



# CaboLabs

## EHRServer v1.1 openEHR Conformance

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

2019-10-16

Compliance with:

*open***EHR**

Pablo Pazos Gutiérrez  
Director at CaboLabs  
[pablo.pazos@cabolabs.com](mailto:pablo.pazos@cabolabs.com)

# Index

Introduction .....	3
openEHR RM 1.0.2 conformance.....	4
openEHR AOM 1.0.2 / ADL 1.4 conformance .....	6
openEHR TOM 1.0.2 conformance .....	7
openEHR REST API v1.0.1 conformance .....	8
openEHR Querying v1.0 conformance.....	9
Queries.....	9
Result Sets.....	9
openEHR Architectural Components v1.0 conformance.....	10
openEHR Versioning v1.0 conformance .....	11
Versioning Design .....	11
Versioned Objects.....	11
Versioning Workflow .....	11
Supported Change Types and Lifecycle States .....	12
openEHR Data Validation v1.0 conformance .....	14

## **Introduction**

Since the openEHR specs are a set of different component definitions, a subset of these specifications applies to the scope of the EHRServer. This Conformance Statement document shows which parts of the openEHR specifications are implemented in the EHRServer.

## 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
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	

<b>rm_type_name</b>	<b>package</b>	<b>comments</b>
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
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) as data set definitions, openEHR Archetypes are used to generate OPTs, so all constraints and terminology are available in the final OPT. Because of this, 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<sup>1</sup> when exporting an OPT.

---

<sup>1</sup> <http://www.openehr.org/downloads/modellingtools>

## **openEHR REST API v1.0.1 conformance**

At the moment of updating this Conformance Statement, we added the openEHR API implementation in our roadmap for the next months.



## **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<sup>2</sup> 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

---

<sup>2</sup> <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

## ***Supported Change Types and Lifecycle States***

The EHRServer versioning mechanism supports to commit COMPOSITIONs with these change types:

- creation: a new document is added to an EHR
- amendment: a document is corrected, had incorrect information
- modification: a document is modified, e.g. missing data is added
- deleted: a document is deleted from an EHR

And the supported lifecycle states are:

- complete: the document is complete
- incomplete: the document is incomplete, further commits are needed to complete
- deleted: when deleting an existing document

## 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. This will also 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](mailto:pablo.pazos@cabolabs.com)