openEHR
Conformance Framework Design

Author	Date	Comment
Pablo Pazos <pablo.pazos@cabolabs.com>	2022-01-25	First Release (WIP).
Pablo Pazos <pablo.pazos@cabolabs.com>	2022-02-20	Added terms to glossary.
Pablo Pazos <pablo.pazos@cabolabs.com>	2022-03-22	Added a new “delete behavior” section.
Pablo Pazos <pablo.pazos@cabolabs.com>	2023-01-29	Fixed links and text in the glossary.
Pablo Pazos <pablo.pazos@cabolabs.com>	2023-10-21	Improved wording, added missing delete behavior sections.
Pablo Pazos <pablo.pazos@cabolabs.com>	2023-11-07	Rewording “versioning mechanism” as “versioning scheme”
Pablo Pazos <pablo.pazos@cabolabs.com>	2023-11-08	Improved Conformance Verification Process section, removed software certification from scope. Added section about conformance asset versioning.

Conformance to openEHR specifications, processes and policies provides consumers and other stakeholders the confidence a product meets the requirements established by the openEHR specifications, and gives companies a competitive edge, allowing them to claim their product is "openEHR compliant".

To be able to verify a product's conformance with openEHR, all the elements participating in such verification / testing should be defined, creating an "openEHR Conformance Framework".

This document defines the basis for such openEHR Conformance Framework, aiming to help providers or consumers of openEHR-based software in the process of openEHR Conformance Verification of openEHR Products.

Glossary:

Conformance Framework
Context, specifications, guides and processes related to Conformance Verification.

Conformance Verification
Is the process to verify an openEHR Product complies with the openEHR Specifications.

openEHR Product
Any software system, application or module based on the openEHR Specifications.

openEHR Product Classification
Specification of different Types of Products that could be created based on the openEHR Specifications. This classification is used to customize Testing Specifications and Test Reports for each Type of Product, allowing to compare openEHR Products under the same category.

Conformance Specification
It's the official openEHR Conformance Specification being developed based on items described on this document and Test Suites designed in the context of the HiGHmed project, then continued by CaboLabs.

Conformance Testing Specification
This is the abstract documentation of the Test Suites and Test Data Sets. The original specification work was done in the context of EHRBASE for Conformance Verification and can be found here. The Conformance Testing Specification was shared with openEHR to be part of the Conformance Specification, though the openEHR copy could be outdated because we continued to work on the Test Cases for EHRBASE, and after that continued to work independently from CaboLabs, so EHRBASE Test Cases might also be outdated at this moment. In openEHR this specification is called Conformance Test Schedule.

Test Data Set
Is a Data Set used as input for the execution of operations provided by interfaces defined by the openEHR Service Model and implemented on each type of product. The term "Test Data Set" is used interchangeably with just "Data Set".

Test Case
Is the list of steps that should be followed to execute an operation and verify the results conform with the expected ones. A Test Case uses the Data Sets as input to execute the operations, and for each input, it defines the expected result.

Test Suite
Test Suite a group of Test Cases and Data Sets that could be used by each Test Case.

Conformance Statement
Document in which a Software Provider can claim compliance with different components of the openEHR Specification, and state any design/implementation deviations from those specifications.

Reference Architecture
Defines a set of architectural components (providers, consumers and internal) for which there could be Test Cases defined to verify conformance of those components with the openEHR Specifications.

SUT
System Under Test, si the product being verified for conformance

Conformance Test Report
This is the output of running concrete Test Suites (implementation of the Conformance Testing Specification) against a specific Type of Product defined by the openEHR Product Classification. In the current openEHR Specifications this is called Conformance Certificate, but that name is not correct in the sense a Test Report is not a formal Conformance Certificate. There should be a process of reviewing the Test Reports from testing a SUT, including some technical implementation considerations, done by a Conformance Expert, in order to deliver a formal Conformance Certificate to the Software Provider.

ITS
Implementation Technology Specification.

RFI
Request For Information.

RFT
Request For Tender.

RFP
Request For Proposals.

RFQ
Request For Quote.

Conformance Score
Numeric measurement on “how compliant” a system is with the openEHR specs.

Index:

1. openEHR Conformance Framework

The Conformance Framework defines the context, components, specifications, processes, documents, and any other conformance-related material needed for Conformance Verification of any kind of openEHR Product.

The Conformance Framework is composed by:

1. Conformance Verification Context
2. Product Classification (conformance verification will depend on the product kind/class)
3. Materials / Assets (guides, specifications, criteria) needed to verify conformance
4. Conformance Verification Process

1.1. Conformance Verification Context

This component defines which stakeholders might participate in the Conformance Verification processes, representing their specific needs and requirements related to Conformance Verification.

1.1.1. openEHR Consumer/Customer/Buyer

This type of stakeholder represents an organization that needs systems that are compliant with the openEHR Specifications. Some examples are: clinics, hospitals, universities and government.

The needs / requirements for this stakeholder are:

1. Create tenders/biddings (RFI, RFT, RFP, RFQ, etc.)
2. Evaluate offerings from vendors

1.1.2. openEHR Software Provider/Vendor

This type of stakeholder represents software companies that build or represent a brand, and sells openEHR compliant software to Customers. Some examples are: software developers, software houses, local representation of products from other international companies, solution integrators.

The needs / requirements for this stakeholder are:

1. Create / deliver proposals / offers
2. Test implementation (self compliance assessment)

1.1.3. openEHR Foundation

This stakeholder represents the openEHR Foundation, which is the official custodian of all openEHR related assets, including the openEHR Specifications (Service Model, REST API, and the newer Conformance Specification, among other specs).

The needs / requirements for this stakeholder are:

1. Define tender / bidding guidelines for openEHR products
2. Custody the openEHR Conformance Specification and Conformance Framework Components
3. Provide implementation assets to verify conformance (runnable test suites, test data sets)
4. Certify openEHR Products
5. Endorse openEHR Conformance Experts

1.1.4. openEHR Conformance Expert

This type of stakeholder represents an individual or company, which should be unrelated to the Software Providers to avoid conflict of interests, that can verify, assess and certify products comply or not with the openEHR Specifications by using the Conformance Framework Components.

The needs / requirements for this stakeholder are:

1. Support Customers on creating tenders / biddings
2. Support Customers on evaluating proposals from Software Providers
3. Support Software Providers on creating proposals / offers
4. Run Conformance Verification processes
5. Formally notify the openEHR Foundation a certain product complies or not with the openEHR Specifications

Note: it's recommended that an openEHR Conformance Expert that works on the customer side, doesn't work on the provider side for the same tender.

1.2. openEHR Product Classification

This section defines different kinds of openEHR products/systems that could be verified for conformance. In practical terms, these products will be verified using a specific Conformance Test Suite and there will be one Test Report structure for each Type of Product.

The need for this classification is because there are many kinds of products that could implement different parts of the openEHR specification, to satisfy a diverse set of requirements. As a consequence, not all products comply with all the components of the openEHR specifications, and not all products are comparable between them

To be able to verify conformance, we need to have a classification of products, then describe a Reference Architecture to each kind of product, to finally define a high level set of requirements for each kind of product. The requirements will be defined in terms of Interfaces provided by some components of the Reference Architecture of each kind of product. Those Interfaces are taken from the definitions of the openEHR Service Model.

The aforementioned components define the context for each Conformance Testing Specification, and without them it would be nearly impossible to formalize a Conformance Specification for openEHR with a certain quality level.

1.2.1. Classes of openEHR Products

This classification is not exhaustive, and an openEHR Product could fall into more than one classification. In the diagram below we tried to differentiate six big categories of products:

Programming Tool

This category includes any tool, component, module or library focused on helping in the development of openEHR-based systems.

User Facing Application

This category includes any application focused on data entry, display and analysis, that faces an end-user like a clinician, patient, researcher, etc., on any platform (mobile, desktop, web, IoT, etc.)

Knowledge Artifact Tool

This category includes any tool that is focused on creating, managing, updating, sharing, etc. any openEHR knowledge artifact like archetypes, templates, forms, processes/flows, rules, terminologies, etc.

Integration / Middleware

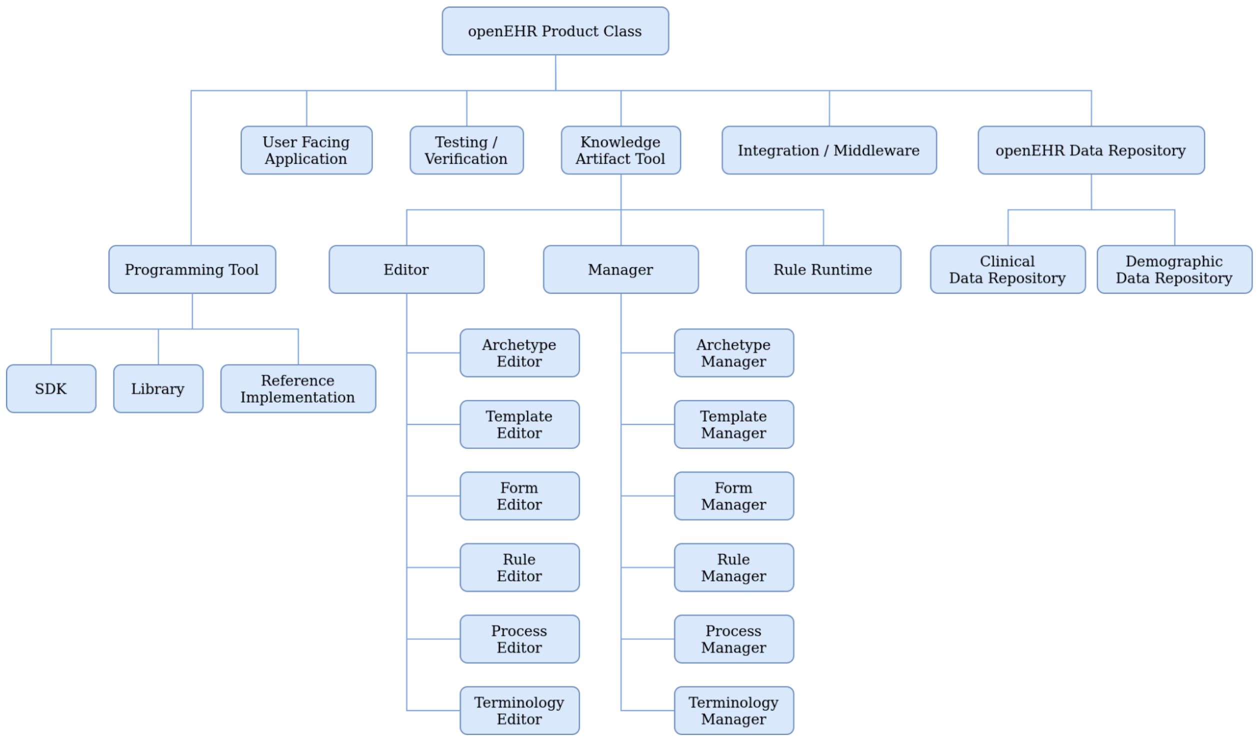
This category includes any tool, component, module or system that is focused on integrating other openEHR-based systems. That is, products of this kind would help to send, receive and transform messages/documents from/to openEHR-based systems.

openEHR Data Repository

This category includes any system that is focused on managing openEHR data based on the openEHR Reference Model that has two main components: EHR (clinical data) and DEMOGRAPHIC (people and organization data).

Testing / Verification

This category includes any tool focused on testing other tools and systems, or that helps on verification areas like conformance. So any tool written to verify conformance, as defined on this document, should fall under this category.



openEHR Product Classification

1.3. Conformance Verification Materials / Assets

Minimal materials/assets required for openEHR Conformance Verification:

1. Conformance Specification
2. Service Model Specification
3. Conformance Testing Specification *
4. Conformance Statement Specification *
5. Reference Architectures *
6. Conformance Test Reports *

Note: The components marked with an asterisk above should be defined for each product class / type. The openEHR Service Model (SM) already defines interfaces for components used in different classes of products, but the SM doesn't define which interfaces will be implemented by which types of products. This is defined in this document, in the Reference Architectures section.

Implementation Technology Specification (ITS) components related with Conformance Verification

1. REST API Specification (including exchange format schemas)
2. REST API Client
3. Conformance Test Suites (implementation of the Conformance Testing Specification)

Other useful materials

1. Tender/Bidding Guide
2. Tender/Bidding Offer Guide

1.3.1. Conformance Specification

The Conformance Specification is the next step for this Conformance Verification Design document. Basically is the glue spec that defines the context, goals, processes, rules, criteria, specifications and assets used in the openEHR Conformance Verification for a specific product kind.

The current conformance specification is under development in the official openEHR Specification site, and is based on the original design for the Conformance Framework described in this document.

1.3.2. Service Model Specification

This asset is a direct reference to the current openEHR Service Model. The Service Model is an abstract specification of Architectural Components named “Services”, that are modeled as packages. Each Component represents a topic or domain related to the openEHR Architecture and contains (as a package) the definition of one or more interfaces. Each interface exposes a set of operations that allow accessing and managing information related to the Component that contains the interface.

Those Service Model Components are not associated with any particular type of product (see Product Classification). The Reference Architecture of each type of product will contain these Service Model components (see section 3).

For running concrete tests against an openEHR software product, an implementation of the aforementioned Services is needed. Since the Service Model is an abstract specification, the concrete technical specification to be implemented is the REST API Specification.

It is key to understand the difference and relation between an Abstract Specification and the corresponding ITS Specification. An Abstract Specification is independent of implementation technologies, communication protocols and message formats. While an ITS Specification might depend on those implementation-related items.

The openEHR REST API Specification is a conceptual specialization of the openEHR Service Model Specification. Then the REST API defines the components that are needed for implementation: using HTTP as the communication protocol, JSON/XML message formats, etc. That is why the REST API spec is an Implementation Technology Specification or ITS.

Considering all these points, there should be a direct mapping between Service Model elements and REST API elements. The key aspect of this layered design approach to design specifications is to allow other technologies and communication protocols that could be implemented in the future instead of the REST approach (HTTP + JSON/XML). That is why the Conformance Testing Specification should rely only on the Service Model and avoid any references to the REST API.

Then different implementations that use other communication approaches than REST could also be verified in terms of openEHR Conformance. If the Conformance Testing Specification depended on the REST API, then Conformance Validation on non REST products would not be possible. Many colleagues that are focused on implementation argued these points, but this specific aspect is key for future-proof Conformance Verification, without the need of creating a new Conformance Specification for each implementation technology.

1.3.3. Conformance Testing Specification

This abstract specification defines WHAT should be tested, not HOW a concrete product will be tested. It relies only on the Service Model Specification and contains test cases for the different Operations (Services) contained on each interface defined in the Service Model.

A set of Test Cases is defined for each Operation. Each Test Case includes:

1. Precondition: state of the SUT before executing the Test Case
2. Flow: set of steps required to execute the corresponding Test Case
3. Postcondition: expected state of the SUT after the flow is completed

In general the last step of the Flow specifies the expected result of the Operation executed, this includes success cases and error cases, in which case the expected error should be specified, for instance “EHR with uid ‘xxx’ doesn’t exist”.

The first specification was created for the EHRBASE product, as a part of the HiGHmed Consortium, and can be found here.

We will call Test Suite a group of Test Cases that could be executed over a specific kind of product. The current Test Suites we have defined are:

1. EHR
2. CONTRIBUTION
3. COMPOSITION
4. DIRECTORY
5. KNOWLEDGE
6. VALIDATION_STRUCTURE
7. VALIDATION_DATATYPES

Note: other Test Suites could be designed in the future to verify different types of products. These initial Test Suites were designed focusing on verifying conformance of a CDR.

1.3.4. Conformance Statement Specification

The Conformance Statement is not directly related with Conformance Verification, but plays a key role in this openEHR Conformance environment. The Conformance Statement is a document with a specific structure (to be defined by the Conformance Statement Specification) that allows a Software Provider to specify claims about a product. Those claims are the first competitive edge a Software Provider can have, and it is insurance for Customers when they buy a product, because the Conformance Statement is part of the signed contract / agreement. So if something doesn’t work as it is specified in the Conformance Statement, the Customer can exercise its right to ask the Software Provider to comply with that claim.

This mechanism is not new. Conformance Statement documents have been used for years in healthcare IT, for instance when buying and evaluating offerings from Software Providers related with DICOM imaginology products. All DICOM products have a Conformance Statement document, and it's an industry standard to have one when selling a product. Also customers will always ask for this document.

The other role played by Conformance Statements is to state any design considerations and deviations from the openEHR specification. This could happen because of the design of the underlying technologies or of the specific product. For instance if a Clinical Data Repository doesn't support the branched openEHR versioning scheme but it supports sequential data versioning, that should be stated somewhere (in the Conformance Statement).

The Conformance Statement is related to the implementation of openEHR related features, and it's not a document to focus on product features, marketing materials, or compliance with other standards, so it should be focused on the openEHR domain.

As a recommendation, Customers shouldn't accept an offering of a product if it doesn't have the matching Conformance Statement.

See section 2. for the full Conformance Statement Specification.

A sample Conformance Statement (it's old but is a good reference) could be found at https://cloudehrserver.com/pages_en/guide/EHRServer_openEHR_Conformance_v1.1.pdf

1.3.5. Reference Architectures

Each kind of system (see Product Classification) will be composed of components, some of which will expose interfaces with services that could be consumed by other products or components. The idea of having a Reference Architecture for each kind of product is to detect which components have testable interfaces, and to associate the specific Test Cases to each product (those that match those interfaces). So the Reference Architecture is just an asset that allows mapping Test Cases to each Product Kind/Class.

The term "Reference" refers to the high level of abstraction of these Architectures. Which means the specific architecture of each product might differ from the matching Reference Architecture, but there should be a mapping between the Reference Architecture Components and the Concrete Architecture Components.

See section 3. for the specification of the Reference Architectures for each Product Kind.

1.3.6. Conformance Test Reports

The execution of the Conformance Test Suites, using a given data set, will generate a Conformance Test Report as a result. Note that each Product Kind will have its own

Conformance Test Report, since the Test Suites will also be executed depending on the Type of Product.

A detailed view of the Conformance Test Report for a specific Product, will have a list of test cases, each test case will have a unique name. For each Test Case, the Report will display the name, also unique, of the Data Set used to evaluate the Test Case. Then for each combination of Test Case and Data Set, the report will display a Result, which could be “accepted” (the expected result was met) or “rejected” (the expected result wasn’t met). Optionally, more information, metadata, code traces, etc. could be provided for each Result.

A summarized view of the Conformance Test Report will display the number of total Test Cases executed, and the total number of Data Sets evaluated for each test case, and the total number of “accepted” and “rejected” results. Optionally, it could display those indicators by Test Suite and include charts to simplify data visualization.

See section 4. for sample Conformance Test Reports.

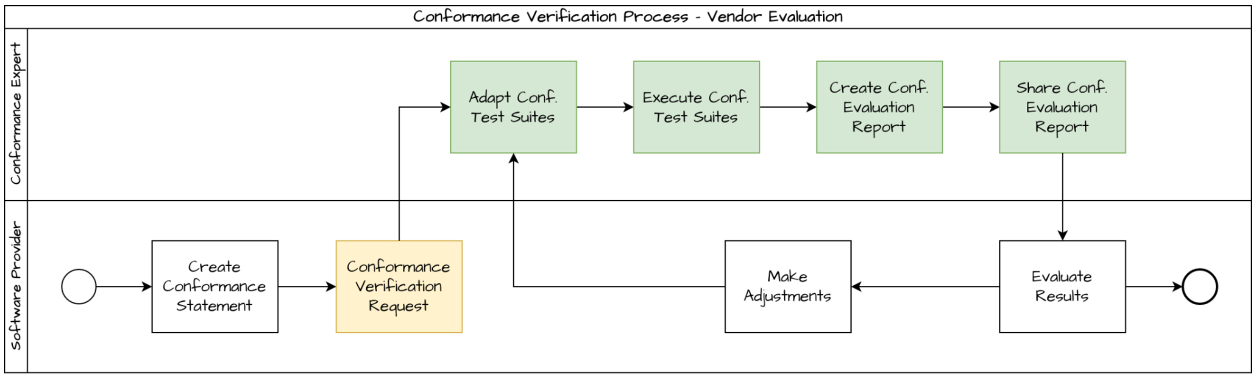
1.3.7. Conformance Asset Versioning

Since the Conformance Verification Framework will be an evolving specification, it requires strict versioning of all documents and assets like the Test Suites and Test Data Sets. So when the Conformance Verification Process is executed for a system, the Conformance Evaluation Report will explicitly state which Conformance Verification Asset versions were used. This will allow to correctly compare results for the same system at different points in time, and between different systems evaluated against the same Conformance Verification Asset baseline.

1.4. Conformance Verification Process

The Conformance Verification Process is required so the Conformance Framework described in this document can work in an objective, unbiased, and high quality environment that gives the needed assurance and reliability to all the stakeholders.

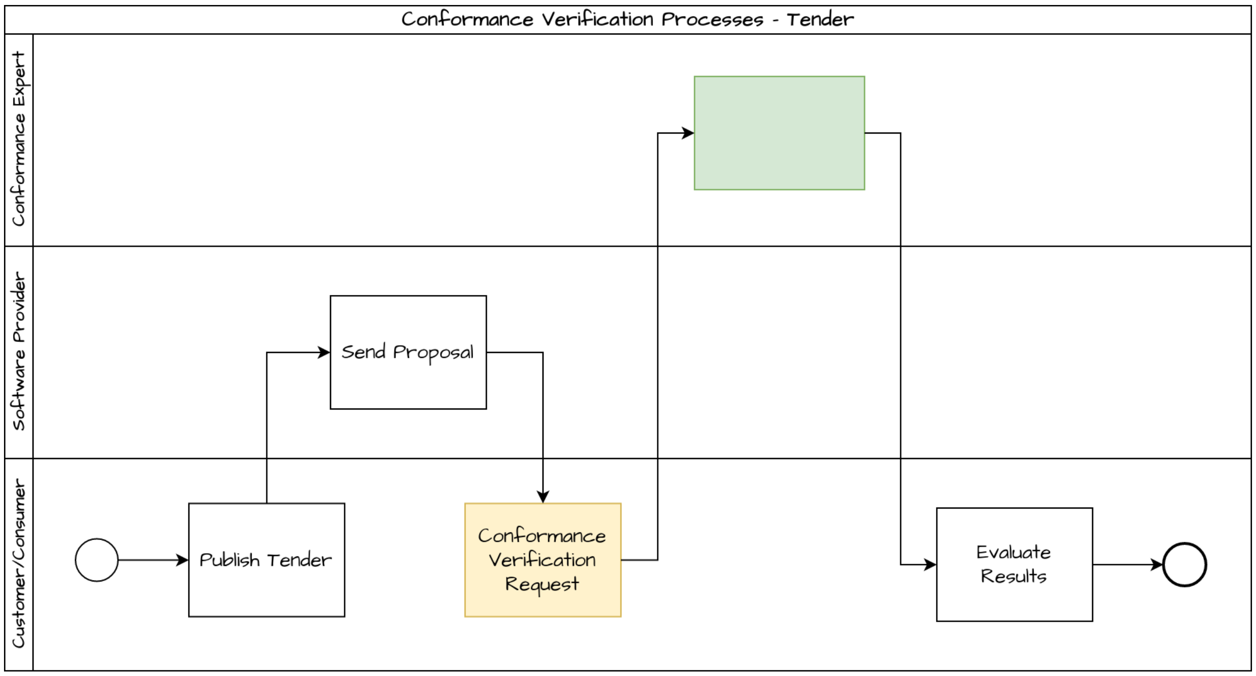
The Process has some core phases shown in green in the diagram below. Such phases are executed by the Conformance Expert, and could be triggered by the Software Provider or by the Software Consumer.


Conformance Verification Process - Vendor Evaluation
Conformance Expert
Software Provider
Adapt Conf. Test Suites
Execute Conf. Test Suites
Create Conf. Evaluation Report
Share Conf. Evaluation Report
Create Conformance Statement
Conformance Verification Request
Make Adjustments
Evaluate Results

Conformance Verification Process - Vendor Request

Before a Provider wants to request an Expert to run the Conformance Verification process on their systems, the Provider needs to create the Conformance Statement document, which is of vital importance for the Expert to be able to adapt the Test Suites before running the tests against that system. When the Test Suites are configured and adapted, the Expert will run the tests and interpret the results. Note that the cycle of adapting the Test Suites and interpreting the results might repeat until the Expert makes sure any test failures or successes are not because of how the tests were designed but are because of how the system being tested was implemented. All the adaptations of the Test Suites should be documented in the final Conformance Verification Report, which is shared with the Provider. Then the Provider can choose to make adjustments and request for another Conformance Verification run or accept the results as they are. For making adjustments, it could happen that a new version of the system is released with changes and fixes, then the Conformance Statement document should be updated, because it’s binded to a specific version of the system being evaluated.

The Conformance Verification can be requested by a Consumer. In that case, the Consumer might have received a proposal from one or many Providers and needs to evaluate if the systems in those proposals comply with openEHR. The Consumer will send a Conformance Verification request to the Expert, the expert will run the same steps as if the request came from the Provider, and the result in the form of a Conformance Evaluation Report will be delivered to the Consumer. Note that the Consumer should request access to the systems that should be evaluated and the Provider should grant access, for instance to a test server. The Consumer could share the report with the Provider if they see value on the solution but it needs some adjustments to improve the conformance score.



These are the basic Conformance Verification Processes and, as you can see, the openEHR Foundation is not included in them yet. The openEHR Foundation plays an important role in the next step for the Conformance Verification Framework, that is Software Certification. So Conformance Verification lays the foundation for Software Certification, which will require formal endorsement of openEHR Conformance Experts, the organization of a Conformance/Certification Program, and the formalization of rules and processes related to Software Certification, all these elements are out of the scope of the current document.

2. openEHR Conformance Statement Specification

2.1. Introduction

The Conformance Statement is a document with a specific structure that allows a Software Provider to specify claims about an openEHR Product.

The Conformance Statement is related to the implementation of openEHR related features, and it's not a document to focus on product features, marketing materials, or compliance with other standards, so it should be focused on the openEHR domain only.

2.2. Basic Requirements

2.2.1. Identification of the Product and openEHR Specifications

The Conformance Statement of any openEHR-based product should begin with a basic set of information required to give context to the document. That is:

1. Name, ID and version of the openEHR Product
2. Type of Product (based on the openEHR Product Classification, see section 1.2)
3. If applicable, which RM version(s) the product complies with
4. If applicable, which AOM version(s) the product complies with
5. If applicable, which REST API version(s) the product complies with
6. If applicable, which querying formalism and version(s) the product complies with

Note: the "if applicable" refers to the applicability of such specification to the Type of Product.

2.2.2. RM Packages and Classes Implemented

If the Product implements parts of the openEHR RM, the Conformance Statement should describe in detail which packages and classes are supported by the Product, and which packages and classes are not supported, including the reason why.

If a RM class is supported but there was any deviation of the openEHR specification in the implementation, it should be stated in the Conformance Statement. For instance, if a class is implemented but an attribute or operation is not supported or has a different type or doesn't work exactly in the way defined by the openEHR specifications.

In general, any design or technical decision that affects the implementation that deviates the implementation from the openEHR Specifications, should be mentioned in the openEHR Conformance Statement document.

2.2.3. Versioning scheme

If the Product supports the openEHR RM Storage, the Conformance Statement should describe the versioning scheme(s) supported by the Product, mentioning which classes are versioned, which aren't and the reason why.

2.2.4. Change Types an Lifecycle States

The Conformance Statement should include which Change Types and Lifecycle States are supported by the Product. The Change Type codes are used for audit logs when storing and versioning openEHR data, and the Lifecycle State is used in versioning of openEHR data.

For Products that don't support all codes for Change Type or Lifecycle State, the Conformance Statement should define how the system reacts to handling those unsupported codes.

2.2.5. Architecture

The Conformance Statement should include a description of the Product's Architecture, mentioning which openEHR Architectural Components are part of the Product.

These components could be based on the components defined in the openEHR Service Model specification and components of the Virtual EHR or vEHR mentioned on many specs of openEHR (this is not formally defined but it's used as a reference).

In a Conformance Verification environment it's not needed to include all the components of the Product in the Conformance Statement, but only the components related to openEHR. Either way it's recommended to include as many components as the Software Provider can, to clarify the Product's Architecture to Customers and Conformance Experts.

2.2.6. Standardization of the Conformance Statement

One goal of the Conformance Statement Specification is to standardize the sections and visual elements (diagrams) included in the Conformance Statement, that would help to compare these documents between different Products.

2.2.7. Delete behavior

Deleting data from a health information system is always a sensible point since it involves several medico-legal constraints and technical challenges. We need to consider that each implementer (openEHR Software Provider) can follow a different process for deleting data, which could depend on technical decisions, business decisions or even on local regulations. In any case, the "delete behavior" of different kinds of data should be stated on the Conformance Statement document.

2.2.7.1. Deleting openEHR templates

The Conformance Statement document should include a section that describes the conditions, process, roles, and final state of a system when a template is deleted, or state if this functionality is not supported at all.

Some considerations to deleting templates are:

1. Is deleting templates supported?
2. Which user roles can delete templates?
3. What happens with the existing data constrained by a template when the template is deleted?
3. Is new data accepted by the system when it references a deleted template?
4. Is the template physically or logically deleted from the system?

The Conformance Statement could mention more than these questions, but should have at least an answer for each of these questions.

2.2.7.2. Deleting openEHR Compositions

An openEHR Composition can represent documents or records that can include clinical information. Deleting a Composition is like destroying a document from a patient's EHR (physical delete) or marking the document as inactive (logical delete). Since this is a common action, this functionality should be supported by openEHR products that manage Compositions.

Some considerations for deleting compositions are:

1. Which user roles can delete Compositions? (note this might not have to do with a role, it could be about the user that created the Composition in the first place)
2. Does the system support undeleting (revert) Compositions?

2.2.7.3. Deleting EHRs

There are many requirements related to deleting EHRs. In some countries there is the "right to be forgotten" where a patient can ask for deleting their EHR from a system, since the owner of the information is the patient. When a patient moves, for instance to another country, they could ask to copy and delete their EHR to a new custodian. In any case, deleting an EHR is a sensible operation that should be controlled, and might not be an action that a normal user can do on a system that manages EHRs.

Some considerations to deleting EHRs are:

1. Is deleting EHRs supported?
2. Which user roles can delete EHRs?
3. Is the EHR physically or logically deleted from the system?

2.3. Sample Conformance Statement

This section defines a sample structure for the Conformance Statement document.

2.3.1. Conformance Document

Version	1.1
Release Date	2019-10-16
Author Name	Pablo Pazos
Author Email	pablo.pazos@cabolabs.com

2.3.2. Product Section

Product Name	EHRServer
Product Description	openEHR Clinical Data Management and Sharing Platform
Website	https://cloudehrserver.com/
Type	cdr
Version	2.3

2.3.3. Developer Section

Company/Organization/Individual	CaboLabs
Contact Name	Pablo Pazos
Contact Email	pablo.pazos@cabolabs.com
Website	https://cabolabs.com

2.3.4. Conformance Declaration Section

ehr_im	1.0.2
common_im	1.0.2
data_structures	1.0.2
data_types	1.0.2
support_im	1.0.2
query_formalism	SAQM
query_version	0.1
opt	1.4
rest_ehr_api	1.0.1
rest_definitions_api	1.0.1
rest_query_api	1.0.1

Note: there is no support for the DEMOGRAPHIC model in the product.

2.3.5. Detailed RM Conformance

package	rm_type_name	notes
ehr	EHR	
ehr	VERSIONED_COMPOSITION	
common.directory	FOLDER	No support for VERSIONED_FOLDER yet
composition	COMPOSITION	
composition	EVENT_CONTEXT	
common.generic	PARTY_PROXY	
content.navigation	SECTION	
entry	OBSERVATION	
entry	EVALUATION	
entry	INSTRUCTION	
package	**rm_type_name**	**notes**
entry	ACTION	
entry	ADMIN_ENTRY	
entry	ACTIVITY	
entry	ISM_TRANSITION	
entry	INSTRUCTION_DETAILS	
common.archetyped	PATHABLE	
common.archetyped	LOCATABLE	
common.archetyped	ARCHETYPED	
common.generic	AUDIT_DETAILS	ATTESTATION is not yet supported.
common.generic	PARTY_SELF	
package	**rm_type_name**	**notes**
common.change_control	VERSIONED_OBJECT	Supported via VERSIONED_COMPOSITION

common.change_control	CONTRIBUTION	
common.change_control	ORIGINAL_VERSION	No support for IMPORTED_VERSION yet.
support.identification	LOCATABLE_REF	
support.identification	TERMINOLOGY_ID	
support.identification	OBJECT_VERSION_ID	
support.identification	HIER_OBJECT_ID	
support.identification	ARCHETYPE_ID	
support.identification	TEMPLATE_ID	
data_structures.history	HISTORY	
package	**rm_type_name**	**notes**
data_structures.history	POINT_EVENT	
data_structures.history	INTERVAL_EVENT	
data_structures.item_structure	ITEM_TREE	
data_structures.item_structure	ITEM_LIST	
data_structures.item_structure	ITEM_TABLE	
data_structures.item_structure	ITEM_SINGLE	
data_structures.representation	CLUSTER	
data_structures.representation	ELEMENT	
data_types.basic	DV_BOOLEAN	
data_types.basic	DV_IDENTIFIER	
package	**rm_type_name**	**notes**
data_types.text	DV_TEXT	
data_types.text	DV_CODED_TEXT	
data_types.text	CODE_PHRASE	
data_types.quantity	DV_ORDINAL	
data_types.quantity	DV_PROPORTION	
data_types.quantity	DV_COUNT	

data_types.quantity	DV_QUANTITY	
data_types.time	DV_DURATION	
data_types.time	DV_DATE	
data_types.time	DV_DATE_TIME	
data_types.time	DV_TIME	
package	**rm_type_name**	**notes**
data_types.encapsulated	DV_PARSABLE	
data_types.encapsulated	DV_MULTIMEDIA	

2.3.6. AOM Conformance

What is needed in this section is to know if the Product uses openEHR Archetypes, Templates or both, which version, and if those are used for data validation.

Item	**Compliance**	**Used for data validation**	**Notes**
Archetype	Not supported		EHRServer is based on openEHR templates
Template	Supported v1.4	Yes. Structural and data value validation is evaluated against the corresponding OPT.	OPTs based on this XSD which is older than the current openEHR OPT XSD.

2.3.7. Versioning Scheme

Versioning Supported	Yes	
Versioning Scheme	Sequential	
Versioned Classes	COMPOSITION	

Note: Versioning Scheme could be Sequential or Branched. The openEHR VERSION_TREE_ID supports version numbers to be specified in a tree structure (branched versioning), though that approach is difficult to implement in real applications because a merging process would be needed. With Sequential versioning, there is always one last version, and each version has 0..1 predecessors.

Sample Versioning Workflow

OBJECT_VERSION_ID	**change_type**	**version assigner 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

The first ID is assigned by the server. The second ID comes in the request for amendment as the preceding ID. The third ID comes in the request for modification as the preceding ID.

2.3.8. Change Types and Lifecycle States

Supported Change Types

Change Type	**Notes**
creation	a COMPOSITION is created in an EHR
amendment	handled exactly like a **modification**
modification	an existing COMPOSITION is modified
deleted	an existing COMPOSITION is marked as deleted

Note: we don't have use cases for the rest of the change types.

Note: the EHRServer doesn't support undeleting COMPOSITIONs.

Supported Lifecycle States

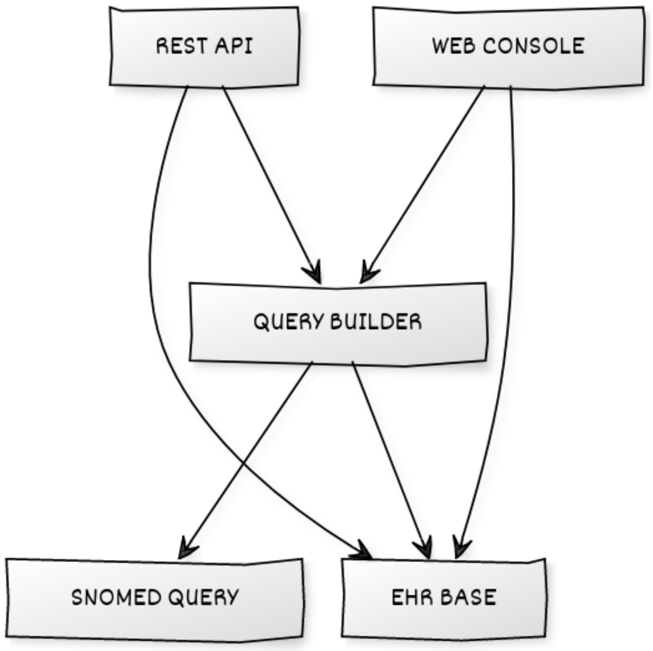
Lifecycle State	**Notes**
complete	used to store a complete COMPOSITION
incomplete	used to store an incomplete COMPOSITION

deleted	used when deleting an existing COMPOSITION

Note: the EHRServer verifies the right combination of Change Type and Lifecycle State is used on each COMPOSITION commit and returns an error if there is a misuse of those codes.

2.3.9. High Level Architecture

This section defines the product's architecture like the Software Provider sees it.



REST API

EHRServer offers two APIs, one created before there was an official openEHR REST API Specification available and one compliant with the openEHR REST API spec.

Web Console

Administrative user interface that comes natively integrated into EHRServer (there is no need for an external management application).

Query Builder

This component is part of the Web Console, and allows users to create and test\ queries based on archetype and template paths, using a formalism called Simple Archetype Query Model (SAQM) which is simpler than AQL.

EHR Base

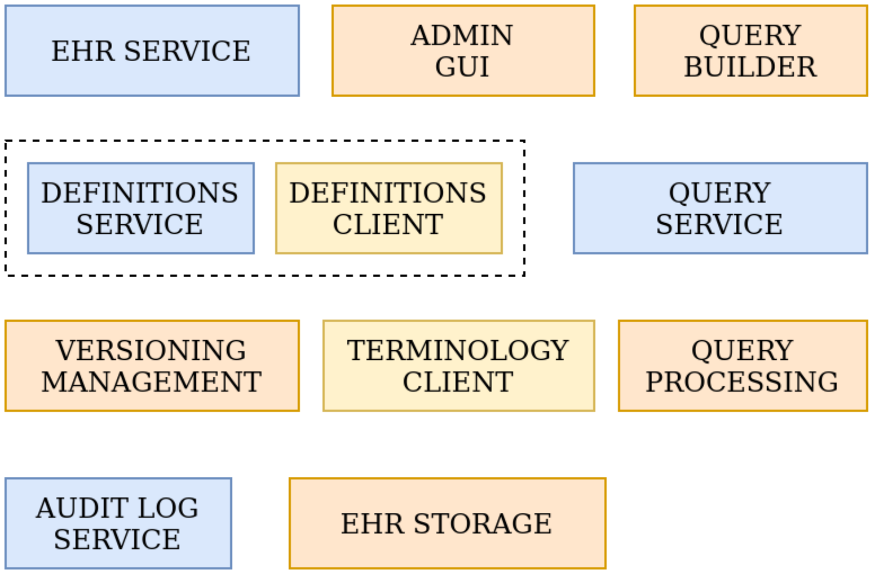
Handles all the data logic, including query evaluation, data access, audit logs and version control for COMPOSITIONs.

SNOMED Query

Service integrated into the query evaluation, which allows to resolve/expand SNOMED CT Expressions used in SAQM queries to filter data by SNOMED codes based on complex terminological conditions.

2.3.10. Reference Architecture

This section defines the product's architecture based on Service Model and Virtual EHR components.



3. Reference Architectures

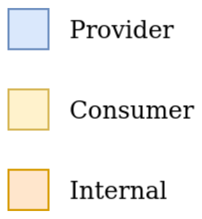
This section follows what was mentioned on section 1.3.5. defining a Reference Architecture for each Type of Product defined on section 1.2.1. A Reference Architecture is defined by components that play different roles in the Product. The term “Reference” is used because these architectures are not exhaustive and might not represent 100% how each openEHR Product is designed, but a Product should have a conceptual correspondence to the architectures mentioned below.

3.1. Reference for Reference Architecture Components

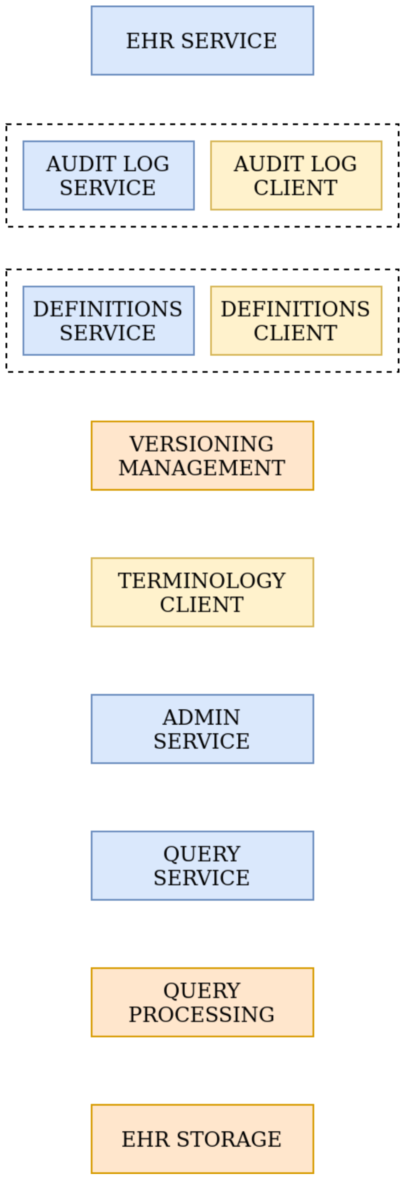
The blue boxes represent components that provide a service to other components, systems and products.

The yellow boxes represent components that consume a service from other components, systems and products.

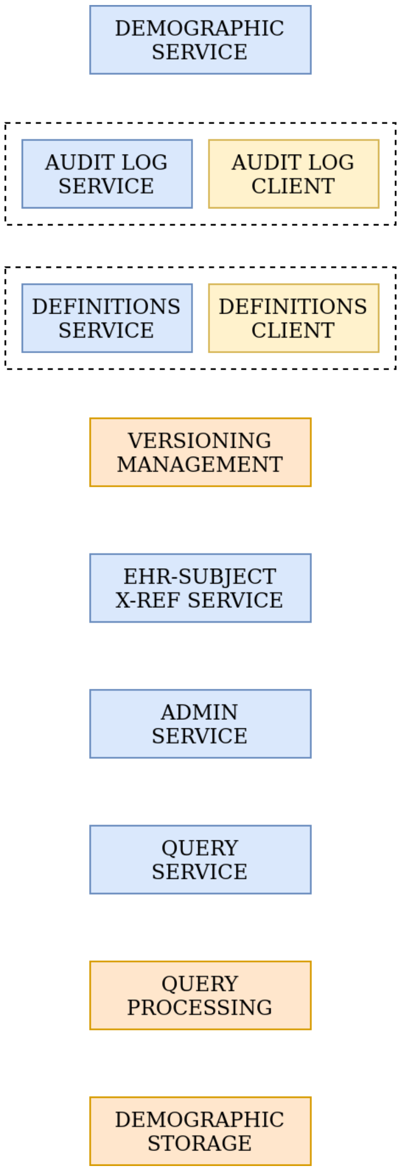
The orange boxes represent components that are internal for the product.



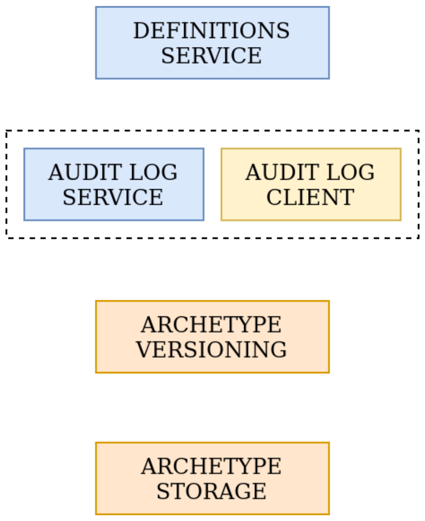
3.2. Clinical Data Repository



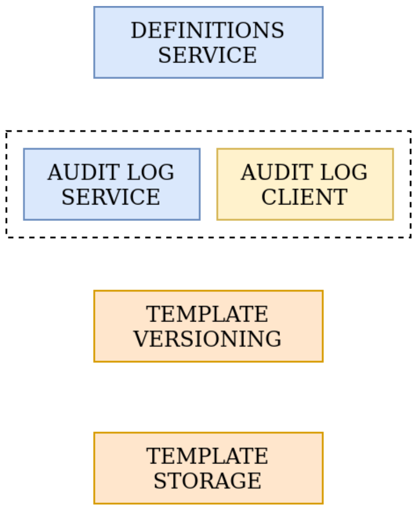
3.3. Demographic Data Repository



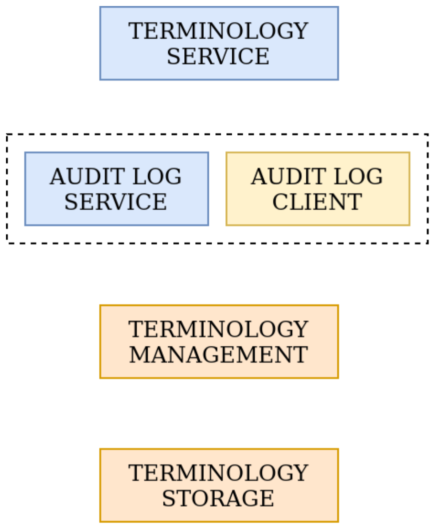
3.4. Archetype Manager



3.5. Template Manager



3.6. Terminology Manager



Note: some use cases to be considered by the Terminology Service and the Terminology Management are:

- return part of one terminology
  - a "part" could be a domain, a chapter, a subset, etc.
  - this includes codes, rubrics and definitions if available
- retrieve descendants, given one terminology item
  - this applies only for hierarchical terminologies
  - a terminology item could be given by any of it's identifiers in the correspondent terminology
- retrieve ancestors, given one terminology item
  - this applies only for hierarchical terminologies
  - a terminology item could be given by any of it's identifiers in the correspondent terminology
- offer suggestions based on given text
  - text might be partial, complete or acronym
  - current language is needed as context
  - part of the terminology might be required to give context and constraint results
  - queried terminology is required
- retrieve mappings of terminology items between different terminologies
  - given a terminology term, it's terminology and the target terminology, retrieve the matching terminology items from the target terminology

3.7. Rule Manager

TBD

3.8. Process Manager

TBD

3.9. Form Manager

TBD

3.10. Reference Implementation

TBD

3.11. Library

TBD

3.12. SDK

TBD

3.13. Integration / Middleware

TBD

3.14. Testing / Verification

TBD

3.15. User Facing Application

TBD

3.16. Archetype Editor

TBD

3.17. Template Editor

TBD

3.18. Form Editor

TBD

3.19. Rule Editor

TBD

3.20. Process Editor

TBD

3.21. Terminology Editor

TBD

3.22. Rule Runtime

TBD

4. Conformance Test Report Specification

This section follows what was introduced in section 1.3.6

4.1. Product Metadata Section

Similar to what was defined for the Conformance Statement Document (see section 2.2.1.), the Conformance Test Reports should include information about the SUT (name, version and type of product), and information about the versions of the specifications implemented in the SUT (e.g. EHR IM 1.0.2). This information should be contained in a standardized structure so it can be used to process, search and filter the report and between reports. As a first approach we will use a JSON structure to represent the report, later we will express it as an UML model and JSON will be just one possible serialization format for that model.

```
{
  “product”: {
    “name”: “EHRServer”,              // string
    “description”: “openEHR Clinical Data Management and Sharing Platform”,
    “website”: “https://cloudehrserver.com/”,
    “type”: “cdr”,                    // code
    “version”: “2.3”,                 // semver
    “implements”: {                   // conformance declaration
      “ehr_im”: “1.0.2”,
      “common_im”: “1.0.2”,
      “data_strucures”: “1.0.2”,
      “data_types”: “1.0.2”,
      “support_im”: “1.0.2”,
      “query_formalism”: “SAQM”,
      “query_version”: “0.1”,
      “opt”: “1.4”,
      “rest_ehr_api”: “1.0.1”,
      “rest_definitions_api”: “1.0.1”,
      “rest_query_api”: “1.0.1”
    },
    “developer”: {
      “company”: “CaboLabs”,
      “contact”: {
        “name”: “Pablo Pazos”,
        “email”: “pablo.pazos@cabolabs.com”
      }
    }
  }
}
```

4.2. Detailed Conformance Test Report

The Detailed Conformance Test Report should be like this table:

Test Suite	Test Case	Data Set	Expected	Actual	Result
A	A.1.	A.1.a.	one EHR exists	one EHR exists	accepted
A	A.1.	A.1.b.	one EHR exists	zero EHR exists	rejected
A	A.2.	A.2.a.	one COMPOSITION exists	one COMPOSITION exists	accepted
…	…	…	…	…	…

As mentioned on section 1.3.6., each Test Suite, Test Case and Data Set should be named or identified in a unique way, so the report shows unambiguous information, and the combination of those three items is what makes one result (one row in the table above). This can also be represented as a JSON structure in order to process, extract information and compare reports automatically.

```
{
  “product”: {
     …
  },
  “results”: [
    {
      “test_suite”: “A”,
      “test_case”: “A.1.”,
      “data_set”: “A.1.a.”,
      “expected”: “one EHR exists”,
      “actual”: “one EHR exists”,
      “result”: “accepted”
    },
    {
      “test_suite”: “A”,
      “test_case”: “A.1”,
      “data_set”: “A.1.b.”,
      “expected”: “one EHR exists”,
      “actual”: “zero EHR exists”,
      “result”: “rejected”
    },
    …
  ]
}
```

4.3. Summarized Conformance Test Report

As mentioned in section 1.3.6., the summarized view of the report should contain the following indicators, which are calculated from the data contained in the detailed report.

Total Test Suites	10
Total Test Cases	120
Total Data Sets	985
Total Accepted Results	735
Total Rejected Results	263

Note:Total Results (Accepted + Rejected) >= Total Data Sets, because each result is evaluated for one Test Case and Data Set, and there could be different Test Cases using the same Data Set.

A matching JSON structure would be:

```
{
  “product”: {
    …
  },
  “results”: [
    …
  ],
  “summary”: {
    “total_test_suites”: 10,
    “total_test_cases”: 120,
    “total_data_sets”: 985,
    “total_accepted_results”: 735,
    “total_rejected_results”: 263
  }
}
```

4.4. Hierarchical Conformance Test Result

This report is the combination of the detailed and summarized views. Following we define a structure that could be drilled down by a user:

- Total Test Suites: 10
    - Test Suite A
        - Total Test Cases: 15
            - Test Case A.1.
                - Total Data Sets: 5
                    - Data Set A.1.a.
                        - Expected: one EHR exists
                        - Actual: one EHR exists
                        - Result: accepted
                    - Data Set A.1.b.
                        - …
                    - …
            - Test Case A.2.
                - Total Data Sets: 58
                    - Data Set A.2.a.
                        - …
                    - Data Set A.2.b.
                        - …
                    - …
            - …
    - Test Suite B
        - Total Test Cases: 13
            - Test Case B.1.
                - …
            - Test Case B.2.
                - …
            - …
    - …

This view could be explicitly represented in the JSON report or could be calculated on the fly from the information contained in the JSON structure for the detailed view.