Introduction

Table of contents

1 Introduction - Reader's guidelines

This document will frequently refer to the Appendix 2.1.1.2 Requirements Specification – Tender area 1: openEHR-based Software.

This document Appendix 2.1.1.1 Introduction - Tender area 1: openEHR-based software describes the Karolinska University Hospital's intention to develop a Digital Health Platform (DHP), provides an overview of the reference architecture and describes the *logical components* that are necessary parts of the platform. The document aims to explain the context for the requirements in the Appendix 2.1.1.2 Requirements Specification – Tender area 1: openEHR-based Software. Certain information headings are specifically referring to this document.

The Appendix 2.1.1.2 Requirements Specification – Tender area 1: openEHR-based Software is a plain list of requirements that is expected from the offered Supplier's Solution, either to be fulfilled when signing the framework agreement or as part of future call-offs.

In the Appendix 2.1.1.2 Requirements Specification – Tender area 1: openEHR-based Software there has deliberately been made a distinction between requirements assigned to the Supplier's Solution and logical components (e.g CDR, PMI, etc.).

- Requirements referring to the Supplier's Solution are intended to specify general expectations on the offered Solution as a whole.
- Requirements referring to logical components are there to specify expectations on certain parts of the Supplier's Solution. These logical components act as handles to which the requirement is attached and may or may not be realized as physical components.

2 Karolinska's ambition

Karolinska University hospital (from now on "Karolinska") has begun the development of an open and standardized Digital Health Platform to create more and new opportunities for data-driven care and research. Karolinska's Digital Health Platform will collect, harmonize, and make available data from applications built on the platform and from source systems integrated with the platform. The platform, and the associated applications will aim to increase efficiency and productivity, reduce administration, raise quality, and provide better support for research.

Karolinska has made a strategic choice to base parts of the Digital Health Platform on openEHR. We believe openEHR has an important role to play in the strategy to reach the following goals:

- Ability to innovate and develop - Faster adaptation of IT systems to meet the constantly changing and developing healthcare, including a more efficient development process.
- Improved governance - Increased control of stored health record data and increased reuse of information structures within and between applications.
- Efficiency - Increased freedom of action by storing data in a vendor neutral and open format.

3 Scope of Procurement

Karolinska is now reaching out to suppliers to establish a framework agreement regarding delivery of parts of the Digital Health Platform. The scope of the procurement is divided into three categories:

- Category 1: openEHR-based Software – Information about scope of procurement is described in this document.

- Category 2: Software for openEHR Content Creation and Transformations – Information about scope of procurement is found in the procurement system TendSign.
- Category 3: Consulting Services – Information about scope of procurement is found in TendSign.

Category 1: openEHR-based Software is described in the following sections.

We see two potential approaches for the operation of the software provided by the supplier:

1. Packaged Software Delivery with on-premise installation at Karolinska:

   The supplier delivers packaged software along with comprehensive documentation and provide 3rd line support. The software shall be compatible with Karolinska's existing technology stack at delivery as described in document from procurementstep 1, Appendix 4 Karolinskas IT-landscape. This alternative is a "shall" requirement in Appendix 2.1.1.2 Requirements Specification – Tender area 1: openEHR-based Software in order to always be an available alternative when doing call-off from the framework agreement.

2. Software as a Service (SaaS):

   The proposed solution should be capable of being delivered as Software as a Service (SaaS) within EU/EEA including life cycle management. This entails the software being hosted and maintained by the supplier, allowing Karolinska to access and utilize the application(s) through a secure connection. This alternative is a "should" requirement in Appendix 2.1.1.2 Requirements Specification – Tender area 1: openEHR-based Software.

4 Motivation and summary of Category 1: openEHR-based Software

As illustrated in the openEHR e overview[1], a functioning openEHR-based eco-system is dependent on several other platform services, not only a clinical data repository. In this category we seek best-of-breed solutions that help us realize this architectural vision into a working platform.

4.1 Software and services relevant to purchase from a vendor in Category 1: openEHR-based Software

The following is included in this framework agreement:

- Software for storing and managing openEHR-based data
  - Clinical Data Repository (CDR) component
  - Patient Master Index (PMI) component
  - Distributed Transaction Manager (DTM) component
  - Auxiliary and administrative tools for the three above components
  - Software for fine-grained access control to be able to comply with Swedish healthcare regulations and other access control needs.
- Software for developing and publishing openEHR-based applications.
  - Software tools for development, publication, and maintenance of openEHR-based applications
- Software services

[1] Architecture Overview (openehr.org)

- Services to support activities within the software development life cycle (installation, design, implementation, testing, operations etc).

4.2 Motivation of bundling software and services in Category 1: openEHR-based Software into a single procurement category

The Swedish openEHR RFI 2023 indicated that many of the tools and functions described in relevant sub-areas are often offered in bundles and thus suitable to procure together. Even if form- and query-tools for developing and publishing openEHR-based applications may be supplied by separate system providers and run independently of procured CDR(s) and access control solutions, we at this stage want the main contractors to be responsible for the compatibility and integration of at least one offered combination of all important tools, CDR, and access control. We ask you to use standardized APIs so that components (including open source ones) from other providers can be mixed with the Tenderer's Solution. All provided Software must also have the possibility to include Support services with a well-defined target SLA – this is described in sub-area Software services and will be further specified in call-offs.

Further it is assumed that some needs may arise that are better covered by tools not offered in category 1 Software for storing and managing openEHR-based data. In such cases additional tools, functions etc. may be procured also via category 2 (Software for openEHR content Creation and Transformation) separately described in Tendsign. Category 1: openEHR-based Software

This chapter contains a more detailed description of Category 1: openEHR-based Software.

4.3 Software for storing and managing openEHR-based data

This section aims at specifying the use cases and specific problems that Karolinska wants to solve with a Clinical Data Storage Solution implementing the openEHR specifications. The Solution will be a core integrated component in the Karolinska Digital Health Platform and must support all aspects of a highly available clinical IT system with low RPO and RTO. The Solution is expected to be modular where individual sub-components are loosely coupled and can be replaced with a reasonable effort.

There is a distinction made between the terms Clinical Data *Storage* referring to the conceptual abstraction of storing the data, and a Clinical Data *Repository* (CDR) referring to one or more technical data storage components which builds up the Clinical Data Storage.

Figure 1 shows our simplified view in ArchiMate[2] notation of the surrounding services that together realize a functioning openEHR-based system for storing and managing data.

[2] The ArchiMate® Enterprise Architecture Modeling Language | opengroup.org

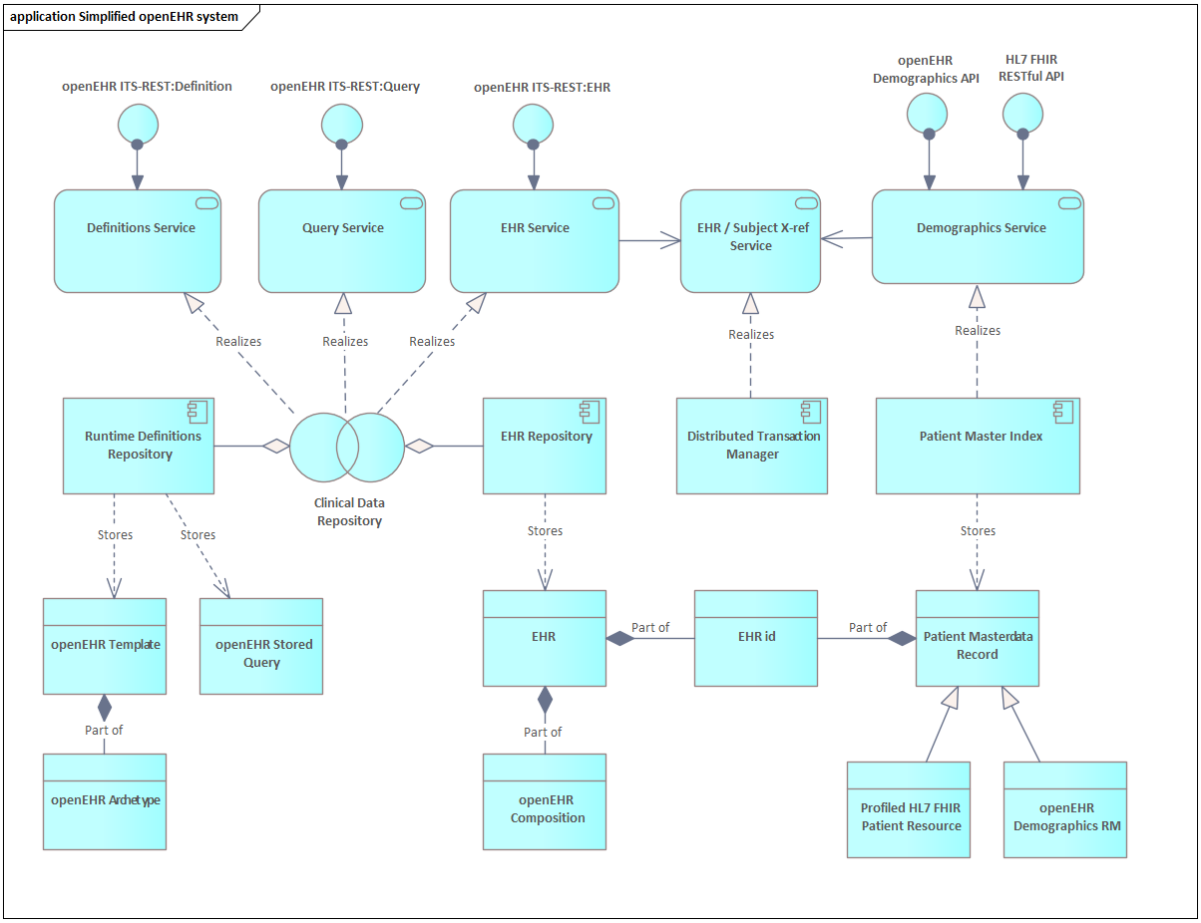

Figure 1 A simplified and opinionated view of a typical openEHR technology landscape, in ArchiMate notation. The standardized openEHR service and interface models realize a CDR where Compositions are stored and queried.

This architecture supports the loosely coupled demographics model of openEHR by only storing directly identifiable patient identifiers in a Patient Master Index component (PMI). The integration between the PMI and the openEHR Services is implemented using standardized APIs for increased modularity. In this picture we use HL7 FHIR as API, but we could also use a future openEHR-based demographics service model as it matures. The PMI maintains a reference from the patient master data record through a shared EHR id to the corresponding EHR record in the CDR. An EHR/Subject X-ref (crossreference) Service maintains consistency between the two repositories using a distributed transaction mechanism like IHE PIX[3]. EHR records are created through this service by coordinating usage of the EHR and Demographics service API:s.

In this sub-area we seek Software components that realize the services and API:s described in Figure 1. For increased modularity and minimization of vendor lock-in, the Software should be treated as three separate logical components, with well-defined interfaces:

- “CDR” - Clinical Data Repository component
- “Patient Master Index” - (PMI) component
- “Distributed Transaction Manager”- (DTM) component

[3] IHE ITI TF Vol1

It is essential that all components needed for the solution work together to form a functioning openEHR eco-system within Category 1: openEHR-based Software.

For the procured openEHR Solution (CDR and PMI components) to support certain non-functional requirements of Clinical Data Storage systems, the offered Solution should have administrative support tools, documentation and documented routines supporting (but not limited to):

- Bulk operations
  - Import and Export
- Business continuity solutions
- Data retention policies (purging/thinning/shredding)
- On premises installation and configuration
- Software as a Service delivery
- Life cycle management (PLM)
- OpenEHR content management
  - E.g. Publishing of templates, definitions etc.
- System monitoring, health checks and alerts
- Component IAM: Support for fine-grained access control with integration to an existing Identity Provider (IDP)
  - Integrations and service accounts
  - Administrative interface
- Logical and physical deletion of Compositions and EHRs

4.3.1 Clinical Data Repository (CDR) component

The Software must implement the openEHR specifications[4]. There is an expectation that the offered solution must comply with the openEHR releases stated as "stable state"[5] e.g., the openEHR ITS domain, as well as having a thought-out process for handling and contributing to the openEHR specification components stated both as "stable state" and "development state", e.g., openEHR Conformance Specifications (CNF) Component. See Appendix 2.1.1.2 Requirements Specification – Tender area 1: openEHR-based Software for detailed requirements and requested versions, that have been selected in order to balance cutting edge against achieving maximum competition of available products in the framework agreement.

To exemplify further, this means that the procured Clinical Data Storage Solution will consist of one or more underlying OpenEHR CDRs with full support of (including, but not limited to):

- openEHR RM+AM and related specifications
- REST APIS such as
  - openEHR EHR API
  - openEHR Query API
  - openEHR Definitions API

It is also of importance that the implementation, database design and supporting technology used ensure that Karolinska can scale the implemented Solution to a vast number of concurrent users and

[4] https://specifications.openehr.org/
[5] https://www.openehr.org/programs/specification/changeprocess

database accesses, without neither losing data integrity nor creating system bottlenecks due to high end-to-end data latency.

4.3.2 Patient Master Index (PMI) component

The procured Solution must implement an openEHR CDR according to the openEHR information model[6] where demographic data is separated from health records ("pseudonymization"). Consequently, there is a need for a component where demographic data is stored, along with references to (i.e., IDs for) related EHRs.

For the procured Solution to be compatible and interoperable with related components in the Karolinska environments, the PMI and related components are expected to support (but not limited to):

- Relevant subset of HL7 FHIR APIs, i.e., the subset supporting the Patient resource
- Bulk import using HL7 FHIR formats
- Bulk export HL7 FHIR format
- Ability to use Karolinska specific FHIR profiles and extensions

The PMI can also be used in a more general way, supporting patient merging and more enterprise features as defined in IHE PMIR[7] or similar.

4.3.3 EHR/Subject X-ref Service and Distributed Transaction Manager (DTM)

In the openEHR architectural model as illustrated in Figure 1, EHR-ids are linked to patient identities in the PMI component. It is critical that these links are correctly attributed to the patient. Hence, we need a robust system which help us maintain strongly consistent links as these are established/created and updated.

This service manages the distributed transaction between the PMI and the CDR, including handling transient failures gracefully. It presents a high-level transactional API to the developer that hides the underlying complexity to allow for safe creation of EHRs that always are linked to the correct PMI record. The design of the API is not standardized but up the Tenderer.

The service can be realized by Distributed Transaction Manager component using a SAGA pattern[8] or similar mechanism. Having such a centralized transaction manager reduces the need for many different applications and integration flows to themselves having to (re)implement correct transactional semantics regarding linked EHR and Patient creation. It's important that the Distributed Transaction Manager is only using standardized API: s of the CDR and PMI. Any proprietary side-effects should be avoided.

4.4 Software for fine-grained Access Control

In this sub-area we seek additional functionality to be able to comply with Swedish healthcare regulations and other access control needs. We are looking for solutions that can be configured in flexible ways rather than having to be developed as separate software components.

The openEHR specifications do not specify how to implement access control; it is left to the implementor to provide this functionality. To maximize developer productivity and simplify

[6] https://specifications.openehr.org/releases/RM/latest/ehr.html#_demographic_data_in_the_ehr
[7] IHE.ITI.PMIR\1:49 Patient Master Identity Registry (PMIR) Profile - FHIR v4.0.1
[8] Sagas (microservices.io)

compliance, we want to externalize common functionality like access control from individual applications to common platform services.

These standardized platform services are necessary to provide some basic level of access control. However, for compliance with Swedish healthcare regulations the openEHR services must be extended with functionality to provide more fine-grained access control than solutions only operating at the HTTP protocol layer. It is not sufficient to only govern access to HTTP methods and resources as the openEHR service model also include services for querying. We need to understand what the query is doing and be able to control the resulting replies.

Key to delivering this functionality is the ability to filter outgoing Compositions and query responses. Before a composition or query response is returned to the user, we want to have the ability to remove, rewrite and/or redact it. The CDR Policy engine should be able to use associated metadata from trusted sources . The type of metadata needed to be managed can be divided into two categories: *ownership and access control* and *information classification*, see Figure 2.

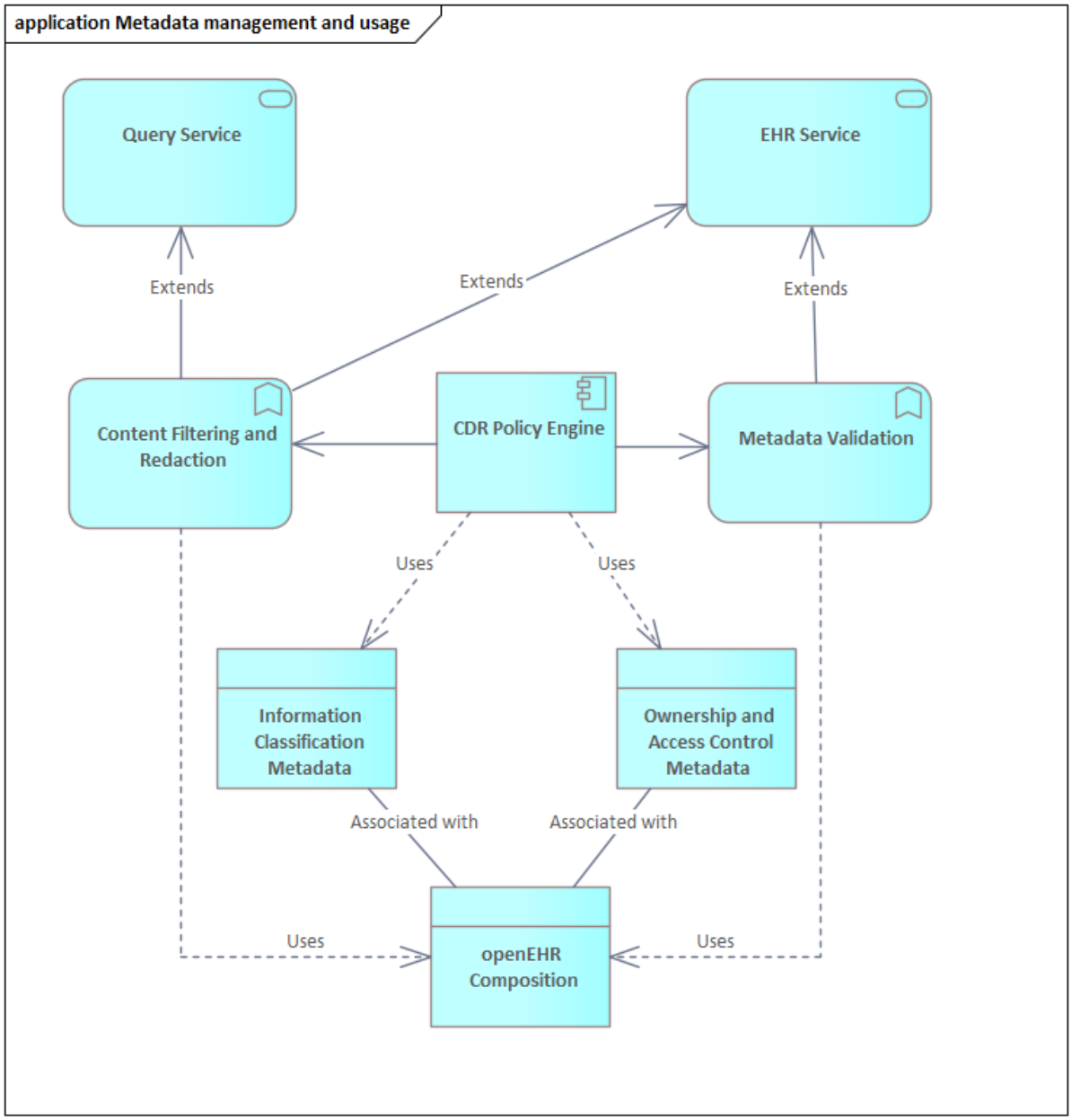

Figure 2 Two types of metadata are associated with openEHR content (content is exemplified by COMPOSITION in the figures but can be other objects accessible via dedicated APIs or AQL queries). This metadata is used by two additional functions governed by a policy engine for regulatory compliance.

An incoming HTTP request will contain various user claims which the authenticated user presents. These claims are typically part of an OAuth2 access token contained in the request. The CDR policy engine should support filtering and validation functions by using checking the user claims against the metadata associated with the Compositions.

As the metadata is the source of truth for authorizations, the Solution should include a metadata validation function that validates the metadata before a composition is written to the CDR. As an example, in Swedish health care, Compositions will be owned by a health care organization. The organizational masterdata (type of metadata) must be checked so that the organization exists before the Composition can be stored.

This functionality can be configurable as part of an existing CDR implementation or implemented as an independent component that integrates with any openEHR compliant CDR.

4.4.1 Components for fine-grained metadata-based access control

The purpose of this chapter is to describe the functions and components resulting from Karolinska's interpretation of the applicable laws and regulations. Figure 3 shows a more detailed overview of the logical components and services needed to augment a standard openEHR-based system to comply with regulations. The different components and their behaviors are described in more detail in the following sections.

These components and their exact behavior will be part of a joint development/integration effort in later stages in the procurement. The scope in this chapter is for the Supplier to understand the complexity of this delivery.

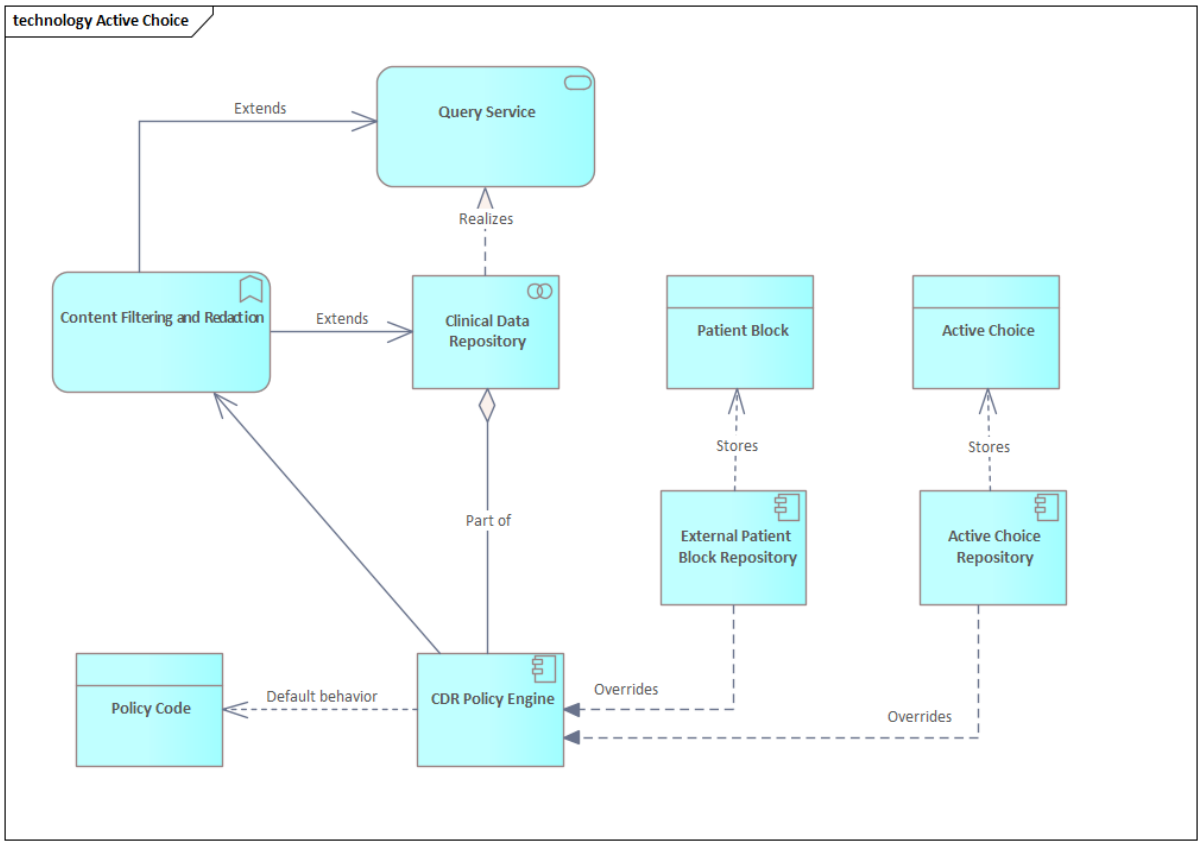

Figure 3 describes the default behavior of the CDR policy engine to the left and the Active Choice Repository and Externa Patient Block Repository to the right.

The components active choice repository and external patient block in Figure 3 can be viewed as logically separated. There are relations between the information in the active choice repository and the external patient block repository, for example can an active choice be used to temporarily unblock a patient's block. An active choice made by a user is only valid for the user who did the active choice and can be valid for a certain period, which is why the active choice needs to be persisted.

The information in the active choice repository and the external patient block repository needs to be used when evaluating which compositions, a user is allowed to access, hence it overrides the default rules.

4.4.1.1 Content Filtering and Redaction

This function shall be able to filter the information leaving the CDR, based on authorization attributes from both static, request and external sources. This function is realized by a CDR policy engine where the Customer can configure access control policies to comply with regulatory requirements. The policy engine should come with an open specification and administrative interfaces (API or file-based). External sources, such as patient blocks and/or active choices can be used to provide runtime overrides to default policy behavior.

4.4.1.2 Active choice repository (and service)

In the applications connected to the CDR, various types of active choices will be made, and certain active choices will remain valid for a specific period. Hence the user will not be required to make the same active choice repeatedly during the valid timeframe.

In the Solution, various types of active choices shall be able to be configured, so the active choices related to the CDR can be uniformed in what information that can or are required to store.

4.4.1.3 Patient block repository (and service)

A patient shall be able to block information in their EHR. Blocking should be possible per care unit/PDL care process. This means that a user from PDL care unit/PDL care process A in the solution shall have no default access to compositions written from the blocked PDL care unit/PDL care process B, but with a valid active choice a user shall be able to provide access to compositions written by PDL care unit/PDL care process B.

The solution needs to integrate with an external source to retrieve information about patient blocks.

4.4.1.4 Identity Access Management

The openEHR specification does not specify in detail how to implement access control. As the service APIs are all exposed using the HTTP protocol it makes sense to adopt relevant web-standards for a modular and open realization.

4.5 Software for rapid development, publication, and maintenance of openEHR-based applications

This sub-area covers important tools and resources that are needed to configure and get an openEHR CDR-based system up and running with end user entry forms, dashboards etc. This should be done in a way so that the combination of software described in previous sections can be used in daily work by e.g., clinicians and informaticians. I.e., it must be possible to configure the user interface and content (templates, forms, simple form logic) by informaticians and super users that are not software developers.

To extract maximum value of an openEHR-based eco-system we want to procure tools for building applications. The openEHR formalism lends itself naturally to generative and low code approaches. Simple application components such as forms can be completely or in parts generated from a template specification by informaticians and subject matter experts who are not software developers.

The forms-based applications can be statically generated or rendered at runtime from an intermediary specification, see Figure . The form builder service typically include support from a

terminology service and a library of snippets or widgets to help developers bind terminology excerpts to form input fields.

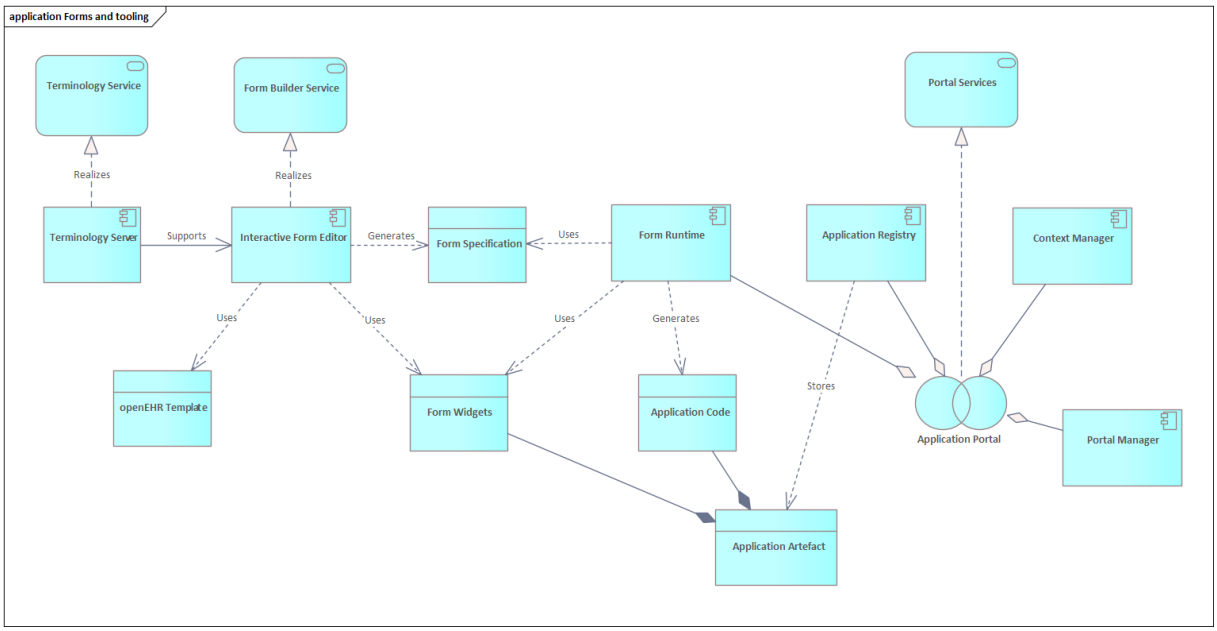

Figure 4 Simplified example architecture for a form builder service and its context. openEHR templates are used by an interactive form editor to create a form definition that can provide input to a code generator (or be interpreted at runtime). Application artifacts are published to an application portal where the application can be operated and accessed. Execution of AQL queries will likely also be part of forms/applications, but are not shown in this simplified diagram.

The applications can then be published to a shared application portal service where the applications can be accessed and operated. Another requested possibility is to embed applications into existing systems like an existing EMR/EHR system or separate web applications.

Karolinska's vision is to be able to quickly build needed clinical functionality based on platform services and expose this functionality as applications in a cohesive and easy to use manner. For further ease of use, the application portal could also expose context management services to be able to maintain a given patient context across multiple application instances.

Main functions that likely will be procured in this area are (but not limited to):

- **Low-code (or similar) tools for creating and rendering UI entry forms, dashboards, other views, and overviews.** Some pluggable openly published architecture for additional widgets may also be requested, see e.g. discussion in https://discourse.openehr.org/t/standardised-api-for-custom-gui-widgets-for-openehr-based-form-editors-renderers/1936
- **AQL query building and execution tools.** A tool or tools are needed to create, manage, validate, and save AQL queries, including parametric queries. There is also the need to visualize and export AQL query responses.
- **End user facing portal functions.** In some cases, software developers will build separate applications that do not need a portal, but often simple applications built/configured using low code tools for data entry forms, dashboards, other views, and overviews will be enough. Such simple applications need a framework to run in to select patient, select view and look at existing data and/or adding new data.
- **Design language and pre-made configurable widgets.** A coherent set of UI design principles and widgets that can be used also in our own custom development in a way that can be seamlessly combined with low code generated parts of the UI.

During the Swedish 2023 RFI responses and demos, proprietary low-code UI building tools were presented. Since there is no standard for these, there is a risk of future lock-in effect regarding forms and applications if we would choose to change vendor for such functions later (requiring a lot of conversion work during short time).

4.5.1 Audit Log

Personal health information is confidential information and protecting its confidentiality is essential.

To protect the consistency of health information, it is important that its entire life cycle be fully auditable. The Solutions functionality shall be designed in a standardized manner, so the customer can be compliant with the following international and national standards[9]. The Audit log functionality in the Solution is expected to adhere to the IHE ATNA profile[10] or equivalent.

The Solution needs to have functionality for the Secure applications in the Solution to detect defined trigger events, (all activity and transaction related events need to generate an Audit record). The Secure Application shall also report the events to the Audit Repository (where the Audit Records shall be stored for a shorter period) and from the Audit record, forwarded to the customers Audit record service, via Syslog TLS, FHIR Feed or FHIR messaging. The audit and analysis of the audit records will be done in the customers Audit record service.

It is also of importance that the content in the Audit record can be extended from the audit message schema, as more attributes need to be captured for the customer to be compliant with regulations and to be mapped to the customers FHIR AuditEvent profile.

[9] ISO/IEC 27002:2022 - Information security, cybersecurity and privacy protection — Information security controls, ISO 27789:2021 - Health informatics — Audit trails for electronic health records
[10] IHE ITI TF Vol1

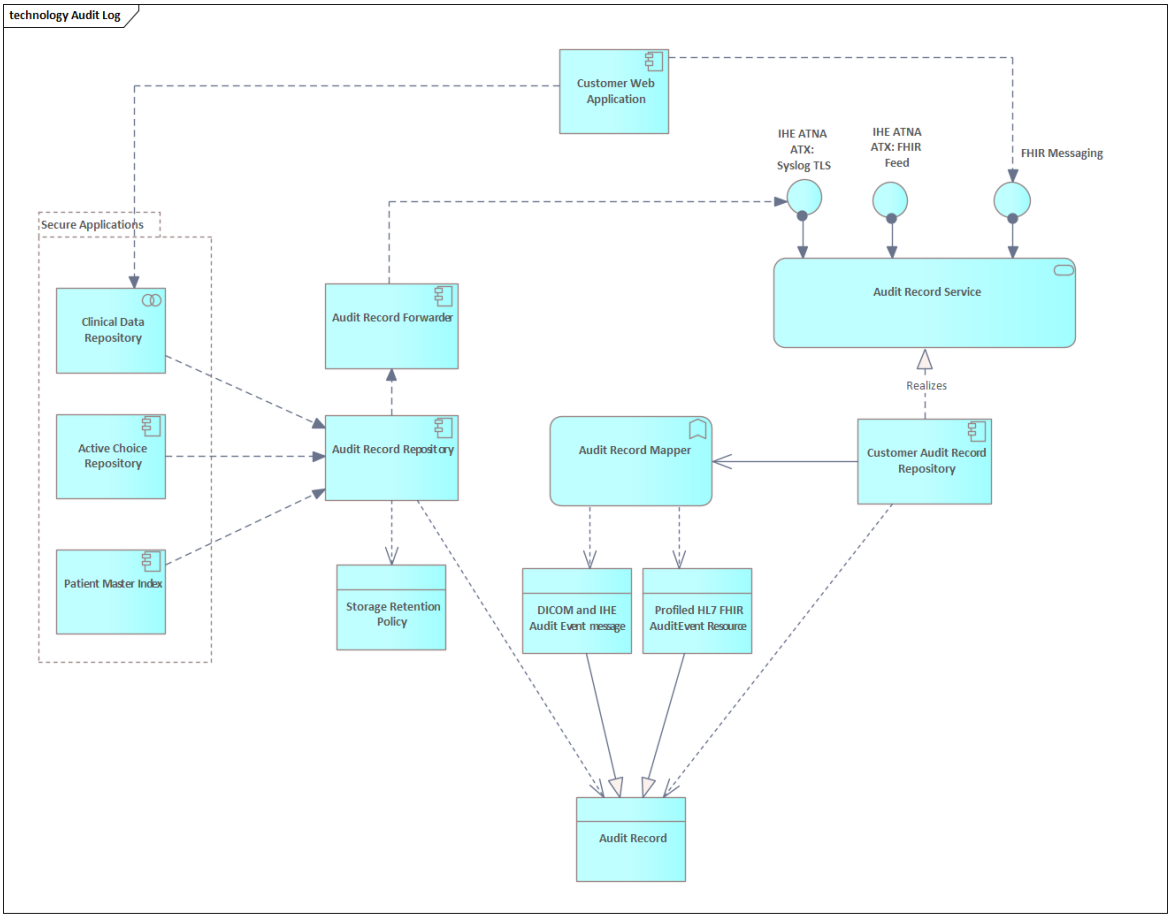

Figure 5 describes an example architecture for Audit Log, with the Secure Applications to the left, which shall detect and report audit events for all activity and transaction-related events. The Audit records are stored in the Audit Record Repository in the Solution and forwarded to Customers Audit Record Service, where they are mapped to the customers FHIR AuditEvent profile, if not received in the same profile.

We will expect solutions to support a soft exit strategy with continued use of *already created* forms/apps (and use of design language) for some years by contractual, technical, or other means. Requirements regarding this will likely be further detailed in later call-off stages.

4.6 Software services

In addition to the delivery of Software, as described above, we also need to purchase various types of services. Supplier shall provide Karolinska with services that include (but not limited to):

- Support and maintenance for the products the Supplier delivers, including life cycle management. This includes upgrades and updates to delivered Software. When the Solution is hosted on-premise or as Software as a Service the Supplier shall deliver support according to appendix ‘Support and maintenance with SLA’.
- Installation, implementation, and configuration services, including acceptance and performance testing services.
- In cases where the Supplier develops a software solution on behalf of Karolinska, there should be agreements on usage and ownership rights, support and maintenance, as well as compensation for these in individual cases. This is agreed upon during procurement.

- It should be possible to order Customer specific modifications and extensions to meet specific needs, including obtaining separate functionality in the product that the supplier develops and maintains.
- The Supplier should have customer forums where the roadmap for the product(s) and new functionality is discussed and agreed upon. Such new functionality becomes a part of the Suppliers product(s).
- The Supplier shall offer expert services, known as Subject Matter Experts, who can support Karolinska if there are questions about the product.
- The Supplier shall offer training. Training involves the process of educating and familiarizing users or administrators with the software products. This includes how to use, manage, and troubleshoot the products effectively. Training programs may cover various aspects such as system functionalities, updates, and best practices to optimize product utilization.
- Collaboration between the customer and supplier is crucial for ensuring the success of the partnership. This involves open communication, mutual understanding of goals and requirements, and a proactive approach to problem-solving. Effective collaboration enables the supplier to align their offerings with the customer's evolving needs and ensures a smooth implementation and support process. The Supplier and Karolinska shall in the call-off agree on how to systematically collaborate during the contract period.
- Documentation is essential for providing a comprehensive reference for the products and services delivered. It includes user manuals, technical specifications, implementation guides, and any other relevant information that facilitates proper understanding, utilization, and troubleshooting of the products. Clear and detailed documentation contributes to the efficient use and maintenance of the supplied solutions. The Supplier and Karolinska shall in the call-off agree on what documentation that needs to be delivered.

The Services will be detailed and decided in the call-off.