EHRServer v1.0 openEHR Conformance

The open source, service-oriented, openEHR clinical data repository

2017-07-31

Compliance with:

Pablo Pazos Gutiérrez
Director at CaboLabs
pablo.pazos@cabolabs.com

Index

Introduction

This conformance statement shows which parts of the openEHR specifications are implemented in the EHRServer. Since the openEHR specs are a set of different component definitions, a subset of the specs really applies to the scope of the EHRServer.

There are some areas of the openEHR specification that are currently being defined by the openEHR Standard Editorial Committee (SEC), and the EHRServer will comply with them when they become part of the released specs.

openEHR RM 1.0.2 conformance

The openEHR Reference Model is the core of the openEHR specifications.

The EHRServer supports these classes from the openEHR RM 1.0.2:

rm_type_name	package	comments
EHR	ehr	
VERSIONED_COMPOSITION	ehr	
FOLDER	common.directory	No support for VERSIONED_FOLDER yet.
COMPOSITION	composition	
EVENT_CONTEXT	composition	
PARTY_PROXY	common.generic	
SECTION	content.navigation	
OBSERVATION	entry	
EVALUATION	entry	
INSTRUCTION	entry	
ACTION	entry	
ADMIN_ENTRY	entry	
ACTIVITY	entry	
ISM_TRANSITION	entry	
INSTRUCTION_DETAILS	entry	
PATHABLE	common.archetyped	
LOCATABLE	common.archetyped	
ARCHETYPED	common.archetyped	
AUDIT_DETAILS	common.generic	Support for ATTESTATION will be added soon.
PARTY_SELF	common.generic	
VERSIONED_OBJECT	common.change_control	Support via VERSIONED_COMPOSITION
CONTRIBUTION	common.change_control	
ORIGINAL_VERSION	common.change_control	
LOCATABLE_REF	support.identification	
TERMINOLOGY_ID	support.identification	
OBJECT_VERSION_ID	support.identification	
HIER_OBJECT_ID	support.identification	
ARCHETYPE_ID	support.identification	
TEMPLATE_ID	support.identification	

rm_type_name	package	comments
HISTORY	data_structures.history	
POINT_EVENT	data_structures.history	
INTERVAL_EVENT	data_structures.history	
ITEM_TREE	data_structures.item_structure	
ITEM_LIST	data_structures.item_structure	
ITEM_TABLE	data_structures.item_structure	
ITEM_SINGLE	data_structures.item_structure	
CLUSTER	data_structures.representation	
ELEMENT	data_structures.representation	
DV_BOOLEAN	data_types.basic	
DV_IDENTIFIER	data_types.basic	
DV_TEXT	data_types.text	
DV_CODED_TEXT	data_types.text	
CODE_PHRASE	data_types.text	
DV_ORDINAL	data_types.quantity	Support for DV_INTERVAL <DV_ORDERED> will be added soon.
DV_PROPORTION	data_types.quantity	
DV_COUNT	data_types.quantity	
DV_QUANTITY	data_types.quantity	
DV_DURATION	data_types.quantity.date_time	
DV_DATE	data_types.quantity.date_time	
DV_DATE_TIME	data_types.quantity.date_time	
DV_PARSABLE	data_types.encapsulated	
DV_MULTIMEDIA	data_types.encapsulated	

openEHR AOM 1.0.2 / ADL 1.4 conformance

EHRServer uses Operational Templates (OPT) directly. openEHR archetypes in ADL are used to generate OPTs that contain the same structure, constraints and terminology as the referenced archetypes. There is no direct support for AOM/ADL in the EHRServer.

openEHR TOM 1.0.2 conformance

EHRServer uses Operational Templates (OPT) in their XML form as clinical document definitions, to index data and to generate data queries.

OPTs in XML should comply with the OperationalTemplate XSD accessible here: https://github.com/ppazos/cabolabs-ehrserver/blob/master/xsd/OperationalTemplate.xsd

That XSD complies with the output from the Ocean Template Designer[1] when exporting an OPT.

[1] http://www.openehr.org/downloads/modellingtools

openEHR REST API v1.0 conformance

TBD.

Currently the openEHR SEC[2] is specifying the openEHR REST API.

The EHRServer will be compliant with the openEHR REST API when it becomes a normative part of the specifications.

[2] http://www.openehr.org/programs/specification/

openEHR Querying v1.0 conformance

Querying conformance includes two parts, the query definition and the query result set model.

Queries

AQL is not yet supported by EHRServer but it's on the roadmap for the next versions.

Currently EHRServer queries are path-based queries, using openEHR Archetype paths as references of values stored in the database, used as projections for queries (values to get) or to specify conditions (query criteria). This is the same approach as AQL. The only step missing from AQL support is the actually integrate the AQL syntax into EHRServer queries. The current functionality won't change much.

Result Sets

The results returned by queries will be defined by the Result Set model of the openEHR REST API (under development). When this is defined and approved, the EHRServer will implement openEHR compliant result sets for queries.

In the mean time, EHRServer Result Sets return openEHR clinical documents (COMPOSITIONs) in openEHR compliant XML and JSON representations. Queries that return datavalues, return simplified results, also in XML and JSON, and grouped in different ways (by COMPOSITION or by archetype path). The format itself is EHRServer specific, but the data is equivalent and consistent with openEHR DATA_VALUEs.

openEHR Architectural Components v1.0 conformance

openEHR defines a set of architectural components and services[3] that enable the implementation of a Virtual EHR (vEHR) API, that is the higher level interface that an Enterprise EHR Platform has, and it is what enables Cross-Enterprise Semantic Interoperability.

The EHRServer implements the following components / services:

- EHRService: includes API + CDR
- Audit Log Service: EHR-related audit log
- Enterprise Knowledge Service: the part that manages Operational Templates
- Security Service: the part that controls security on the EHR
- Notification Service: focused on EHR management and audit

[3] https://openehr.atlassian.net/wiki/spaces/spec/pages/50561091/Architectural+concepts

openEHR Versioning v1.0 conformance

Versioning Design

EHRServer support linear versioning, using the trunk_version component of the VERSION_TREE_ID class, without opening branches for different OBJECT_VERSION_ID.creating_system_id. So two instances of OBJECT_VESION_ID for two versions of the same object can be:

7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::1
7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::ATHENA_HEALTH_EMR::2

The decision of not creating branches for CABOLABS_EMR and ATHENA_HEALTH, is because 1. need of tracking of the latest version per creating system to resolve pulling data from the REST API, and 2. a merging functionality, that is rarely used in health care, should be implemented to merge branches.

The openEHR specifications allow branching and merging, but that adds complexity to the platform, with arguable value or usability to the end user.

VERSION_TREE_ID.trunk_version is controlled by the EHRServer, that means clients should not worry about tracking, assigning and updating version numbers. Only the first version number is needed from the client.

Versioned Objects

Currently EHRServer supports VERSIONED_COMPOSITION (that is equivalent to VERSIONED_OBJECT<COMPOSITION>) for versioning clinical documents.

Versioning Workflow

When a clinical document is committed for the first time to the EHRServer, it should have an OBJECT_VERSION_ID with: 1. object_id is the UID of the object that represents this document and all its versions, 2. creating_system_id is the identifier of the client system using the EHRServer API, 3. version_tree_id should be “1”. For that commit, the change_type is “creation”.

The OBJECT_VERSION_ID will look like:

7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::**1**

Then, when a version of that document needs to be created, another document should be committed, and the OBJECT_VERSION_ID should be the same "7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::**1**". And the change_type should not be "**creation**", can be "**amendment**", "**modification**", etc.

The EHRServer will know that the document "7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::**1**" should be versioned, so it will assign the OBJECT_VERSION_ID "7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::**2**" to the new document.

If another version of the document needs to be created, the client will commit using the latest version OBJECT_VERSION_ID "7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::**2**", and the EHRServer internally assigns "7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::**3**" to the new version. And so on.

When pulling compositions from the API or executing queries, only data from the latest versions will be retrieved.

As aforementioned, the OBJECT_VERSION_ID.creating_system_id might vary on the different versions. So this sequence of commits should be valid:

OBJECT_VERSION_ID	change_type	version assigned by server
7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CABOLABS_EMR::**1**	creation	1
7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::CLINWEB::**1**	amendment	2
7b83a3fa-c29f-4c73-b852-eb3e4c0c7d69::PATIENT_PORTAL::**2**	modification	3

That should be read as:

1. creation of document 7b83... version 1
2. amendment of document 7b83... version 1, creates version 2
3. modification of document 7b83... version 2, creates version 3

openEHR Data Validation v1.0 conformance

The data validation is focused on validating data from clinical documents that are committed to the server. There are three types of data validation:

1. Syntactic: verifies the data received by the EHRServer complies with the openEHR format to only accept well formed data.
2. Technical: verifies rules over specific data points for data consistency.
3. Semantic: verifies the data received by the EHRServer complies with the semantic constraints of Operational Templates / Archetypes.

The EHRServer supports syntactic validation using the openEHR XSD as the main source of verification of well formed data. For JSON commits, the JSON is transformed to XML then validated with the XSD.

The EHRServer supports technical validations besides well formed / syntactic validations. Some examples are:

1. existence of the EHR when committing a document
2. can't commit with change_type "creation" and version_tree_id > 1
3. existence of the object_id if change_type is "amendment" or "modification"
4. a new version is created from the latest version of the same object
5. consistency of contribution identifiers when committing more than one document at the same time

This enables the maintenance of data consistency, considering what is pushed from clients can't be controlled by the server.

Semantic data validation is not yet implemented but is on the roadmap. Also will be exposed as a service on the EHRServer REST API, so clients can validate documents before committing them. This feature will be particularly useful for developers.

Pablo Pazos Gutiérrez
Director at CaboLabs
pablo.pazos@cabolabs.com