Appendix A: openEHR – an Implementors Guideline related to Swedish laws and regulations in healthcare

1 Introduction

This document provides a short discussion of implementation needs and context for those already familiar with openEHR, HL7 FHIR, OAuth2, REST, authentication, and access control mechanisms. This document reflects our level of ambition and different solutions are discussed. Please feel free to be inspired by this document, we also look forward to receiving alternative solutions and discussion.

In 2022, a collaboration of several Swedish Healthcare providers and EHR system suppliers, coordinated by openEHR Sverige, has suggested a draft version of a shared standardized way of storing access control related metadata in openEHR systems. This document is partly based on those agreements but also explains other needs and requirements expressed by organizations backing a coordinated openEHR-RFI in 2023.

2 Aspects of Laws, Regulations and their Implementations

Described below are some essentials for fulfilling the laws and regulations in the Swedish healthcare system. The focus is to maintain the patient's privacy by controlling access to the patient's personal data. The term "personal data" is used throughout this document to describe every piece of information related to a specific patient kept by a healthcare organization. An "information domain" is used to describe groupings of personal data based on sensitivity, type of information and/or clinical terminology and use.

2.1 Metadata

In this document we identify and describe various types of metadata needed for managing access to and governance of personal data. Examples of metadata are:

- Patient identifiers
- Domain or type classification of personal data
- Organizational ownership
- The source/lineage of personal data i.e., how the data was entered and/or imported
- General descriptive metadata such as timestamps
- Status information (e.g., signed/unsigned, blocked/not blocked)

Metadata needs to be managed in a flexible, scalable, and automated way. We need support for hierarchical models to manage metadata related to organizations and domains and methods of personal data bulk updates based on filters. We need to be able to create new

types of metadata and to choose which metadata to link to each specific personal data. Metadata from different source systems should accompany corresponding data in the CDR.

2.2 Use of Metadata

We anticipate at least the following possible use of metadata to support legal compliance:

1. Maintaining correctness of metadata, i.e., the ability to validate incoming COMPOSITIONs[1] according to the metadata model used
2. Use metadata to mask or redact (filter) response data passed through the openEHR REST-APIs. We need a way to control either the queries or the response data to make sure user can only see data they are authorized to see
3. A way to dynamically add or override static metadata to support temporary escalation of privileges. Additional metadata can be retrieved from a combination of sources such as the active session/context, integrations with external services such as EHR/EMR system, consent and/or block services.
4. Maintain a consistent inverse index[2] mapping organizational units to COMPOSITIONSs within each EHR record. This index is needed to provide data for informing a practitioner where there are COMPOSITIONs, regarding a selected patient, but owned by other organizational units (and thus not visible to the practitioner by default).

3 Access Control Models

A healthcare professional can be assigned multiple roles based on his/her assignment, profession, and employment. These real-world roles can be modeled in RBAC or ABAC type authorization systems where roles, attributes, and permissions are managed outside of the CDR solution. This type of metadata can be made available to a policy engine dynamically at each request or from static sources. In the dynamic case metadata can come from trusted sources in the form of access tokens or similar and/or from external APIs.

Access control policies should be able to consider many different forms of metadata and these policies should be flexible and configurable to match legal requirements. Policies can be maintained using a combination of external/internal rule engines, domain specific languages and/or configuration files.

4 Multi-tenancy and Overall Solution Architecture

The Swedish healthcare model is inherently multi-tenant as information is segmented by organization ownership whilst in some situations allowing some users access across several

[1] Of course, we have the same need for FOLDERs and other EHR content, but to make the text easier to read we only write COMPOSITION(s) in the rest of the document

[2] If ownership metadata is efficiently queryable from the CDR (via e.g., AQL) across tenants, then indexing built into the CDR may of course be used.

different tenants (healthcare providers and care units). From an architectural perspective this leads to at least two fundamentally different system design options:

1. Multiple physically segmented instances, i.e., one instance per organization with information logically accessible through federation
2. Single physical instance, i.e., one single instance with information logically segmented within and accessible from that single instance

In the first model we need to distinguish between the case of a common EHR space shared between the instances and the case of one private EHR space per instance. A shared EHR space will require co-operation between the tenants to a larger degree and the private EHR space per instance will likely require a master patient index service to be able to cross-reference patients[3].

The obvious drawback of the first model is that every organization needs to operate their own instance whereas in the second model one organization operates the system on behalf of the others reusing shared infrastructure. Governance might also be easier in the second model as it is easier to manage and govern access control, openEHR definitions and content in a single instance.

In the case of the single physical instance, all stored EHR content needs to include metadata or be annotated/linked in ways that makes it possible to determine what healthcare unit (and healthcare provider) it belongs to (the logical segmentation). In the case of the federated approach there needs to be an external registry where we can lookup organizational metadata and mappings to the different instances (addressing part of the federation).

The federated approach comes with its own set of problems[4] managing the federation and providing a logically coherent view on top of the physical instances[5]. Even in this case, it makes sense to logically segment information on the healthcare unit level. Otherwise, we would need to operate one CDR per healthcare unit which is unfeasible as large healthcare providers can have hundreds of healthcare units[6]. Hence, we focus on the logically segmented model in the rest of the document acknowledging that it can be combined with a federated approach on the healthcare provider level if needed.

[3] See for example IHE PIX: IHE ITI TF Vol1

[4] Consistency when updating, addressing and robustness, query performance, see Federated architecture - Wikipedia and Shard (database architecture) - Wikipedia

[5] One example of this is the Swedish national infrastructure for federated access to health data https://rivta.se

[6] Even if we could manage hundreds of CDRs, organizational structures change frequently, which will lead to significant problems re-distributing data

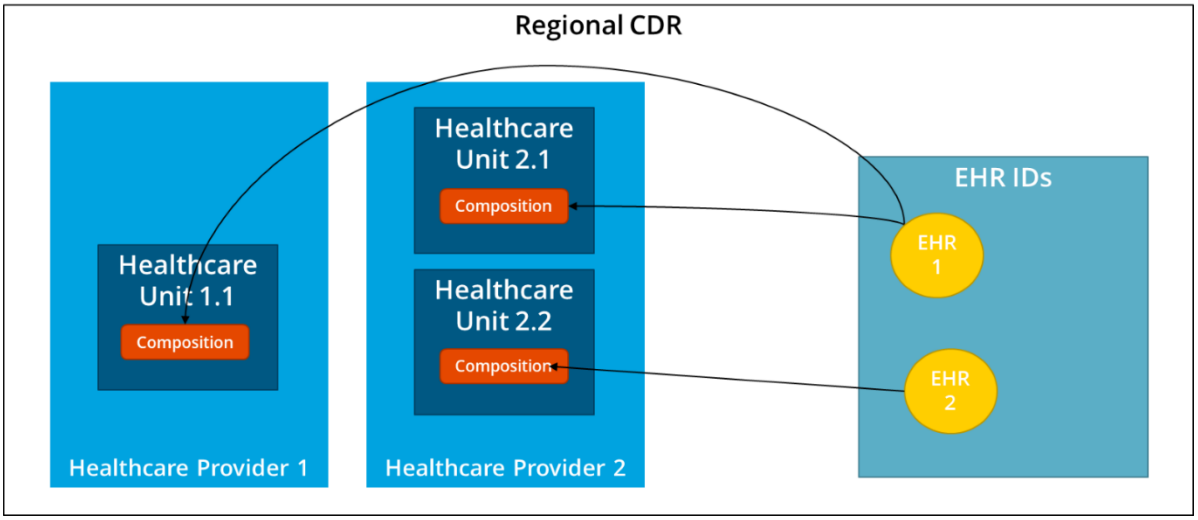

Figure 1 Basic logical model of a multi-tenant single-instance CDR with an additional cross-referencing EHR id record

In this multi-tenant single-instance setup, see Figure 1, we want to use an instance-global EHR (id) space representing all patients in a region[7]. Each EHR record will contain COMPOSITIONs owned by many different (logically segmented) organizations. Furthermore, COMPOSITIONs are in most cases owned by one healthcare unit/provider. To produce a static segmentation of information in a single instance we see at least 3 viable approaches:

1. A Swedish openEHR collaboration suggested a way to store such metadata inside COMPOSITION.context → EVENT_CONTEXT.other_context, using two nested instances of the Organisation archetype containing name and identifier of healthcare unit and healthcare provider. The lowest level of organizational unit could be stored in EVENT_CONTEXT.health_care_facility and can be expected to be shown in user interfaces.
2. Other openEHR-based options could be to annotate the compositions with Tags (e.g. with keys for healthcare unit and healthcare provider and values containing identifier strings) or to link to them from healthcare-unit-specific Folders (the folders' "details" attribute could be modelled using Organisation archetypes, as mentioned above in approach #1)
3. This metadata can also be stored in an external system (not in the openEHR RM) and integrated into the access control policy engine

[7] These EHR ids should be globally unique and cross-referenced in an external service to national patient identifiers such as the Swedish "personnummer". We don't believe it's feasible today to construct a national EHR id space given each region's sovereignty over patient data. As a result, each patient will have at least one EHR id in every region.

5 Access Control and Integrations into Regional IT-Environments

CDRs rarely operate on their own as they are usually part of a larger application infrastructure. In the Swedish healthcare system, many regions follow Inera ABs reference architecture for identity and access management[8].

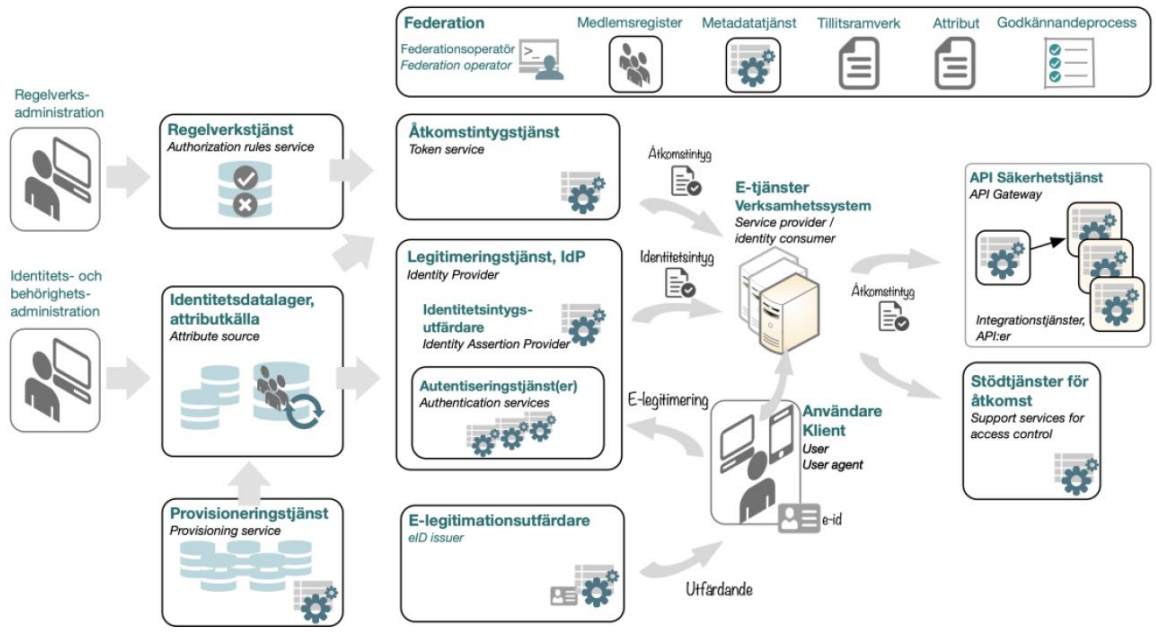

Figure 2 Overview of Inera ABs reference architecture for identity and access management

In this architecture, applications using the CDR will request access tokens to access the openEHR REST-APIs, often passing through an API gateway. The current recommendation is to use OAuth2 and OpenID Connect (OIDC) standards.

In an OAuth/OIDC setting, access control attributes and user metadata are usually communicated to the service using claims inside access and/or identity tokens. Access control policies can be enforced directly on the identity provider or externalized, using an API gateway pattern (reverse proxies) to a separate policy enforcement point.

As an example, we need to communicate filter/masking parameters from the user context to the CDR sub-system enforcing access control to be able to dynamically configure queries and/or to filter response data. These parameters could be taken off claims inside an access token to harmonize with OIDC/OAuth2 ideas. An abbreviated and simplified example flow could look like

1. The user loads a web application into his/her browser and is redirected to log in using an OIDC compliant IdP
2. The application will request the scopes it needs for the user to approve. After authentication the application can request access tokens to act on the user's behalf

[8] Referensarkitektur för Identitet och åtkomst (rivta.se)

3. The IdP might need customization to allow for selection of roles to specify the user intent for this session
4. The IdP is also configured to include custom claims[9] that support the access model. Claims are typically dependent on the intent specified in the previous step
5. The CDR, gateway, or sub-system responsible for access control will interpret and act on the provided claims to enforce policies

When an application temporarily needs to escalate privileges, a new updated token needs to be requested from the IdP. The UX components can be realized as extensions to the IdP. The IdP can request information from the CDR where in the organization tree there is personal data. This way, we do not need to build this type of functionality into every application and the IdP can cryptographically sign the tokens for increased security. Additional metadata could be part of the claim or looked up from an external system in the IdP or on the policy enforcement point. Access control policies can also be enhanced by considering encounters, bookings and/or other contextual information.

6 Audit Logging

All information access attempts, both successful and failed ones, need to be audit logged. Ideally, the system should provide a rich set of triggers (instrumentation) where we can configure delivery of the logs[10]. It's also important that audit logs are managed in a secure way as they themselves are sensitive data.

In large organizations, these logs are typically sent to an anomaly detection (SIEM) system to fully- or semi-automatically track practitioner behavior across systems. To be able to perform real-time analysis and/or forensic analysis we need to collect audit logs from all components in the application eco-system. Application logs show client-side rendering, IdP and API gateway logs, and CDR sub-system logs show actual data transfer between the client and the CDR.

To facilitate cross-system analysis it's also vital that the audit logs are harmonized and follow international and/or local standards[11]. Furthermore, it's also important that the audit logs are self-contained meaning that they cannot contain references to subjects (such as EHR IDs, or organizational units). We cannot guarantee that the references to business identifiers are stable in case of a later forensic analysis based on historical data.

Regions may also have requirements to integrate into national infrastructure where audit logs must be able to be transformed to the StoreLog data model used in this service[12].

---

[9] An example of how these claims could look like in a Swedish healthcare context (HSA catalog) can be found in the Sambi federation attribute (claim) reference: Attributreferenser för Sambi | Sambi

[10] See "Health informatics – Audit trails for electronic health records (ISO 27789:2021)"

[11] See "Health informatics – Audit trails for electronic health records (ISO 27789:2021)", IHE ITI TF Vol1 or IHE.ITI.BALP\Basic Audit Log Patterns (BALP) - FHIR v4.0.1

[12] See https://rivta.se/tkview/#/domain/informationsecurity:auditing:log