attribute from Quantity package. Add simple measurement interface. **Formally validated using ISE Eiffel 5.4.**	T Beale DSTC, S Heard	09 Mar 2004
0.9.9	CR-000063. ATTESTATION should have a status attribute.	D Kalra	13 Feb 2004
0.9.8	CR-000068. Correct errors in INTERVAL class.	T Beale	20 Dec 2003
0.9.7	CR-000032. Basic numeric type assumptions need to be stated CR-000041. Visually differentiate primitive types in openEHR documents. CR-000043. Move External package to Common RM and rename to Identification (incorporates CR-000036 - Add HIER_OBJECT_ID class, make OBJECT_ID class abstract.)	DSTC, D Lloyd, T Beale	09 Oct 2003
0.9.6	CR-000013. Rename key classes. Based on CEN ENV13606. CR-000038. Remove *archetype_originator* from multi-axial archetype id. CR-000039. Change archetype_id section separator from ':' to '-'.	T Beale	18 Sep 2003
0.9.5	CR-000036. Add HIER_OBJECT_ID class, make OBJECT_ID class abstract.	T Beale	16 Aug 2003
0.9.4	CR-000022. Code TERM_MAPPING.*purpose*.	G Grieve	20 Jun 2003
0.9.3	CR-000007. Added forgotten terminologies for Subject_relationships and Provider_functions.	T Beale	11 Apr 2003
0.9.2	Detailed review by Ocean, DSTC, Grahame Grieve. Updated valid characters in OBJECT_ID.*namespace*.	G Grieve	25 Mar 2003
0.9.1	Added specification for BOOLEAN type. Corrected minor error in ISO 639 standard strings - now conformant to TERMINOLOGY_ID. OBJECT_ID.*version_id* now optional. Improved document structure.	T Beale	18 Mar 2003
0.9	Initial Writing. Taken from Data types and Common Reference Models. **Formally validated using ISE Eiffel 5.2.**	T Beale	25 Feb 2003

Acknowledgements

The work reported in this paper has been funded in by a number of organisations, including The University College, London and Ocean Informatics, Australia.

Table of Contents

1 Introduction

1.1 Purpose

This document describes the *open*EHR Support Reference Model, whose semantics are used by all *open*EHR Reference Models. The intended audience includes:

- Standards bodies producing health informatics standards;
- Software development organisations developing EHR systems;
- Academic groups studying the EHR;
- The open source healthcare community;

1.2 Related Documents

Prerequisite documents for reading this document include:

- The *open*EHR Modelling Guide

1.3 Status

This document is under development, and is published as a proposal for input to standards processes and implementation works.

The latest version of this document can be found in PDF format at `http://www.openEHR.org/repositories/spec-dev/publishing/architecture/reference_model/support/REV_HIST.html`. New versions are announced on `openehr-announce@openehr.org`.

1.4 Peer review

Areas where more analysis or explanation is required are indicated with “to be continued” paragraphs like the following:

`To Be Continued:    more work required`

Reviewers are encouraged to comment on and/or advise on these paragraphs as well as the main content. Please send requests for information to `info@openEHR.org`. Feedback should preferably be provided on the mailing list `openehr-technical@openehr.org`, or by private email.

1.5 Conformance

Conformance of a data or software artifact to an *open*EHR Reference Model specification is determined by a formal test of that artifact against the relevant *open*EHR Implementation Technology Specification(s) (ITSs), such as an IDL interface or an XML-schema. Since ITSs are formal, automated derivations from the Reference Model, ITS conformance indicates RM conformance.

2 Overview

The Support Reference Model comprises types which are used throughout other *open*EHR models, but are defined elsewhere, either by standards organisations or which are accepted *de facto* standards. The package structure is illustrated in FIGURE 1.

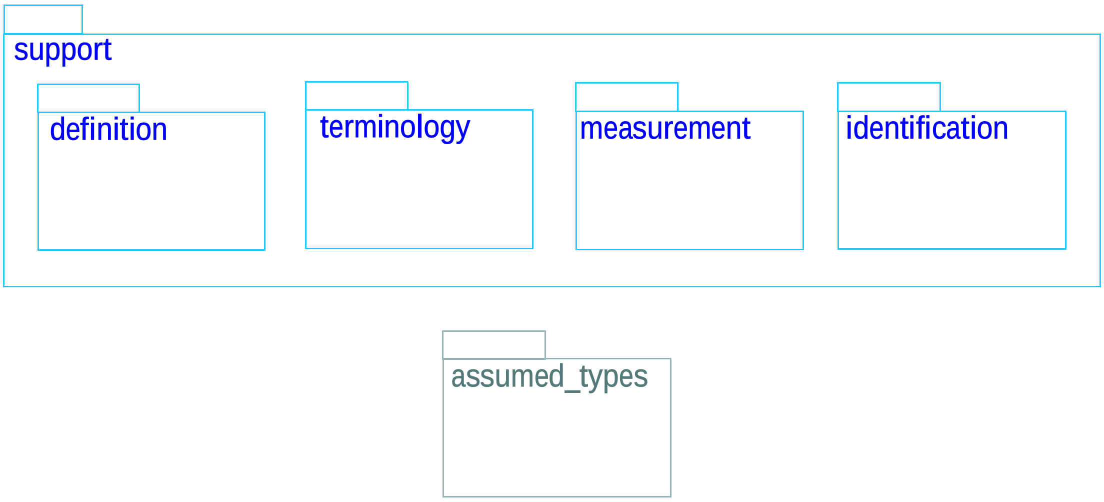

FIGURE 1 rm.support and assumed_types Packages

The four Support packages define the semantics respectively for constants, terms, scientific measurement and identifiers, which are assumed by the rest of the *open*EHR specifications.

3 Assumed Types

3.1 Overview

This section describes types assumed by all *open*EHR models. The set of types chosen here is based on a lowest common denominator set from threes sources, as follows.

- ISO 11404 (2003 revision).
- Well-known interoperability formalisms, including OMG IDL, W3C XML-schema.
- Well-known object-oriented programming languages, including C++, Java, C#, and Eiffel.

The intention in *open*EHR is to make the minimum possible assumptions about types found in implementation formalisms, while making sufficient assumptions to both enable *open*EHR models to be conveniently specified, and to allow the typical basic types of these formalisms to be used in their normal way, rather than being re-invented by *open*EHR. The ISO 11404 (2003) standard contains basic semantics of “general purpose data types” (GPDs) for information technology, and is used here as a normative basis for describing assumptions about types. The operations and properties described here are compatible with those used in ISO 11404, but not always the same, as 11404 has not chosen to use object-oriented functions. For example, the notional function `has(x:T)` (test for presence of a value in a set) defined on the type `Set<T>` below is not defined on the ISO 11404 Set type; instead, the function `IsIn(x: T; s: Set<T>)` is defined. However, in object-oriented formalisms, the function `IsIn` defined on a Set type would usually mean “subset of”, i.e. true if this set is contained inside another set. In the interests of clarity for developers, an object-oriented style of functions and properties has been used here.

Two groups of assumed types are identified: primitive types, which are those built in to a formalism’s type system, and library types, which are assumed to be available in a (class) library defined in the formalism. Thus, the type `Boolean` is always assumed to exist in a formalism, while the type `Array<T>` is assumed to be available in a library. For practical purposes, these two categories do not matter that much - whether a String is really a library class (the usual case) or an inbuilt type doesn’t make much difference to the programmer. They are shown separately here mainly as an explanatory convenience.

The assumptions that *open*EHR makes about existing types are documented below in terms of interface definitions. Each of these definitions contains *only the assumptions required for the given type to be used in the openEHR Reference Model* - **it is not by any means a complete interface definition**. The name and semantics of any function used here for an assumed type might not be identical to those found in some implementation technologies, but should be very close. Any mapping required should be stated in the relevant ITS. The definitions are compatible with the ISO 11404 standard, 2003 revision. Operation semantics are described formally using pre- and post-conditions. The keyword “Current” stands for “the current instance” (known as “this” or “self” in various languages). The keyword “like” anchors the type of the reference to the type of the object whose reference follows *like*. Not all types have definition tables - only those which add features to their inheritance parent have a table.

3.2 Date/Time Types

Date/time types deserve special mention. Although the ISO 11404 (2003) standard defines a date-and-time type generator (section 8.1.6), and a `timeinterval` type (section 10.1.6), the reality is that dates and times are provided in significantly differing ways in implementation formalisms, and as a result, *open*EHR assumes nothing at all about them. Accordingly, types for date, time, date/time and duration are defined in the *open*EHR Data Types Information Model, ensuring standardised meanings of these types within *open*EHR. ISO 8601 is used as the normative basis for both string literal representation and properties chosen within these models.

3.3 Inbuilt Primitive Types

The following types consititute the minimum built in set of types assumed by *open*EHR of an implementation formalism.

Type name in *open*EHR	Description	ISO 11404 Type
`Character`	represents a type whose value is a member of an 8-bit character-set (ISO: “repertroire”).	`Character`
`Boolean`	represents logical True/False values; usually physically represented as an integer, but need not be	`Boolean`
`Integer`	represents 32-bit integers	`Integer`
`Real`	represents 32-bit real numbers in any interoperable representation, including single-width IEEE floating point	`Real`
`Double`	type which represents 64-bit real numbers, in any interoperable representation including double-precision IEEE floating point.	`Real`

As shown in the table, *open*EHR assumes that Character is an 8-bit type. This is because the only use of Character in *open*EHR is in encapsulated data (*open*EHR Data Types), where the intention is to represent opaque data. Note that “octet” would probably be a more correct name to use here, but it generally is not used in programming languages.

FIGURE 2 illustrates the inbuilt types. Simple inheritance relationships are shown which facilitate the type descriptions below, although they are not assumed in *open*EHR (i.e. there is no assumption of types `Any` or `Ordered_Numeric`, nor of any related substitutability). This simply enables basic operations like ‘=’ to be described once for the type `Any`, rather than in every type in this section. Similarly, it is not assumed, or meant to be implied that in a real type system that there is a type called `Ordered_Numeric`, or that Integer etc inherit from it. What is assumed are that the operations defined here on Ordered_Numeric are available on the types `Integer`, `Real` and `Double` in implementation type systems, where relevant. Data-oriented implementation type systems such as XML-schema are not expected to have such operations.

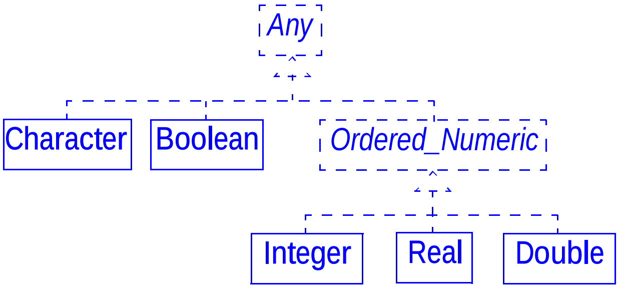

FIGURE 2 Primitive Types Assumed by *open*EHR

3.3.1 Any

INTERFACE	*Any (abstract)*	
Description	Abstract supertype. Usually maps to a type like “Any” or “Object” in an object system. Defined here to provide the value and reference equality semantics.	
Abstract	Signature	Meaning
	is_equal (other: `Any`): `Boolean`	Value equality
Functions	Signature	Meaning
	infix ‘=’ (other: `Any`): `Boolean`	Reference equality
Invariants		

3.3.2 Boolean Type

INTERFACE	Boolean	
Purpose	Boolean type used for two-valued mathematical logic.	
Abstract	Signature	Meaning
	infix "**and**" (other: `Boolean`): `Boolean` ***require*** *other_exists*: other /= void ***ensure*** *de_morgan*: Result = **not** (**not** Current **or not** other) *commutative*: Result = (other **and** Current)	Logical conjunction
	infix "**and then**" (other: `Boolean`): `Boolean` ***require*** *other_exists*: other /= void ***ensure*** *de_morgan*: Result = **not** (**not** Current **or else not** other)	Boolean semi-strict conjunction with *other*

INTERFACE	Boolean	
	infix "**or**" (other: `Boolean`): `Boolean` ***require*** *other_exists*: other /= void ***ensure*** *de_morgan*: Result = **not** (**not** Current **and not** other) *commutative*: Result = (other **or** Current) *consistent_with_semi_strict*: Result **implies** (Current **or else** other)	Boolean disjunction with *other*
	infix "**or else**" (other: `Boolean`): `Boolean` ***require*** *other_exists*: other /= void ***ensure*** *de_morgan*: Result = **not** (**not** Current **and then not** other)	Boolean semi-strict disjunction with `other'
	infix "**xor**" (other: `Boolean`): `Boolean` ***require*** *other_exists*: other /= void ***ensure*** *definition*: Result = ((Current **or** other) **and not** (Current **and** other))	Boolean exclusive or with `other'
	infix "**implies**" (other: `Boolean`): `Boolean` ***require*** *other_exists*: other /= void ***ensure*** *definition*: Result = (**not** Current **or else** other)	Boolean implication of `other' (semi-strict)
Invariants	*involutive_negation*: is_equal (**not** (**not** Current)) *non_contradiction*: **not** (Current **and** (**not** Current)) *completeness*: Current **or else** (**not** Current)	

3.3.3 Ordered_Numeric Type

INTERFACE	*Ordered_Numeric (abstract)*	
Purpose	Abstract notional parent class of ordered, numeric types, which are types which have various arithmetic and comparison operators defined. All ordered, quantified types (i.e. types with a notion of precise “magnitude”) have these operations.	
Abstract	**Signature**	**Meaning**

INTERFACE	***Ordered_Numeric (abstract)***	
	infix "*" (other: ***like*** Current): ***like*** Current ***require*** *other_exists*: other /= void ***ensure*** *result_exists*: Result /= void	Product by `other'. Actual type of result depends on arithmetic balancing rules.
	infix "+" (other: ***like*** Current): ***like*** Current ***require*** *other_exists*: other /= void ***ensure*** *result_exists*: Result /= void *commutative*: equal (Result, other + Current)	Sum with `other' (commutative). Actual type of result depends on arithmetic balancing rules.
	infix "-" (other: ***like*** Current): ***like*** Current ***require*** *other_exists*: other /= void ***ensure*** *result_exists*: Result /= void	Result of subtracting `other'. Actual type of result depends on arithmetic balancing rules.
	infix '<' (other: ***like*** Current): `Boolean`	Arithmetic comparison. In conjunction with '=', enables the definition of the operators '>', '>=', '<=', '<>'. In real type systems, this operator might be defined on another class for comparability.
Invariants		

3.4 Assumed Library Types

The types described in this section are also assumed by *open*EHR, but are expected to come from type libraries rather than be built into target formalisms.

Type name in *open*EHR	Description	ISO 11404: 2003 Type
`String`	represents unicode-enabled strings	`Character-String/ Sequence`
`Array<T>`	physical container of items indexed by number	`Array`
`List<T>`	container of items, implied order, non-unique membership	`Sequence`
`Set<T>`	container of items, no order, unique membership	`Set`
`Bag<T>`	container of items, no order, non-unique membership	`Bag`

Type name in *open*EHR	Description	ISO 11404: 2003 Type
`Hash<T,K>`	a table of values of any type, keyed by values of any basic comparable type, typically String or Integer.	`Table`
`Interval<T>`	Intervals	

FIGURE 3 illustrates the assumed library types. As with the assumed primitive types, inheritance and abstract classes are used for convenience of the definitions below, but are not formally assumed in *open*EHR.

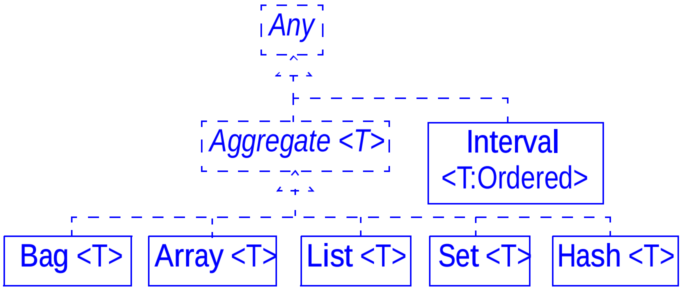

FIGURE 3 LibraryTypes Assumed by *open*EHR

3.4.1 String

INTERFACE	String	
Description	Strings of characters, as used to represent textual data in any natural or formal language.	
Functions	**Signature**	**Meaning**
	infix '+' (other: `String`): `String`	Concatenation operator - causes 'other' to be appended to this string
	is_empty: `Boolean`	True if string is empty, i.e. equal to "".
Invariants		

3.4.1.1 UNICODE

It is assumed in the *open*EHR specifications that Unicode is supported by the type `String`. Unicode is needed for all Asian, Arabic and other script languages, for both data values (particularly plain text and coded text) and for many predefined string attributes of the classes in the *open*EHR Reference Model. It encompasses all existing character sets.

3.4.2 Aggregate <T>

INTERFACE	*Aggregate <T> (abstract)*	
Description	Abstract parent of of the aggregate types List<T>, Set<T>, Bag<T>, Array<T> and Hash<T,K>.	
Functions	Signature	Meaning
	has (v: T): Boolean	Test for membership of a value
	count: Integer	Number of items in container
Invariants		

3.4.3 Table <T, K>

INTERFACE	Table <T, K: Comparable>	
Description	Type representing a keyed table of values. T is the value type, and K the type of the keys.	
Functions	Signature	Meaning
	has_key (a_key: K): Boolean	Test for membership of a key
	item (a_key: K): T	Return item for key 'a_key'. Equivalent to ISO 11404 fetch operation.
Invariants		

3.4.4 Interval <T> Type

INTERFACE	Interval <T:Ordered>	
Purpose	Interval of ordered items.	
Attributes	**Signature**	**Meaning**
	lower: `T`	lower bound
	upper: `T`	upper bound
	lower_unbounded: `Boolean`	lower boundary open (i.e. = -infinity)
	upper_unbounded: `Boolean`	upper boundary open (i.e. = +infinity)
	lower_included: `Boolean`	lower boundary value included in range if not *lower_unbounded*
	upper_included: `Boolean`	upper boundary value included in range if not *upper_unbounded*
Functions	**Signature**	**Meaning**
	has(e:`T`**)**: `Boolean`	True if (lower_unbounded **or** ((lower_included **and** v >= lower) **or** v > lower)) **and** (upper_unbounded **or** ((upper_included **and** v <= upper **or** v < upper)))
Invariants	***Limits_consistent***: (***not*** upper_unbounded ***and not*** lower_unbounded) ***implies*** lower <= upper ***Limits_comparable***: (***not*** upper_unbounded ***and not*** lower_unbounded) ***implies*** lower.strictly_comparable_to(upper)	

4 Identification Package

4.1 Overview

The `identification` package describes a model of references and identifiers for information entities only and is illustrated in FIGURE 4. Real-world entity identifiers are defined in the *open*EHR Data Types information model.

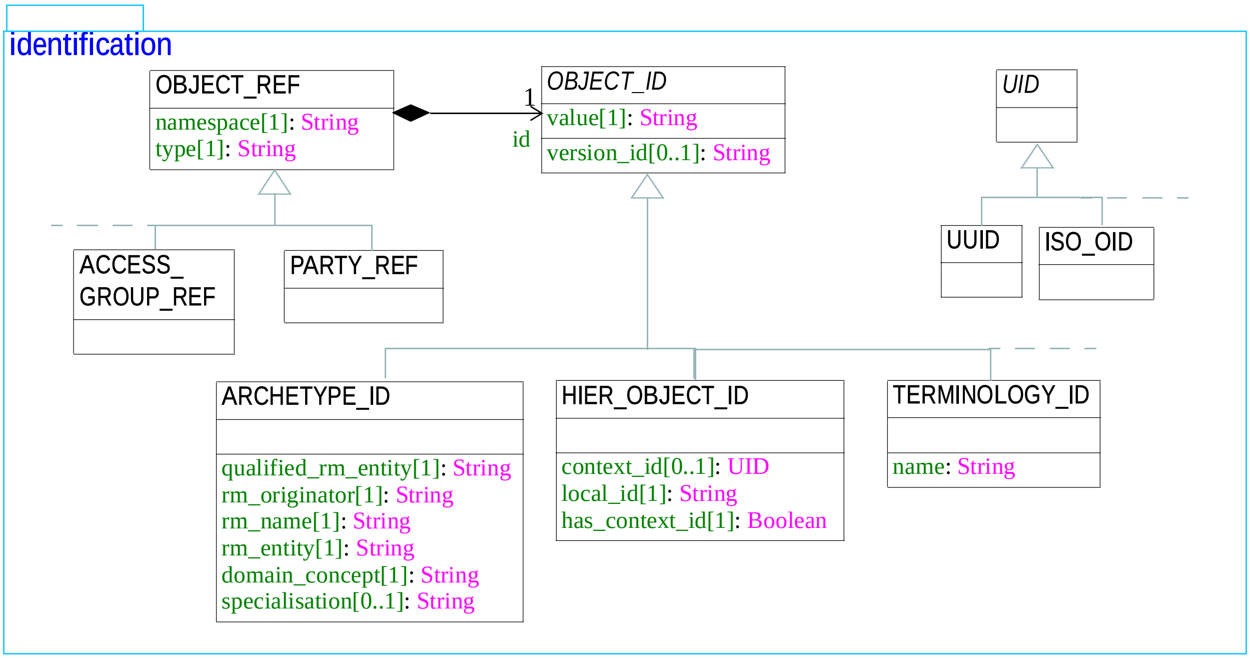

FIGURE 4 rm.common.identification Package

4.1.1 Requirements

Identification of entities both in the real world and in information systems is a non-trivial problem. The scenarios for identification across systems in a health information environment include the following:

- real world identifiers such as social security numbers, veterans affairs ids etc can be recorded as required by health care facilities, enterprise policies, or legislation.
- identifiers for informational entities which represent real world entities or processes should be unique.
- it should be possible to determine if two identifiers refer to information entities which are linked to the same real world entity, even if instances of the information entities are maintained in different systems;
- versions or changes to real-world entity-linked informational entities (which may create new information instances) should be accounted for in two ways:
  - it should be possible to tell if two identifiers refer to distinct versions of the same informational entity in the same version tree;
  - it should not be possible to confuse same-named versions of informational entities maintained in multiple systems which purport to represent the same real world entity. E.g. there is no guarantee that two systems' "latest" version of the Person "Dr Jones" is the same.

Medico-legal use of information relies on previous states of information being identifiable in some way.

- it should be possible for an entity in one system or service (such as the EHR) to refer to an entity in another system or service in such a way that:
    - the target of the reference is easily finable within the shared environment, and
    - the reference does is valid regardless of the physical architecture of servers and applications.

The following subsections describe some of the features and challenges of identification.

Identification of Real World Entities (RWEs)

Real world entities such as people, car engines, invoices, and appointments all have identifiers. Although many of these are designed to be unique within a jurisdiction, they are often not, due to data entry errors, bad design (ids which are too small or incorporate some non-unique characteristic of the identified entities), bad process (e.g. non-synchronised id issuing points); identity theft (e.g. via theft of documents of proof or hacking). In general, while some real world identifiers (RWIs) are "nearly unique", none can be guaranteed so. It should also be the case that if two RWE identifiers are equal, they refer to the same RWE.

Identification of Informational Entities (IEs)

As soon as information systems are used to record facts about RWEs, the situation becomes more complex because of the intangible nature of information. In particular:

- the same RWE can be represented simultaneously on more than one system ("spatial multiplicity");
- the same RWE may be represented by more than one "version" of the same IE in a system ("temporal multiplicity").

At first sight, it appears that there can also be purely informational entities, i.e. IEs which do not refer to any RWE, such as books, online-only documents and software. However, as soon as one considers an example it becomes clear that there is always a notional "definitive" or "authoritative" (i.e. trusted) version of every such entity. These entities can better be understood as "virtual RWEs". Thus it can still be said that multiple IEs may refer to any given RWE.

The underlying reason for the multiplicity of IEs is that "reality" - time and space - in computer systems is not continuous but discrete, and each "entity" is in fact just a snapshot of certain attribute values of a RWE.

If identifiers are assigned to IEs without regard to versions or duplicates, then no assertion can be made about the identified RWE when two IE ids are compared.

Referencing of Informational Entities

Within a distributed information environment, there is a need for entities not connected by direct references in the same memory space to be able to refer to each other. There are two competing requirements:

- that the separation of objects in a distributed computing environment not compromise the semantics of the model. At the limit, this mandates the use of proxy types which have the same abstract interface as the proxied type; i.e. the "static" approach of Corba.
- that different types of information can be managed relatively independently; for example EHR and demographic information can be managed by different groups in an organisation

or community, each with at least some freedom to change implementation and model details.

4.1.2 Design

The class OBJECT_ID is an abstract model of identifiers of IEs. It is assumed *a priori* that there can in general be more than one IE referring to the same underlying real world entity (RWE), such as a person or invoice; this is due to the possible existence of multiple copies, and also multiple versions. An OBJECT_ID therefore explicitly includes an optional *version_id* attribute. The rule for versioning is that if any attribute value of the IE changes, the version attribute value should be updated, e.g. by incrementing a simple integer. The *version_id* attribute should be used for object identifiers whose targets change, such as demographic entities; it can usually be omitted for ids of things like terminology codes, where the terminology obeys the rule that a given code never changes its meaning through all versions of the terminology (i.e. ICD10 code F40.0 will mean "Agoraphobia" for all time (in English)).

The subtype HIER_OBJECT_ID defines a hierarchical identifier model, along the lines of ISO Oids; it includes the attributes *context_id* and *local_id*, to make up a complete, unique identifier. The *context_id* is optional, since it is possible for *local_id* values to exist in a single global namespace. When a HIER_OBJECT_ID has a *context_id*, it is of type UID, meaning it has the properties of a timeless unique object identifier. Subtypes of UID include the ISO_OID and DCE UUID types.

The other subtypes, ARCHETYPE_ID and TERMINOLOGY_ID define different kinds of identifier, the former being a multi-axial identifier for archetypes, and the latter being a globally unique single string identifier for terminologies.

All OBJECT_IDs are used as identifier attributes within the thing they identify, in the same way as a database primary key. To *refer* to an identified object, an instance of the class OBJECT_REF is required, in the same way as a database foreign key. OBJECT_REF is provided as a means of distributed referencing, and includes the object namespace (typically 1:1 with some service, such as "terminology") and type. The general principle of object references is to be able to refer to an object available in a particular namespace or service. Usually they are used to refer to objects in other services, such as a demographic entity from within an EHR, but they may be used to refer to local objects as well. The type may be the concrete type of the referred-to object (e.g. "GP") or any proper ancestor (e.g. "PARTY"). The notion of object reference provided here is a compromise between the static binding notion of Corba (where each model is dependent on all the interface details of the classes in other models) and a purely dynamic referencing scheme, where the holder of a reference cannot even tell what type of object the reference points to.

4.2 Class Descriptions

4.2.1 OBJECT_REF Class

CLASS	OBJECT_REF
Purpose	Class describing a reference to another object, which may exist locally or be maintained outside the current namespace, e.g. in another service. Services are usually external, e.g. available in a LAN (including on the same host) or the internet via Corba, SOAP, or some other distributed protocol. However, in small systems they may be part of the same executable as the data containing the Id.

CLASS	OBJECT_REF	
Attributes	**Signature**	**Meaning**
	id: `OBJECT_ID`	Globally unique id of an object, regardless of where it is stored.
	namespace: `String`	Namespace to which this identifier belongs in the local system context (and possibly in any other *open*EHR compliant environment) e.g. “terminology”, “demographic”. These names are not yet standardised. Legal values for the namespace are `“local” | “unknown” | “[a-zA-Z][a-zA-Z0-9_-:/&+?]*”`
	type: `String`	Name of the class of object to which this identifier type refers, e.g. “PARTY”, “PERSON”, “GUIDELINE” etc. These class names are from the relevant reference model. The type name “ANY” can be used to indicate that any type is accepted (e.g. if the type is unknown).
Invariant	***Id_exists***: id /= Void ***Namespace_exists***: namespace /= Void ***and then not*** namespace.empty ***Type_exists***: type /= Void ***and then not*** type.empty	

4.2.2 ACCESS_GROUP_REF Class

CLASS	ACCESS_GROUP_REF	
Purpose	Reference to access group in an access control service.	
Inherit	`OBJECT_REF`	
Functions	**Signature**	**Meaning**
Invariant	***namespace_validity***: namespace.is_equal(“access_control”) ***type_validity***: type.is_equal(“ACCESS_GROUP”)	

4.2.3 PARTY_REF Class

CLASS	PARTY_REF
Purpose	Identifier for parties in a demographic service. There are typically a number of subtypes of the `PARTY` class, including `PERSON`, `ORGANISATION`, etc.
Inherit	`OBJECT_REF`

CLASS	PARTY_REF	
Functions	Signature	Meaning
Invariant	***namespace_validity***: namespace.is_equal("demographic")	

4.2.4 OBJECT_ID Class

CLASS	*OBJECT_ID (abstract)*	
Purpose	Ancestor class of identifiers of informational objects. Ids may be completely meaningless, in which case their only job is to refer to something, or may carry some information to do with the identified object.	
Use	Object_ids are used inside an object to identify that object. To identify another object, use an Object_ref.	
Attributes	Signature	Meaning
	value: `String`	The value of the id in the form defined below.
Functions	Signature	Meaning
	version_id: `String` ***ensure*** Result /= Void ***implies not*** Result.is_empty	Version of information pointed to by this ID, if versioning is supported.
Invariant	***Value_exists***: value /= Void ***and then not*** value.empty	

4.2.5 HIER_OBJECT_ID Class

CLASS	HIER_OBJECT_ID	
Purpose	Hierarchical identifiers.	
HL7	The HL7v3 II Data type.	
Functions	Signature	Meaning
	context_id: `UID`	The identifier of the conceptual namespace in which the object exists, within the identification scheme. May be Void.
	has_context_id: `Boolean`	True if there is at least one "." in identifier before version part.

CLASS	HIER_OBJECT_ID	
	local_id: `String` ***ensure*** Result /= Void ***and then not*** Result.empty	The local identifier of the object within the context.
Invariant		

4.2.5.1 Syntax

The syntax of the *value* attribute by default follows the following pattern:

```
[ context_id “.” ] local_id [ “(” version_id “)” ]
```

The syntax may be redefined in subtypes.

4.2.6 ARCHETYPE_ID Class

CLASS	ARCHETYPE_ID	
Purpose	Identifier for archetypes.	
Inherit	`OBJECT_ID`	
Functions	**Signature**	**Meaning**
	qualified_rm_entity: `String`	Globally qualified reference model entity, e.g. `“openehr-ehr_rm-entry”`.
	domain_concept: `String`	Name of the concept represented by this archetype, including specialisation, e.g. `“biochemistry result-cholesterol”`.
	rm_originator: `String` ***ensure*** Result /= Void ***and then not*** Result.is_empty	Organisation originating the reference model on which this archetype is based, e.g. `“openehr”`, `“cen”`, `“hl7”`.
	rm_name: `String` ***ensure*** Result /= Void ***and then not*** Result.is_empty	Name of the reference model, e.g. `“rim”`, `“ehr_rm”`, `“en13606”`.
	rm_entity: `String` ***ensure*** Result /= Void ***and then not*** Result.is_empty	Name of the ontological level within the reference model to which this archetype is targeted, e.g. for openEHR, “folder”, “composition”, “section”, “entry”.

CLASS	ARCHETYPE_ID	
	specialisation: `String` ***ensure*** Result /= Void ***implies not*** Result.is_empty	Name of specialisation of concept, if this archetype is a specialisation of another archetype, e.g. “cholesterol”.
	local_id: `String` ***ensure then*** Result.is_equal(value)	
Invariant	***Qualified_rm_entity_valid***: qualified_rm_entity /= Void ***and then not*** qualified_rm_entity.is_empty ***Domain_concept_valid***: domain_concept /= Void ***and then not*** domain_concept.is_empty	

4.2.6.1 Archetype ID Syntax

Archetype ids obey the general pattern of object ids. They are defined in a single global namespace, hence the *context_id* attribute is always empty. The remaining part of the id is “multi-axial”, meaning that each identifier instance denotes a single archetype within a multi-dimensional space. In this case, the space is essentially a versioned 3-dimensional space, with the dimensions being:

- reference model entity, i.e. target of archetype
- domain concept
- version

As with any multi-axial identifier, the underlying principle of an archetype id is that all parts of the id must be able to be considered immutable. This means that no variable characteristic of an archetype (e.g. accrediting authority, which might change due to later accreditation by another authority, or may be multiple) can be included in its identifier. The syntax of an `ARCHETYPE_ID` is as follows:

```
archetype_id: qualified_rm_entity ‘.’ domain_concept ‘.’ version_id

qualified_rm_entity: rm_originator ‘-’ rm_name ‘-’ rm_entity
rm_originator: NAME
rm_name: NAME
rm_entity: NAME

domain_concept: concept_name { ‘-’ specialisation }*
concept_name: NAME
specialisation: NAME

version_id: ‘v’ NUMBER

NUMBER: [0-9]*
NAME: [a-z][a-z0-9()/%$#&]*
```

The field meanings are as follows:

rm_originator: id of organisation originating the reference model on which this archetype is based;

rm_name: id of the reference model on which the archetype is based;

rm_entity: ontological level in the reference model;

domain_concept: the domain concept name, including any specialisations;

version_id: numeric version identifier;

Examples of archetype identifiers include:

- `openehr-ehr_rm-section.physical_examination.v2`
- `openehr-ehr_rm-section.physical_examination-prenatal.v1`
- `hl7-rim-act.progress_note.v1`
- `openehr-ehr_rm-entry.progress_note-naturopathy.v2`

Archetypes can also be identified by other means, such as ISO oids.

4.2.7 TERMINOLOGY_ID Class

CLASS	TERMINOLOGY_ID	
Purpose	Identifier for terminologies such accessed via a terminology query service. In this class, the value attribute identifies the Terminology in the terminology service, e.g. "SNOMED-CT". A terminology is assumed to be in a particular language, which must be explicitly specified. The value if the id attribute is the precise terminology id identifier, including actual release (i.e. actual "version"), local modifications etc; e.g. "ICPC2"	
Inherit	`OBJECT_ID`	
Functions	Signature	Meaning
	name: `String` ***ensure*** Result /= Void ***and then not*** Result.empty	Return the terminology id (which includes the "version" in some cases). Distinct names correspond to distinct (i.e. non-compatible) terminologies. Thus the names "ICD10AM" and "ICD10" refer to distinct terminologies.
	as_string: `String` ***ensure*** Result = key	
	as_canonical_string: `String`	Result = "<key>" + value + "</key>"
Invariants		

4.2.7.1 Identifier Syntax

The syntax of the *value* attribute is as follows:

```
name [ “(” version “)” ]
```

Examples of terminology identifiers include:

- `“snomed-ct”`
- `“ICD9(1999)”`

Versions should only be needed for those terminologies which break the rule that the thing being identified with a code loses or changes its meaning over versions of the terminology. This should not be the case for well known modern terminologies and ontologies, particularly those designed since the publication of Cimino's 'desiderata' [1] of which the principle of "concept permanance" is applicable here - "A concept's meaning cannot change and it cannot be deleted from the vocabulary". However, there maybe older terminologies, or specialised terminologies which may not have obeyed these rules, but which are still used; version ids should always be used for these.

4.2.8 UID Class

CLASS	***UID (abstract)***	
Purpose	Anstract parent of classes representing unique identifiers which identify information entities in a durable way. UIDs only ever identify one IE in time or space and are never re-used.	
HL7	The HL7v3 UID Data type.	
Attributes	**Signature**	**Meaning**
	value: `String`	The value of the id.
Invariant	***Value_exists***: value /= Void ***and then not*** value.empty	

4.2.9 ISO_OID Class

CLASS	**ISO_OID**	
Purpose	Model of ISO's Object Identifier (oid) as defined by the standard ISO/IEC 8824 . Oids are formed from integers separated by dots. Each non-leaf node in an Oid starting from the left corresponds to an assigning authority, and identifies that authority's namespace, inside which the remaining part of the identifier is locally unique.	
HL7	The HL7v3 OID Data type.	
Inherit	`UID`	
Functions	**Signature**	**Meaning**
Invariant		

4.2.10 UUID Class

CLASS	**UUID**
Purpose	Model of the DCE Universal Unique Identifier or UUID which takes the form of hexadecimal integers separated by hyphens, following the pattern 8-4-4-4-12 as defined by the Open Group, CDE 1.1 Remote Procedure Call specification, Appendix A.

CLASS	UUID	
HL7	The HL7v3 UUID Data type.	
Inherit	UID	
Functions	Signature	Meaning
Invariant		

5 Terminology Package

5.1 Overview

This section describes the *open*EHR terminology and code sets which provide values for the dozen or so structural attributes in the *open*EHR Reference Model, along with a simple way of accessing them. There are two types of coded terms used. The first are ‘proper’ coded terms, where each code is a concept identifier, for which there can be a rubric and description in multiple languages. In other words, they way of ‘saying’ the concept is dependent on the language one is working in. Most clinical terminologies are in this category, e.g. ICD10, ICPC. Terminologies in this category are modelled in *open*EHR by the `TERMINOLOGY` class, and by terms expressed as instances the `DV_CODED_TEXT` class, each of which has as an attribute a defining `CODE_PHRASE` - the actual code.

The second category is codes which are self-defining, and which do not have separate rubrics. The ISO country and language codes are examples of this, as are code groups for such concepts as ‘integrity check algorithm names’. This category is modelled in *open*EHR by the `CODE_SET` which is made up of `CODE_PHRASE`s.

The `TERMINOLOGY` and `CODE_SET` classes are defined below in a simple terminology interface, while the `DV_CODED_TEXT` and `CODE_PHRASE` types are defined in the *open*EHR Data Types Information Model.

Both code set definitions and terminology groups provide mappings to other recognised terminologies or vocabularies. Given that the attributes defined here are mostly structural attributes (i.e. predefined in the *open*EHR Reference Model), mappings tend to be to terms in vocabularies defined by standards organisations such as CEN and HL7, rather than large clinical vocabularies such as ICD10 (WHO). *Open*EHR does not specify the use of these vocabularies.

5.2 Service Interface

A simple terminology service interface is defined according to FIGURE 5, enabling *open*EHR terms to be referenced formally from within the Reference Model.

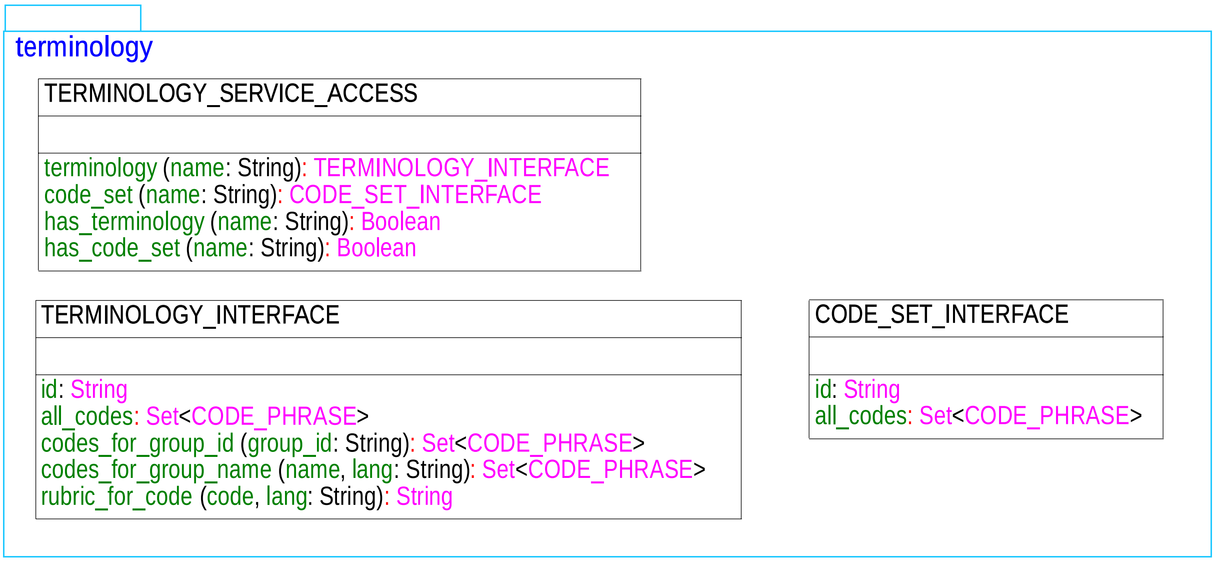
terminology
TERMINOLOGY_SERVICE_ACCESS
terminology (name: String): TERMINOLOGY_INTERFACE
code_set (name: String): CODE_SET_INTERFACE
has_terminology (name: String): Boolean
has_code_set (name: String): Boolean
TERMINOLOGY_INTERFACE
id: String
all_codes: Set<CODE_PHRASE>
codes_for_group_id (group_id: String): Set<CODE_PHRASE>
codes_for_group_name (name, lang: String): Set<CODE_PHRASE>
rubric_for_code (code, lang: String): String
CODE_SET_INTERFACE
id: String
all_codes: Set<CODE_PHRASE>

FIGURE 5 rm.support.terminology Package

Structural attributes in the Reference Model, such as FEEDER_AUDIT.*change_type* are defined by an invariant in the enclosing class, such as the following:

> ***Change_type_valid***: terminology("openehr").codes_for_group_name("audit change type", "en").has(change_type.defining_code)

This is a formal way of saying that the attribute *change_type* must have a value such that its *defining_code* (its CODE_PHRASE) is in the set of CODE_PHRASEs in the *open*EHR Terminology which are in the group called (in english) "audit change type".

A similar invariant is used for attributes of type CODE_PHRASE, which come from a code_set:

> ***Media_type_terminology***: media_type /= Void ***and then***
> code_set("media types").all_codes.has(media_type)

5.2.1 Class Definitions

5.2.1.1 TERMINOLOGY_SERVICE_ACCESS Class

CLASS	TERMINOLOGY_SERVICE_ACCESS	
Purpose	Defines an object providing proxy access to a terminology service.	
Functions	Signature	Meaning
	terminology (name: String): TERMINOLOGY_INTERFACE ***require*** name /= Void ***and then*** has_terminology (name: String) ***ensure*** Result /= Void	Return an interface to the terminology named 'name'
	code_set (name: String): CODE_SET_INTERFACE ***require*** name /= Void ***and then*** has_code_set (name: String) ***ensure*** Result /= Void	Return an interface to the code_set named 'name'
	has_terminology (name: String): Boolean ***require*** name /= Void ***and then*** not name.is_empty	True if terminology named 'name' known by this service.
	has_code_set (name: String): Boolean ***require*** name /= Void ***and then*** not name.is_empty	True if code_set named 'name' known by this service.

CLASS	TERMINOLOGY_SERVICE_ACCESS
Invariants	

5.2.1.2 TERMINOLOGY_INTERFACE Class

CLASS	TERMINOLOGY_INTERFACE	
Purpose	Defines an object providing proxy access to a terminology.	
Functions	**Signature**	**Meaning**
	id: `String`	Identification of this Terminology
	all_codes: `Set<CODE_PHRASE>`	Return all codes known in this terminology
	codes_for_group_id (group_id: `String`): `Set<CODE_PHRASE>`	Return all codes under grouper 'group_id' from this terminology
	codes_for_group_name (name, lang: **String**): `Set<CODE_PHRASE>`	Return all codes under grouper whose name in 'lang' is 'name' from this terminology
	rubric_for_code (code, lang: `String`): `String`	Return all rubric of code 'code' in language 'lang'.
Invariants	***id_exists***: id /= Void ***and then not*** id.is_empty	

5.2.1.3 CODE_SET_INTERFACE Class

CLASS	CODE_SET_INTERFACE	
Purpose	Defines an object providing proxy access to a code_set.	
Functions	**Signature**	**Meaning**
	id: `String`	Identification of this Terminology
	all_codes: `Set<CODE_PHRASE>`	Return all codes known in this terminology
Invariants	***id_exists***: id /= Void ***and then not*** id.is_empty	

5.3 Code Sets

Code sets are not shown in full here, since their codes are derived from resources published by outside authorities; however, the *open*EHR code-set databases contain the full set of codes in each case.

5.3.1 Languages

This ISO code set defined by the ISO 639 standard consists of the "alpha-2" form of names of languages. This does not cover all languages, whereas ISO 639 "alpha-3" covers many more languages of cultural or indigenous interest, but which nevertheless are unlikely to be supported by current software or operating systems. See http://www.loc.gov/standards/iso639-2/langhome.html.

Issuer: *ISO* Code set name: *"languages"*		
Code	**Description**	**Mappings**
"ab"	"Abkhazian"	
...	...	
"bg"	"Bulgarian"	
...	...	
"zh"	"Chinese"	
...	...	

5.3.2 Countries

This ISO code set defined by the ISO 3166 standard consists of 2-character names of countries and country subdivisions. For a definitive online rendition see http://www.unicode.org/unicode/onlinedat/countries.html.

Issuer: *ISO* Code set name: *"countries"*		
Code	**Description**	**Mappings**
"af"	"Afghanistan"	
"al"	"Albania"	
...	...	

5.3.3 Character Sets

This IANA (Internet Naming Authority) code set consists of the names of recognised character sets. See http://www.iana.org/assignments/character-sets for authoritative source.

Issuer: *IANA* Code set name: *"character sets"*		
Code	**Description**	**Mappings**
ISO-10646-UTF-1		
...		
ISO_8859-3:1988		
...		

5.3.4 Media Types

This IANA (Internet Naming Authority) code set consists of the names of MIME media types. See http://www.iana.org/assignments/media-types/text/ for authoritative source.

Issuer: *IANA* Code set name: *"media types"*		
Code	**Description**	**Mappings**
"text/plain"	Plain text encoded according to RFC3676	HL7_MediaType::14826
"text/html"	HTML text encoded according to RFC2854	HL7_MediaType::14828
"text/richtext"	Rich text encoded according to RFC2046	
"text/rtf"	Rich text encoded according to ftp://indri.primate.wisc.edu/pub/RTF/RTF-Spec.rtf.	HL7_MediaType::14831
"text/sgml"		HL7_MediaType::14829

Issuer: *IANA* Code set name: *“media types”*		
Code	**Description**	**Mappings**
“text/ rfc822-headers”		
“text/xml”		HL7_MediaType::14830
“audio/basic”		HL7_MediaType::14836
“audio/mpeg”		HL7_MediaType::14837
“application/pdf”		HL7_MediaType::14833
“application/msword”		HL7_MediaType::14834
...	...	...

5.3.5 Compression algorithms

This code set consists of the names of algorithms used to compress data, and is drawn from HL7’s CompressionAlgorithms domain.

Issuer: *openehr* Code set name: *“compression algorithms”*		
Code	**Description**	**Mappings**
“compress”	Original UNIX *compress* algorithm and file format using the LZC algorithm (a variant of LZW).	HL7_CompressionAlgorithm::10624
“deflate”	The *deflate* compressed data format as specified in RFC 1951. See `ftp://ftp.isi.edu/in-notes/rfc1951.txt`.	HL7_CompressionAlgorithm::10621
“gzip”	A compressed data format that is compatible with the widely used GZIP utility as specified in RFC 1952. See `ftp://ftp.isi.edu/in-notes/rfc1952.txt`.	HL7_CompressionAlgorithm::10622
“zlib”	A compressed data format that also uses the deflate algorithm. Specified as RFC 1950 See `ftp://ftp.isi.edu/in-notes/rfc1950.txt`	HL7_CompressionAlgorithm::10623
“other”	Some other type of compression; might be retrievable upon direct inspection of data.	

5.3.6 Integrity check algorithms

This code set consists of the names of algorithms used to generate hashes for the purpose of integrity checks on data; its initial values are drawn from the HL7 IntegrityCheckAlgorithm domain.

Issuer: *openehr* Code set name: “integrity check algorithms”		
Code	**Description (en)**	**Mappings**
“SHA-1”	Secure hash algorithm - 1. Defined in FIPS PUB 180-1: Secure Hash Standard. As of April 17, 1995.	HL7_IntegrityCheckAlgorithm::17386
“SHA-256”	secure hash algorithm - 256. Defined in FIPS PUB 180-2: Secure Hash Standard	HL7_IntegrityCheckAlgorithm::17387
...	...	

5.4 Vocabularies and Terminologies

5.4.1 Act Status

This vocabulary codifies act statuses of Entries.

Terminology*: openehr* Group_name(“en”): *“act status”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**

5.4.2 Attestation Status

This vocabulary codifies attestation statuses of Compositions or other elements of the health record, and is drawn from the HL7 ParticipationSignature domain, as used in CDA.

Terminology*: openehr* Group_name(“en”): *“attestation status”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
240	“signed”	This attestation has been signed by its required signatory/ies.	HL7_ParticipationSignature::10284
241	“intended”	This attestation is awaiting the signature of its signatory/ies.	HL7_ParticipationSignature::13977
242	“required”	This attestation requires the signature of its signatory/ies.	HL7_ParticipationSignature::10283

5.4.3 Audit Change Type

This vocabulary codifies the kinds of changes to data which are recorded in audit trails.

Terminology*: openehr* Group_name(“en”): *“audit change type”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
249	“creation”	Change type was creation.	HL7_CDA: CEN:
250	“amendment”	Change type was amendment.	HL7_CDA: CEN:
251	“modification”	Change type was modification.	HL7_CDA: CEN:
252	“synthesis”	Change type was synthesis - creation by a conversion gateway.	HL7_CDA: CEN:
253	“unknown”	Type of change unknown.	HL7_CDA: CEN:

5.4.4 Composition Category

This vocabulary codifies the values of the *category* attribute of the `COMPOSITION` class in the `rm.composition` package.

Terminology: *openehr* Group_name("en"): *"composition category"*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
	"persistent"	This Composition contains information which remains valid for (more or less) the life of the EHR. Typical persistent Compositions include "family history", "problem list", "current medications", and "vaccination history".	
	"event"		
	"process"		

5.4.5 Event Math Function

This vocabulary codifies mathematical functions of non-instantaneous events.

Terminology: *openehr* Group_name("en"): *"event math function"*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
145	"minimum"	Value of the interval-event is the minimum value of the discrete events which the interval-event summarises.	
144	"maximum"	Value of the interval-event is the maximum value of the discrete events which the interval-event summarises.	
267	"mode"	Value of the interval-event is the modal (most common) value of the discrete events which the interval-event summarises.	
268	"median"	Value of the interval-event is the median (centre value in sorted series) value of the discrete events which the interval-event summarises.	
146	"mean"	Value of the interval-event is the average value of the discrete events which the interval-event summarises.	
147	"delta"	Value of the interval-event is the net change over the period which the interval-event summarises.	
148	"total"	Value of the interval-event is the sum of the values of the discrete events which the interval-event summarises (typically differential flow measurements, e.g. blood loss).	

5.4.6 Measurable Properties

This vocabulary codifies purposes for physical properties corresponding to formal unit specifications, and allows comparison of Quantities with different units but which measure the same property. The vocabulary values are taken from:

- CEN ENV 12435 - "Medical Informatics - Expression of results of measurements in health sciences"

- HL7 “Unified Codes for Units of Measure”

Terminology*: openehr* Group_name(“en”): *“measurable properties”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
...	...	...	...

5.4.7 Null Flavours

This vocabulary codifies “flavours of null” for missing data items.

Terminology*: openehr* Group_name(“en”): *“null flavours”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
271	“no information”	No information provided; nothing can be inferred as to the reason why, including whether there might be a possible applicable value or not.	HL7_NullFlavor::V10610
253	“unknown”	A possible value exists but is not provided.	HL7_NullFlavor::V10612
272	“masked”	The value has not been provided due to privacy settings.	HL7_NullFlavor::17932
273	“not applicable”	No valid value exists for this data item.	HL7_NullFlavor::10611

5.4.8 Participation Function

This vocabulary codifies functions of participation of parties in an interaction (used in `PARTICIPATION` class).

Terminology*: openehr* Group_name(“en”): *“participation function”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**

5.4.9 Participation Mode

This vocabulary codifies modes of participation of parties in an interaction (used in PARTICIPATION class). The initial set has been defined to be the same as HL7's ParticipationMode vocabulary domain.

Terminology: *openehr* Group_name("en"): *"participation mode"*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
193	"not specified"	Mode of participation is not specified; use only for legacy data.	
216	"face-to-face communication"	Face to face communications between parties in the same room.	HL7_ParticipationMode::16545
223	"interpreted face-to-face communication"	Face to face communications between parties in the same room with an interpreter	HL7_ParticipationMode::16545
217	"signing (face-to-face)"	Live face-to-face communication using a recognised sign language.	
195	"live audiovisual; videoconference; videophone"	Any audio-visual communication in real time	
198	"videoconferencing"	Live audio-visual communication over videoconferencing or other similar equipment.	HL7_ParticipationMode::16548
197	"videophone"	Live audio-visual communication	
218	"signing over video"	Live video communication using sign language.	
224	"interpreted video communication"	Live audio-visual communication involving an interpreter	
194	"asynchronous audiovisual; recorded video"	Audio-visual communication that is not live	
196	"recorded video"	Recorded video or video mail	
202	"live audio-only; telephone; internet phone; teleconference"	Any live audio-only communication.	HL7_ParticipationMode::V16544 (includes live)
204	"telephone"	Live verbal communication over a telephone.	HL7_ParticipationMode::16546
203	"teleconference"	Live verbal communication over teleconference	HL7_ParticipationMode::16546
204	"internet telephone"	Live verbal communication over a the internet.	HL7_ParticipationMode::16546
222	"interpreted audio-only"	Any live audio-only communication using an interpreter.	HL7_ParticipationMode::V16544 (includes live)
199	"asynchronous audio-only; dictated; voice mail"	Audio-only communication that is not live.	
200	"dictated"	Non-interactive audio-only information recorded on some medium, such as cassette tape.	HL7_ParticipationMode::16547
201	"voice-mail"	Audio messaging system	

Terminology*: openehr* Group_name(“en”): *“participation mode”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
212	“live text-only; internet chat; SMS chat; interactive written note”	Any live text-only communication	
213	“internet chat”	Live text-only communication over the internet	
214	“SMS chat”	Live text-only chat over mobile/cell phone	
215	“interactive written note”	Live text-only communication using written notes	HL7_ParticipationMode::16550
206	“asynchronous text; email; fax; letter; handwritten note; SMS message”	Any text-only communication including email, written text, SMS message etc.	HL7_ParticipationMode::V16549
211	“handwritten note”	Written communication by handwritten document.	HL7_ParticipationMode::16550
210	“printed/typed letter”	Written communication by typewritten document.	HL7_ParticipationMode::16551
207	“email”	Written communication by email.	HL7_ParticipationMode::16553 [inlcude HL7_ParticipationMode::16554 (electronic data)]
208	“facsimile/telefax”	Non-interactive written communication using a fax machine.	HL7_ParticipationMode::16552
221	“translated text”	Non-interactive written communication requiring translation	HL7_ParticipationMode::V16549
209	“SMS message”	Messages sent via mobile/cell phone	
219	“physically present”	Participation by actions, where the participant is physically present.	HL7_ParticipationMode::16556
220	“physically remote”	Participation by actions, where the participant is not physically present, and the actions are transmitted by electronic means.	HL7_ParticipationMode::16557

5.4.10 Related Party relationship

This vocabulary codifies the relationship between the subject of care and some other party mentioned in the health record.

Terminology*: openehr* Group_name(“en”): *“related party relationship”*			
Concept id	**Rubric (en-uk)**	**Description (en)**	**Mappings**
0	“self”	The party is the subject of EHR	HL7_RoleCode:: CEN:
3	“foetus”	The party is a foetus	HL7: CEN:
10	“mother”	The party is the mother of the subject of EHR	HL7: CEN:
9	“father”	The party is the father of the subject of the EHR	HL7: CEN:

Terminology*: openehr* Group_name(“en”): *“related party relationship”*			
Concept id	**Rubric (en-uk)**	**Description (en)**	**Mappings**
6	“donor”	The party is a donor of organs or other body products to the EHR subject.	HL7: CEN:
253	“unknown”	Relationship to party is unknown.	HL7: CEN:
261	“adopted daughter”	Relationship of adopted daughter to subject of EHR	HL7: CEN:
260	“adopted son”	Relationship of adopted son to subject of EHR	HL7: CEN:
259	“adoptive father”	Relationship of adoptive father to subject of EHR	HL7: CEN:
258	“adoptive mother”	Relationship of adoptive mother to subject of EHR	HL7: CEN:
256	“biological father”	Relationship of biological father to subject of EHR	HL7: CEN:
255	“biological mother”	Relationship of biological mother to subject of EHR	HL7: CEN:
23	“brother”	Relationship of brother to subject of EHR	HL7: CEN:
28	“child”	Relationship of child to subject of EHR	HL7: CEN:
265	“cohabitee”	Lives with the subject of EHR	HL7: CEN:
257	“cousin”	Relationship of cousin to subject of EHR	HL7: CEN:
29	“daughter”	Relationship of daughter to subject of EHR	HL7: CEN:
264	“guardian”	Relationship of guardianto subject of EHR	HL7: CEN:
39	“maternal aunt”	Relationship of maternal aunt to subject of EHR	HL7: CEN:
8	“maternal grandfather”	Relationship of maternal grandfather to subject of EHR	HL7: CEN:
7	“maternal grandmother”	Relationship of maternal grandmother to subject of EHR	HL7: CEN:
38	“maternal uncle”	Relationship of maternal uncle to subject of EHR	HL7: CEN:
189	“neonate”	Relationship of neonate to subject of EHR	HL7: CEN:
254	“parent”	Relationship of parent to subject of EHR	HL7: CEN:
22	“partner/spouse”	The husband or wife or life partner of the subject of EHR	HL7: CEN:
41	“paternal aunt”	Relationship of paternal aunt to subject of EHR	HL7: CEN:
36	“paternal grandfather”	Relationship of aternal grandfather to subject of EHR	HL7: CEN:
37	“paternal grandmother”	Relationship of paternal grandmother to subject of EHR	HL7: CEN:

Terminology*: openehr* Group_name(“en”): *“related party relationship”*			
Concept id	**Rubric (en-uk)**	**Description (en)**	**Mappings**
40	“paternal uncle”	Relationship of paternal uncle to subject of EHR	HL7: CEN:
27	“sibling”	Relationship of sibling to subject of EHR	HL7: CEN:
24	“sister”	Relationship of sister to subject of EHR	HL7: CEN:
31	“son”	Relationship of son to subject of EHR	HL7: CEN:
263	“step father”	Relationship of step father to subject of EHR	HL7: CEN:
262	“step mother”	Relationship of step mother to subject of EHR	HL7: CEN:
25	“step or half brother”	Relationship of step or half brother to subject of EHR	HL7: CEN:
26	“step or half sister”	Relationship of step or half sister to subject of EHR	HL7: CEN:

5.4.11 Setting

This vocabulary codifies broad types of settings in which clinical care is delivered. It is not intended to be a perfect classification of the real world, but instead a practical coarse-grained categorisation to aid querying.

Terminology*: openehr* Group_name(“en”): *“setting”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
225	“home”	Care delivered in the patient’s home by patient or health professional.	
227	“emergency care”	Care delivered in emergency situation, e.g. by ambulance workers.	
228	“primary medical care”	Care delivered by a doctor within a primary care framework (generalist, non-referred).	
229	“primary nursing care”	Care delivered by nurses within a primary care framework (community based, generalist clinic).	
230	“primary allied health care”	Care delivered by allied health practitioners such as physiotherapists, osteopaths, chiropracters, optometrists, chiropodist/pediatrist etc. within a primary care framework (community based, generalist clinic)	
231	“midwifery care”	Midwifery care in any framework	
232	“secondary medical care”	Care delivered in an institutional or specialist setting - usually as a result of a referral.	
233	“secondary nursing care”	Care delivered by nurses within a secondary care framework (inpatient, specialist clinic).	
234	“secondary allied health care”	Care delivered by allied health care professionals within a secondary care framework (inpatient, specialist clinic).	

Terminology: *openehr* Group_name(“en”): *“setting”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
235	“complementary health care”	Care delivered by chinese, ayurvedic, naturopath, homeopath etc practitioner.	
236	“dental care”	Care delivered in a dental practitioner setting.	
237	“nursing home care”	Care to the needs of patients in nursing homes, delivered in an institutional setting.	
238	“other care”	Care delivered in setting not described by other terms in this vocabulary.	

5.4.12 Term Mapping Purpose

This vocabulary codifies purposes for term mappings as used in the class `TERM_MAPPING`. The use-case for this vocabulary is yet to be determined.

Terminology: *openehr* Group_name(“en”): *“term mapping purpose”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
...	to be determined	...	

5.4.13 Version Lifecycle State

This vocabulary codifies lifecycle states of Compositions or other elements of the health record.

Terminology: *openehr* Group_name(“en”): *“version lifecycle state”*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
244	“draft”	Item is in draft state: not ready for viewing by other users.	
245	“active”	Item is active and available for shared use.	
246	“inactive”	Item is marked inactive due to logical deletion or other similar operation.	
247	“awaiting approval”	Item is awaiting to approval to go into active state.	

6 Measurement Package

6.1 Overview

The Measurement package defines a minimum of semantics relating to quantitative measurement, units, and conversion, enabling the Quantity package of the *open*EHR Data Types Information Model to be correctly expressed. As for the Terminology package, a simple service interface is assumed, which provides useful functions to other parts of the reference model. The definitions underlying measurement and units come from a variety of sources, including:

- CEN ENV 12435, Medical Informatics - Expression of results of measurements in health sciences (see `http://www.centc251.org`);
- the Unified Code for Units of Measure (UCUM), developed by Gunther Schadow and Clement J. McDonald of The Regenstrief Institute (available in HL7v3 ballot materials; `http://www.hl7.org`).

These of course rest in turn upon a vast amount of literature and standards, mainly from ISO on the subject of scientific measurement.

6.2 Service Interface

A simple measurement data service interface is defined according to FIGURE 6, enabling quantitative semantics to be used formally from within the Reference Model. Note that this service as currently defined in no way seeks to properly model the semantics of units, conversions etc - it provides only the minimum functions required by the *open*EHR Reference Model.

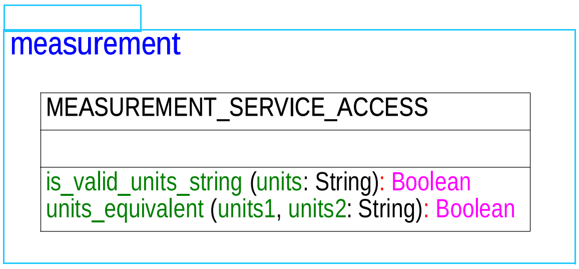
measurement
MEASUREMENT_SERVICE_ACCESS
is_valid_units_string (units: String): Boolean
units_equivalent (units1, units2: String): Boolean

FIGURE 6 rm.support.measurement Package

6.2.1 Class Definitions

6.2.1.1 MEASUREMENT_SERVICE_ACCESS Class

CLASS	**MEASUREMENT_SERVICE_ACCESS**	
Purpose	Defines an object providing proxy access to a measurement information service.	
Functions	**Signature**	**Meaning**
	is_valid_units_string (units: `String`): `Boolean` ***require*** units /= Void	True if the units string ‘units’ is a valid string according to the HL7 UCUM specification.

CLASS	MEASUREMENT_SERVICE_ACCESS	
	units_equivalent (units1, units2: `String`): `Boolean` ***require*** units1 /= Void ***and then*** is_valid_units_string(units1) units2 /= Void ***and then*** is_valid_units_string(units2)	True if two units strings correspond to the same measured property.
Invariants		

7 Definition Package

This section describes symbolic definitions used by the *open*EHR models.

7.1 Constants

Constants used in the *open*EHR Reference Model specifications:

- **CR**: Character is ‘\015’
- **LF**: Character is ‘\012’

A References

A.1 General

1 Cimino J J. *Desiderata for Controlled Medical vocabularies in the Twenty-First Century*. IMIA WG6 Conference, Jacksonville, Florida, Jan 19-22, 1997.

END OF DOCUMENT