Tender
5/9/2023

Procuring organization

Region Östergötland
Bernadett Brink

Procurement

RFI, Request for Information of openEHR platforms and related tools
RFI-2020-08:2
Version 2: published 4/18/2023 11:46 AM
Tender closing date: 5/10/2023 11:59 PM

Legend

▤ The text is included in the advert	☰ The text is included in the qualification
✍ The text will be part of the contract	🌍 The text will be published in the contract catalogue
❗ The text/question contains requirements to be met	ESPD The text/question contains ESPD requirements
① The question is weighted and included in the evaluation	⚖ The question is weighted and included in the evaluation
ⓘ The question is asked for information only	🔒 The question is answered by the buyer
✓ The question is marked for special follow-up	⚠ The answer does not meet the requirement in the question
⚠ Updated section or question	

Tenderers

Supplier	Tender	Corporate ID	Qual.
Medblocks	Medblocks Ignite	0007919	

Contents

1. Invitation to openEHR RFI and demo

1.1 Invitation to openEHR RFI and demo

Sydöstra sjukvårdsregionen (including Region Östergötland, Region Kalmar län and Region Jönköpings län), Västra Götalandsregionen, Region Uppsala, Region Stockholm, and Region Skåne hereby invites suppliers of openEHR platforms and related tools (in this document called "Solution") to a request for information and a product demonstration.

1.1.1 RFI process

This RFI process is divided into two (2) parts:

- The first part is open for all suppliers of openEHR solutions and consists of questions to be answered in written format, plus an appendix for context.
- The second part consists of an online product demonstration and is subject to specific qualification criteria. See Part 2: Demonstration sessions for details.

1.1.2 Date and time for demonstration sessions

The following time slots are available:

Date	Time (CEST/UTC+2)			
May 31	8:00-10:00 AM	10:00-12:00 AM	1:00-3:00 PM	3:00-5:00 PM
June 1	8:00-10:00 AM	10:00-12:00 AM	1:00-3:00 PM	3:00-5:00 PM
June 2	8:00-10:00 AM	10:00-12:00 AM	1:00-3:00 PM	3:00-5:00 PM

June 5 is reserved as an extra date for back-up purposes.

State which is your company's preferred demo time slot, and also state all other time slots being acceptable alternatives.

Text field

Preferred slot - June 1st 2023 - Time: 10:00-12:00 AM
Alternative slot - June 1st 2023- Time: 8:00-10:00 AM and 1:00-3:00 AM

1.1.3 Terms and definitions

Solution	The openEHR platform, related tools, and supporting applications that the RFI respondent can offer
RFI respondent	The part responding to the RFI
RFI document	This document

Application	A CDR external application integrated with the CDR, as part of - or not part of - the Solution.
CDR	Clinical Data Repository implementing the openEHR specifications
We	The group of county counsils issuing the RFI document
Request Context	All request metadata on the incoming HTTP request such as methods, headers, access tokens etc
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.

1.1.4 No procurement

This is not a procurement. Please note that this does not constitute an RFP. Response to this invi

However, this is not bound to accept any of such information and/or expression of interest or to consider it further in any associated documents such as a RFP.

1.1.5 Confidentiality

During the RFI process, confidentiality prevails according to Chapter 19, Section 3 of the Public and Confidentiality Act (2009: 400).

Upon completion of the RFI, continued confidentiality may apply if there is reason to fear that a disclosure of information concerning the individual's business and operating conditions could cause harm to the individual. Furthermore, continued confidentiality may apply for the protection of the public interest.

When appealing decisions on confidentiality of information, RFI respondent shall assist the county councils and be responsible for their own costs arising from this.

In the event that the RFI respondent requests confidentiality, the RFI respondent must enclose documents describing the scope of the confidentiality and describe what damage the RFI respondent may suffer in the event of a publication. If the RFI respondent requests confidentiality, the RFI respondent must enclose a document specifying the parts of the RFI document for which the RFI respondent requests confidentiality and describe the damage the RFI respondent may suffer in the event of a publication.

a. Is privacy requested?
Yes/No

Answer
No

b. In those cases that the RFI respondent requests confidentiality, the RFI respondent must here attach what the privacy includes and describe which damage the bidder will suffer upon publication.
Attachment

0 attached documents

-

1.1.6 Questions about the request for information

All questions regarding the RFI must be asked via the VISMA TendSign RFI system, www.tendsign.com.

The wishes to receive questions in such a way that, together with the county counsils answer, they can be published without taking measures. The questions should therefore not contain information about the questioner's company, products or other information that can identify the questionnaire.

The county counsils want the RFI respondent to ask questions one at a time with reference to the point in the RFI document to which the question relates.

The county counsils answer the questions electronically in VISMA TendSign.

1.2 About this RFI

Region Östergötland, Västra Götalandsregionen, Region Uppsala, Region Stockholm, Region Skåne, and Region Kalmar (collectively referred to as “we” and “us” in this document) cover two thirds (⅔) of Sweden’s population. The majority of the county councils manage university hospitals with an extensive share of research and advanced healthcare. This RFI initiates the way forward, towards better healthcare and documentation solutions in Sweden.

This RFI aims at reaching all suppliers of openEHR solutions with an interest in the European market, in order to get an update on the latest news within the field. Doing this as a joint activity ensures higher quality results and is also timesaving for all parties.

The RFI may result in one or several procurements, either by each county council separately or by two or more county councils together. No decisions regarding possible joint procurements are taken yet and more county councils and organizations than these 5 may initiate procurements based on this RFI. Also note that all suppliers are welcome to take part in later coming procurements. There is no obligation to participate in the RFI and demo sessions, and participation does not affect later evaluation.

1.2.1 Facts about the County councils

The table shows some facts in figures about the county councils.

	Inhabitants (Total Swedish population is 10,5 million)	Hospitals	Health clinics	Dental care clinics	National specialized medical care assignments (46 different ones available)	Current main EHR system
Region Stockholm	2 440 027	5 (N/A)	Appr 600 (appr 1900)	Appr 80 (N/A)	36	CGM TakeCare
Region Uppsala	400 682	2 (3)	36 (58)	25 (80)	14	Cambio Cosmic
Region Östergötland	471 912	3 (3)	33 (47)	33 (109)	6	Cambio Cosmic

Region Skåne	1 414 324	9 (10)	100 (182)	69 (69)	25	Cerner Millenium
Västra Götalandsregionen	1 758 656	18 ()	117 ()	167 ()	29	Cerner Millenium
Region Kalmar län	247 711	3 (3)	26 (37)	18 (31)	N/A	Cambio Cosmic
Region Jönköpings län	369 184	3 (3)	28 (40)	26 (86)	N/A	Cambio Cosmic
Sum	7 102 496					

Population 2022 according to https://www.statistikdatabasen.scb.se/

National specialized medical care according to https://www.socialstyrelsen.se/en/clinical-practise-guidelines-and-regulations/regulations-and-guidelines/national-specialised-medical-care/.
Numbers within parenthesis () include collaborating private clinics etc.

1.2.2 Business impact goals

Three business impact goals of introducing openEHR-based healthcare systems are:

- Faster adaptation of IT systems to the constantly changing needs of the healthcare clinicians, including a more efficient system development process
- Increased control of stored health record data and increased reuse of information structures within and between applications, and between caregivers
- Increased freedom of action for the regions when the data is stored in a vendor neutral and open format

1.2.3 Purpose

The Swedish county councils are in the process of establishing an infrastructure for information management and information governance based on an information strategy and its target architecture. A key component of this infrastructure is to be able to store healthcare related information in a standardized and application neutral way.

The interoperability solution is an addition to existing healthcare information systems. A subset of the patients' medical records must be possible to handle in the CDR component both as master record as well as copies. We need a standardized reference model for how the information and data is structured and implemented in the CDR. Each application that renders information should have the ability to select, and customize its information stored in the CDR, in accordance with the reference model.

An example where this CDR capability would be relevant, is when an independent health app is used, but is not part of the main healthcare information system. In the long term, the CDR component will also be used for other applications of healthcare related information. Another early application will be remote/home monitoring.

Other secondary uses of interest are: patient created data, biobank data, healthcare business development, BI, AI, CDR, research, and quality registries.

2. Part 1: Questions

2.1 Questions

Answer the questions in this section in writing. Answer the questions that are relevant to your Solution. Not all questions in this RFI need to be answered, but the majority needs to be answered in order for you to be invited to the demonstration.

The supplier must enter all answers in the system.

The supplier may not attach documents.

2.1.1 General

a. What is the name and intended purpose of your Solution? Please name and (very briefly) describe the openEHR-related tools and platform components that you may be referring to in other parts of your RFI response.

Text field

Medblocks CDR: Medblocks provides an openEHR CDR based on EHRbase and a FHIR CDR based on HAPI JPA. This is our default choice for obtaining fully compliant openEHR and FHIR APIs for use with the rest of the Medblocks Ecosystem.

Medblocks Ignite: Serves as a single entry point for clinicians, patients, and administrators to access and modify clinical data and help launch multiple apps from it. Provides User Management, Access Control, and the ability to install CDS Hooks, Web Components and iFrame apps - including SMART on FHIR/openEHR apps.

Medblocks UI: A library of open-source web components that can be used to build and deploy openEHR and FHIR-based web forms. It can work with any framework that supports web components, including React, VueJS, Angular, Svelte, and many more.

Medblocks Connect: It is an ecosystem of Kafka connectors for moving data into, out of, and between openEHR and FHIR CDRs. It provides multiple source and sink connectors – some that work exclusively with Medblocks CDR and others that work with any openEHR and FHIR compatible APIs.

b. In which country is your company located? Are there any sales partners or support partners in Sweden or Swedish speaking staff? Can your Solution or parts of it, e.g. additional services or license packs, be delivered via existing national Swedish framework agreements (see https://www.avropa.se/topplankar/In-English/).

Text field

Medblocks is registered in India. We have a Cooperation agreement with Atea, which operates in Sweden with existing deployments. We also have a good working relationship with Vitagroup GmbH.

c. Describe the overall architecture of your Solution.

Text field

Medblocks CDR: We believe a vendor-neutral clinical data repository should be available for anyone to configure and deploy easily. We use EHRbase and HAPI JPA FHIR with any compatible Postgres database to form the foundation of our Clinical Data Repository. Both of these projects provide a high degree of customization to fit the needs of the end user - HAPI FHIR via Interceptors and EHRbase via Plugins.

We use HAPI JPA, a fully-fledged reference FHIR server, as the primary data source for demographics and encounters. We prefer storing FHIR Resources like Patient, Encounter, Practitioner, Location, and Organization in the FHIR server and using openEHR for clinical data. Using Medblocks Connect, the HAPI FHIR server is also able to synchronize between Composition instances in the openEHR CDR and corresponding clinical FHIR resources like Observation, Condition, AllergyIntolerance etc. against country/region-specific FHIR Profiles and Implementation Guides. This enables the use of SMART on FHIR applications when used with Medblocks Ignite.

EHRbase serves as the primary data source for clinical information based on openEHR APIs. We synchronize Patients created in the FHIR server with EHRs created on the openEHR side. This provides support for both the official openEHR REST API along with simplified APIs like EHRScape. We have also built a few plugins that integrate with EHRbase for authorization interception and Eventing.

Medblocks Ignite: Our flagship product is built to enable the next generation of vendor-neutral applications. Ignite serves as a dashboard that abstracts away most of the common requirements of any health information system while respecting internal and external third-party apps as first-class citizens.

Access Control: Ignite provides User Management and Access Control primitives on top of the openEHR and FHIR APIs. All requests go through a decision engine based on the Open Policy Agent that accepts, rejects, or modifies the request before it hits the destination system. This is used in conjunction with a policy language like Rego to create a highly customizable access control system.

Ignite implements an OAuth2.0 / OIDC server with full compatibility for SMART on FHIR. This includes fine-grained scope selection as well as multi-factor authentication. Ignite is the first implementation of the proposed SMART on openEHR protocol that intends to extend SMART on FHIR to also cover openEHR APIs - https://discourse.openehr.org/t/help-in-drafting-specification-for-building-and-distributing-apps-on-top-of-openehr-and-fhir/2936.[Link]

Ignite provides a UI that can serve as the primary interface for clinicians to access multiple apps from. For example, Ignite provides a generic way to assign patients to practitioners by configuring FHIR search parameters on Patients and Encounters FHIR APIs. This provides practitioners with an easy-to-use, yet powerful patient list that serves as their entry point into other applications. Once the practitioner enters a patient, we try our best to get out of the way and let apps do their jobs. That being said, most applications are specific to a particular use case, so we also try our best to cover the more mundane use cases. We enable administrators to build and configure pages that can render reusable widgets using AQL and FHIRPath - This can for instance be used to show a table of the Patient’s allergies or show a Blood Pressure chart simply by using a few AQL statements and choosing the widget. This same mechanism can be used to enhance the patient list and the banner with more useful information that can be pulled in from the openEHR CDR - For example, comorbidities can, for instance, show up in the patient banner, while the latest lab test for Covid within a week if positive, can be configured to show up on the patient list.

Apart from the above, most of the functionality of Ignite is expected to be derived from building and installing internal or third-party apps. Ignite provides 90% of the screen real estate for these applications. Applications can also choose to use Ignite to just authenticate via an OAuth2 and launch standalone. The following launch modes are currently supported:
Practitioner Standalone Launch using OAuth2 / SMART
Practitioner EHR Launch using embedded iFrames

Patient Standalone Launch using OAuth2 / SMART
Patient EHR Launch using OAuth2 / SMART
Web component launch within the EHR context is being passed in as properties/attributes
Backend Launch and Service Account Access

Ignite can be configured with Clinical Decision Support via External CDS Hooks (which have been extended to include openEHR resources) as well as its inbuilt policy engine is powered by Rego and Open Policy Agent. Internal and third-party applications can communicate back to Ignite via SMART Web Messaging (for iFrames) or Custom Events (for Web Components) to relay back the state to trigger CDS. This can be used to trigger cards and suggestions at the right time. For example, an app built using Medblocks UI can natively emit and receive events from Ignite based on the state of a particular field in a form. Whenever the openEHR archetype “Problem / Diagnosis” is used and the “Problem / Diagnosis name” is set to a subtype of “Hernia”, a CDS policy can be configured to show up a card that provides more instructions on the Hernia protocol in the hospital as well as automatically suggest medication, prescriptions, review notes, and investigations that are common for Hernia within the app so that the physician can enter this information with a few taps.

Ignite requires its own datastore for configuration and can runs on any compliant combination of FHIR and openEHR APIs. Apart from our own Medblocks CDR, we have also tested and evaluated it against other vendors’ APIs as well. We have tested the FHIR APIs against multiple major cloud vendors like Google FHIR API, Azure Healthcare API, Health Samurai’s Aidbox, Smile CDR, and Better’s Demographics. We have tested the openEHR APIs against Vitagroup’s HIP CDR, Better CDR, Solit Cloud’s EHRDB and EHRServer. We are planning to test compatibility with Cambio soon.

Medblocks UI: A library of web components that can be used in any modern Javascript framework. It is open-source and enables the creation of forms that output simplified openEHR compositions natively from the browser. The degree of customizability that Medblocks UI offers is second to none since it is like building a normal frontend application. Any custom logic can be represented, and conditional rendering is second nature. It is compatible with any openEHR CDR that supports the simplified composition data formats.

Medblocks UI comes with a VSCode Extension that can display a sidebar with all data points in a particular template being worked on and indicate if the data point in a template matches with what a developer is working on. It also enables drag (technically click) and drop style designing of forms quickly based on an openEHR web template. It also offers diffing between different versions of templates to quickly identify parts of an application that needs to be corrected or changed if a template changes.

Medblocks UI is extensible with new custom components, and also includes special plugins to interact with Terminology APIs for SNOMED CT, RxNorm, LOINC, ICD and other Custom APIs that are needed to populate search boxes while capturing data.

Medblocks UI also provides a powerful suggestion and autocomplete framework that can be driven by external APIs to automatically populate parts of the form based on inputs in other fields. We have extensively used this in practice for physicians to autocomplete prescriptions and quickly enter a normal physical examination for instance.

Medblocks Connect:
Provides Kafka Connect Source and Sink connectors to enrich the already abundant Apache Kafka ecosystem of connectors to integrate and move data between almost any system. We have built the following connectors:
openEHR sink connector: Uses the openEHR REST API to push TEMPLATE, COMPOSITION, EHR, CONTRIBUTION, FOLDER data in Kafka Topics to destination systems. Works with any compliant openEHR API endpoint.
FHIR sink connector: Uses the FHIR Bundle API to validate and push FHIR Resources into destination systems. Works with any compliant FHIR API endpoint.

EHRbase source connector: Works with EHRbase and vitagroup's HIP CDR to retrieve any changes made to TEMPLATE, COMPOSITION, EHR, CONTRIBUTION, FOLDER and outputs it as a changelog in Kafka topics.

EHRbase sink connector: Works with EHRbase and vitagroup's HIP CDR, to validate and push TEMPLATE, COMPOSITION, EHR, CONTRIBUTION, FOLDER data in Kafka topics into a target database system efficiently.

HAPI JPA source connector: Works with HAPI JPA server and Smile CDR to retrieve changes made to FHIR Resource and outputs a changelog in Kafka topics.

HAPI JPA sink connector: Works with HAPI JPA server and Smile CDR to validate and import multiple FHIR resources in bulk into a database efficiently.

If the customer is invested in the Nifi ecosystem, we also provide the following Nifi Processors:
openEHR Composition validation
FHIR Resource validation
EHRbase PUT Composition
EHRbase PUT EHR

d. Describe if/how openEHR's Task Planning functionality (or other process support) is supported by your Solution now, and your future roadmap for such support.

Text field

We do not currently have support for openEHR's Task Planning. However, we are planning on incorporating Task Planning as a part of Medblocks Ignite to power customizable workflows.

e. Describe if/how the Solution supports development and use of clinical decision support (CDS), for example using openEHR's GDL or GDL2 specifications now, and your future roadmap for such support.

Text field

Medblocks Ignite supports displaying CDS cards based on CDS Hooks that have been extended to work with openEHR compositions as well as FHIR resources. Changes to a composition or resource in SMART on FHIR and openEHR applications can thus trigger an event to indicate the change, and CDS hooks can be sent to an external server for this purpose. We are fully compliant with the CDS Hooks specification and plan on supporting any engine that supports GDL / GDL2 via CDS Hooks.

Link: https://cds-hooks.org/

2.1.2 Delivery models

a. List the delivery/deployment models you support, such as local installation (OnPrem) or cloud installation (for instance SaaS)?

Text field

We deliver our products OnPrem or on the Private cloud of the client as long as the client provides a reasonable Kubernetes, Postgres and S3 abstraction for us to use.

b. Describe, in the case of SaaS deployments, your subcontractor structure used to deliver the service. List any hyperscalar public cloud services used and the jurisdiction they operate in with relation to the EU/GDPR and transfer of personal data.

Text field

We currently do not offer Hosted SaaS for any of our solutions

c. If you are dependent on third-party suppliers in your solution proposal, how do you package this with an overall responsibility regarding usability, licenses and support?

Text field

All third-party solutions used are open-source. We have a good working relationship with vitagroup to provide support for EHRbase, and a working relationship with Smile Digital Health to provide support for HAPI JPA FHIR server.

d. Can applications based on output from your products be published as open source? If so, are there any restrictions on usage? This implies e.g. that generated code, forms, configuration information etc. and exported runtime components should be perpetually allowed to be included in open source based systems and in associated, possibly public, versioning systems (like GitHub).

Text field

Medblocks CDR: Plugins for Medblocks CDR will invoke the same license agreement as the underlying license for EHRbase and HAPI FHIR, both of which are Apache 2.0. There are no restrictions in publishing this code.

Medblocks UI: Open-source and based on the Apache 2.0 license. Any forms created can be published openly without any restrictions.

Medblocks Ignite: The apps built on top of Ignite can be published openly. They have a good chance of working with existing EHR systems that support SMART on FHIR if the apps fall back to FHIR APIs.

Medblocks Connect: KSQL Scripts, Connector configuration and any additional plugins can freely be published openly.

e. Describe how your product can be installed using containers and container orchestration tools such as Kubernetes.

Text field

All our products are built into OCI containers and can be deployed on Kubernetes. The resources can be installed using tools like Kustomize or Helm. There are the following deployment options we provide:
A) Medblocks CDR, Medblocks Ignite:
1) Kubernetes Deployments, Jobs and Services:
With no Ingress resources
With Kubernetes Ingress resources
With Ambassador Ingress CRDs
With Contour Ingress CRDs
With Linkerd service mesh
2) Knative Service Deployment
With Ambassador
With Contour
With Istio
B) Medblocks Connect:
Kubernetes Deployment
Strimzi Kafka Connect CRD

f. Describe your approach to scaling your Solution. Describe known limitations, for instance regarding performance.

Text field

Medblocks CDR: Scales horizontally with the increase in the number of running instances. Most of the performance bottleneck would come from the underlying Postgres database. There are a few known performance issues with the upstream EHRbase repository on performing AQL on large datasets - most of these are being rectified with newer deployments with YugaByte DB. HAPI JPA FHIR has been tested at large scales and works well at scale.

Medblocks Connect: Kafka has been proven at massive scales. The plugins that Medblocks provides have the same guarantees about consistency and ordering that Kafka Connect provides. Scaling can be achieved by increasing the partition of Kafka topics and increasing the number of instances of Kafka Connect.

g. Briefly describe your three (3) largest or most interesting customer installations based on an openEHR CDR. Also describe how long it took to go from purchase to operational system with real patient data and actual use.

Text field

1. Clinikk, India: Clinikk is a company in India providing affordable subscription-based primary healthcare at 18 locations. They required that their doctor’s dashboard needed some protocols and decision support in place. They also wanted to use an open API stack that they could use to build applications on. Most of the patient registrations and subscriptions happened on their own internal apps which used MongoDB as a database. We used Medblocks Connect to set up pipelines from their database to synchronize with our FHIR and openEHR servers. Medblocks Ignite was used to manage the patient visits and practitioner login. The specialized doctor’s dashboard for Clinikk was built jointly with the Clinikk team using Preact and Medblocks UI to embed as an iFrame app within Ignite. We designed and used multiple openEHR templates on the same application to cover prescriptions, care plans and patient summaries. The doctor’s dashboard integrated with SNOMED CT and we were able to provide automatic suggestions on the HOPI based on the chief complaints entered. We were also able to configure “protocols” that were designed and managed by doctors at Clinikk in a separate content management system based on a diagnosis. The protocol included drugs to prescribe, investigations to order and instructions to give the patient. An algorithm to detect any deviation from the protocol was also designed, which asks the doctor for a “reason for deviation” in case their prescription does not align with the protocol. Most of the clinical decision support and deviation algorithms were configured to use the openEHR data model. Medblocks Ignite’s configurable dashboards also served some of the data viewing needs. The data from Medblocks CDR was then sent back to their system via webhooks in order to process billing and prescription PDF generation that is given to the patient. From the date of purchase, we went live with the Medblocks Platform in the pilot location of Bannergatta Road, Bangalore in 4 months. They rolled out to the rest of the 18 locations in 6 months.
Link: https://www.clinikk.com/

2. Kenko, India: Kenko is a medical insurance company. They wanted to digitize all documents coming into their platform to gain insights into customer behavior. Most documents were uploaded by customers before or after an In-Patient admission or out patient visit on their mobile app which was running on a MongoDB backend. We used Medblocks Connect to stream all old data as well as new mobile app submissions in the database to Medblocks CDR. Ignite was used as the portal for a team of digitizers. An app was jointly built using Svelte and Medblocks UI to embed in Ignite. The app showed the document to be digitized on the left and the forms based on openEHR on the right to make it easy to manually verify and fill in any incomplete information. Other external OCR and Machine Learning APIs were also being used to prepopulate the forms and offer suggestions as the digitizers were working on them. We integrated with the Indian Extension of SNOMED CT to populate Diagnosis, Medications and Procedures. Most of the data was viewed directly on a dashboard configured with charts and widgets within Ignite. With thousands of records arriving every minute, our stack was really put to the test. We pushed out all the digitized records from the CDR into Elastic Search using Medblocks Connect. Visualizations were built on top of Kibana and Elasticsearch to do population-level reporting and analytics. The Kibana Dashboards were also embedded within Ignite with appropriate access controls for administrators. From purchase to deployment, it took our team 5 months to go live. We are still iteratively improving the workflows by helping make adjustments to the app.
Link: https://kenkohealth.in/

3. Atlantic, South Africa: A videoconferencing company that wanted to provide an “EHR” along with a telemedicine product they have been testing out. The telemedicine product includes an array of devices including – electronic stethoscopes, multipurpose scopes with inbuilt cameras, pulse sensors, electronic blood pressure cuffs and ECGs. These devices along with the telemedicine product would be provided to a nurse in remote centers, and doctors from all over the country could provide a consult via video conferencing. The requirement was to have a single portal where the doctor could see all records of the patient, including the real-time data coming from the devices. These devices sent out data in HL7v2 over HTTP. We used Medblocks Connect to convert these into openEHR data points to ingest in the Medblocks CDR. We built out tables and charts using Medblocks Ignite’s customizable widgets. We were able to go live in 2 months.
Link: https://www.atlantic.co.za/

h. Describe what kind of infrastructure your Solution requires from a customer. Also describe your normal implementation/deployment process.

Text field

For a production workload, the minimum requirements would look like this:
1) A Kubernetes cluster with at least 3 nodes in different availability zones. Each node configured with 4 vCPU and 16 GB RAM.
2) A Postgres cluster with high availability with node configuration of at least 2 vCPU, 4 GB RAM and 100GB storage with automated backups.
3) S3 compatible storage

Our Implementation and Deployment model process looks like this:
1) We first test the infrastructure and check if it meets all requirements. This includes checking compatible Kubernetes and Postgres versions.
2) We start educating the customer’s team on openEHR and FHIR and provide access to our educational content as well as curated playlists on YouTube.
3) We set up Gitops for the deployment on a shared git repository and provide a clear explanation of what components are being deployed. All updates to Medblocks CDR, Ignite, Connect or Applications must be made on this repository.
4) We start integrating existing data using Medblocks Connect. Our team works on mapping the data and configuring the scripts for transforming data.
5) We parallelly start configuring workspaces and access control policies in Ignite. We test if the requirements can be met using the configurable widgets in Ignite.
6) We start building templates and apps with Medblocks UI and a framework of choice if needed and embed this inside Ignite.
7) These apps are released quickly for testing by the clinical team and iterated upon. Our CI/CD releases automatically by making commits to the shared Gitops repositories.

i. Describe your software lifecycle strategy and release cadence.

Text field

Our software lifecycle strategy is based on best practices in software development, focusing on continuous improvement, rapid innovation, and agile methodologies to deliver high-quality software.

Development Methodology: We follow the Agile manifesto, which emphasizes iterative and incremental development, allowing us to adapt quickly to changing requirements and market dynamics. Our development teams work in sprints, typically lasting two to four weeks, to deliver working software increments regularly.

Continuous Integration and Continuous Deployment (CI/CD): Our development process leverages CI/CD to automate the build, test, and deployment phases, ensuring rapid and reliable delivery of software updates.

Release Cadence: We have a flexible release cadence and get the customer’s approval before any major release. Major releases, containing significant new features and improvements, are typically scheduled every three to six months. Minor releases, consisting of bug fixes, security patches, and small enhancements, occur more frequently, usually on a weekly basis.

Quality Assurance and Testing: Our QA team employs a combination of manual and automated testing techniques, including unit, integration, system, and acceptance tests, to ensure the quality and reliability of our software.

Maintenance and Support: We are committed to providing ongoing support and maintenance for our software products. Our support team is available 24/7 to address any technical issues, while our maintenance team ensures that software remains up-to-date with the latest security patches and bug fixes.

End-of-life (EOL) Planning: To ensure a smooth transition for our customers, we provide advance notice and clear communication regarding the EOL of our software products. We offer migration assistance, data export capabilities, and continued support for a predefined period after the EOL to minimize disruptions to our customers' operations.

j. Describe your future roadmap. What major features are planned and when are they planned to be released?

Text field

1) More clinical widgets for Ignite - September 2023
2) Integration of Medblocks UI forms within Ignite Dashboards - December 2023
3) Easy drag and drop UI for configuration of Ignite dashboards - February 2024
4) Installable specialty clinical apps for Ophthalmology, Internal Medicine and Surgery on Ignite - January 2024
5) GDL2 Engine that supports CDS Hooks - August 2024
6) Task Planning within Ignite - October 2024

2.1.3 Legal and regulatory aspects

Please refer to background information in appendix “OpenEHR – an Implementors Guideline related to Swedish laws and regulations in healthcare”. It also reflects our level of ambition, and discusses some different possible openEHR-based solutions. Please feel free to be inspired by this document; we also look forward to receiving alternative solutions and discussions. We refer to COMPOSITIONs below to make the text more readable but we are actually interested in corresponding behavior regarding all relevant VERSIONED_OBJECTs (for example FOLDERs).

2.1.3.1 Multi-tenancy, Federation and Metadata

a. Describe how the Solution can be configured to support multi-tenancy where clinical data for hundreds of organizations (care providers/care units) can be managed efficiently.

Text field

Medblocks CDR and Ignite provide multi-tenancy by partitioning different tenants into their own physical database and logins. As long as the Patient across their different organisations uses the same identifier, we are able to perform both cross-tenant querying as well as syncing records between organisations. Cross-tenant queries can however be performed by setting up data pipelines with Medblocks Connect and having a parent CDR that syncs data from multiple child CDRs. The latency of Medblocks Connect is minimal and provides near real-time syncing. However, the queries across multiple tenants are eventually consistent, and the child CDRs do not wait for the parent to sync before committing data points in their local database.

This has multiple advantages over having a single CDR that has “multi-tenancy”.
1) Since the entire stack is naturally partitioned, we have great read-write performance on the local CDR that serves an organization with minimal infrastructure.
2) Since the system is naturally distributed an outage in one instance does not bring the entire system down.
3) A parent CDR (let’s say regional) is able to either automatically pull data from multiple local child CDRs, or have a data governance process where each TEMPLATE, COMPOSITION, EHR, CONTRIBUTION, FOLDER, Resource is validated and signed by another authority before it is indexed in the parent CDR. The same applies to pulling data from a parent CDR into local child CDRs - it can be configured to be automatically done, or require a manual verification process before data points are merged between CDRs. This models the natural ownership of data between different organizations and each can have its own data governance policy before merging foreign data into their system.
4) Complex data topologies can be established within the different organisations as per their business needs and data governance policies. This can be modelled to fit local laws and organizational structures. Each CDR gets to configure where they pull data from and require consent and authorization from the source CDR where they pull from.
5) The parent CDRs that mainly serve analytics purposes and downstream syncing being separate from the operational day-to-day local CDRs have a significant advantage on infrastructure provisioning and performance.
Most of the syncing process is done through industry-proven technologies like Kafka Mirroring along with the Medblocks Connect ecosystem

b. Describe how the Solution can be configured in a fine-grained multi-tenant model (see Appendix A) so that a COMPOSITION and/or parts of a COMPOSITION within an EHR record can be attributed organizational ownership. Also describe how and where this metadata can be persisted.

Text field

Since multi-tenancy is achieved by having separate physical CDRs per organization, the COMPOSITIONs within an EHR are naturally stored as it is. Foreign COMPOSITIONS from other organizations can be configured to be signed and tagged.For example, this can be configured to filter result set by values in COMPOSITION.context and limit AQLs to only return local CDR data, even when there is foreign data in the CDR.

The metadata of organisation information is configurable and is a property of the Medblocks Connect Sink Connectors:
For FHIR Resources we support: meta.source, custom extension
For openEHR resources: Any valid path representation can be appended to the COMPOSITION

c. Describe how metadata about organizational ownership/multi-tenancy, and about source (e.g. originating/feeder-system), can be verified/validated against the Request Context and/or external attribute sources to make sure that the proposed metadata is valid and that the user has sufficient permissions to write/modify data for this unit.

Text field

With Medblocks, each organizational unit gets its own set of login credentials and access control and is strictly managed at the organizational level. Foreign compositions entering the organization come through Medblocks Connect – this can be configured to require a signature by a local organisational personnel before it is inserted into the CDR, or can be automatically inserted.

The original COMPOSITION still maintains all the original fields like composer, and signature if any along with the full versioned history. However, when it enters the local CDR, the ownership and audit log of how it entered the system is delegated to the assigned personnel in charge of data governance or the Medblocks Connect service account.

2.1.3.2 Querying and Multi-tenancy

a. Describe how (see Appendix A) the Solution can be configured to filter a response from the EHR API resource endpoints based on metadata from the Request Context, external attribute source and/or metadata on the COMPOSITION itself (such as validated metadata for organizational ownership).

Text field

By default, local organizational queries only retrieve local compositions. This provides a good degree of isolation between multiple tenants. However, if the CDRs are configured to pull data from other sources using Medblocks Connect, there are still ways to limit the responses using access control.

There are multiple solutions that are available:
1) Spin up CDRs that meet specific requirements and have data pull policies configured correctly with Medblocks Connect so that only the data that will need to be accessed by end users is indexed.
2) A plugin for the EHRbase CDR can be built that checks for the user's access token against specific data points in a composition before returning it to them. This can call an external decision endpoint like Open Policy Agent to model more complex decision logic using Rego.

b. Describe how (see Appendix A) the Solution can be configured to block or filter out parts of a RESULT_SET from the Query Execute API resource endpoints based on metadata from the Request Context, external attribute source and/or metadata on the COMPOSITION itself, such as validated metadata for organizational ownership. (Example of possible solution: Incoming ad-hoc queries and/or stored queries may be temporarily modified to support the filtering.)

Text field

In the case of a CDR having compositions from multiple organizations, these solutions are available:
1) Stored AQL can be used to include these attributes in the templated query: eg: composition/context. Specific access control rules in Rego can be configured in Ignite to allow only certain users to access cross-tenant information.

2) Medblocks Ignite access control policies can also intercept AQL queries, inspect the user accessing the data and add append an additional WHERE clauses to limit the result sets. The additional WHERE clauses and the policy itself can be delegated to the decision engine to enable specific use like - “only return local organization’s compositions for this user”

c. Describe if and how (a possibly extended set of) the openEHR Reference Model can be used to block or filter out parts of a RESULT_SET from the Query Execute API resource endpoints based on metadata from the Request Context, and/or external attribute sources. Describe at least support for using the following classes for blocking/filtering data

i. FOLDERs

ii. TAGsF

iii. EEDER_AUDIT

Text field

Our authentication and access control stack supports intercepting and modifying AQL queries before it’s sent to the destination openEHR CDR based on the logged-in agent. Any query that can be handled by the underlying openEHR API can be used to filter out RESULT_SETs.

Regarding FOLDERs, TAGs and FEEDER_AUDIT: We rely on the underlying CDR for AQL support.

d. Describe how the Solution can be configured to block and/or allow requests to resource endpoints from the ITS-REST specification based on metadata from the Request Context and/or external attribute sources.

Text field

A full degree of access control can be performed on the openEHR REST API (as well as FHIR APIs) using the decision engine based on Open Policy Agent and Rego. This includes calling third-party systems before making decisions - like calling an HTTP endpoint to check if a particular user is blacklisted.

A few examples of the attributes that can be used to accept, reject, and modify requests are listed below (for the full list, send us an email):
HTTP Method
HTTP Route
HTTP Query Parameters
HTTP Request Body
Origin Domain
Origin IP Address

2.1.3.3 Bulk Operations

a. Describe any tooling and/or APIs available for managing bulk operations on COMPOSITIONs. Describe how the target set of COMPOSITIONs (bundle/batch) can be defined from a result of an AQL query.

Text field

Medblocks does not provide any APIs natively. However, there have been multiple use cases with Medblocks Connect where COMPOSITION data from openEHR is simplified and indexed in destination analytics databases like Elasticsearch, AWS Redshift and Google BigQuery. Medblocks Connect guarantees consistency between the CDR and the external analytics database of choice. This can enable direct querying of openEHR data using other query languages like SQL and can help enable visualization using tools like Apache Superset or Kibana

b. Describe any tooling and/or APIs available for managing bulk import operations of COMPOSITIONs. Describe how metadata on COMPOSITIONs are validated/verified.

Text field

Medblocks Connect EHRbase Sink connector, as well as openEHR Sink connector, can be used to bulk import massive amounts of COMPOSITIONs into an openEHR CDR. Medblocks Connect can be configured to check for signatures on the incoming COMPOSITIONs from a specific user to index it into the CDR as an import action by the said user.

2.1.3.4 Audit Logging

a. Describe the set of triggers (instrumentation) the Solution can use for audit logging. What is logged and when?

Text field

All requests coming into the Medblocks CDR, Ignite and Connect need to pass through a centralized decision engine based on Open Policy Agent. The decision engine logs every single invocation with the Trace headers (W3 Trace context, B3, Zipkin headers supported) and Decision ID along with the result of whether it has allowed, denied or modified the request. This logging includes all metadata including trace IP address, device used, and authentication mode for both HTTP requests and Kafka TCP invocations.

All logging from the decision engine goes to the STDOUT of the deployment by default. The configuration can be changed to direct it to a file as well.

b. Describe how the Solution can be configured to export audit logs and/or integrated to external SIEM systems. Also describe and/or list the supported technical interfaces.

Text field

Most log aggregation tools that work with Kubernetes can be used to collect audit logs from the decision engine deployment. This includes, but is not limited to:
OpenTelemetry Collector
Splunk Log Observer
Elastic Logstash
Datadog Agent
FluentBit
Grafana Agent
Grafana Promtail

2.1.3.5 Certification of products, tools and modules

a. Are any of your openEHR products, tools or modules certified (CE labeled) according to EU Medical Device Directive 93/42/EEC or the EU Medical Devices Regulation (MDR)? If yes, please state which product or module that fulfills which regulation.

Text field

No. Our products are currently not CE labelled.

b. Describe your experience of the process to CE label a software as a medical device?

Text field

We have no experience in the process of obtaining a CE label for our software. However, we'd be keen to understand the legal requirements and work with our European partners to obtain one wherever required.

2.1.3.6 Accessibility

Describe how the Solution supports (or helps creating) end user interfaces in accordance with the European accessibility directive European accessibility act - Employment, Social Affairs & Inclusion - European Commission (europa.eu).

Text field

All Medblocks UI components are built with accessibility in mind. Any applications built using them will also support the same accessibility patterns given that appropriate aria labels are provided while building them.

Medblocks Ignite is also accessible via screen readers and direct keyboard manipulation. It has excellent aria labelling for all important interactions.

2.1.4 Platform and development

a. What parts of the Solution are open source and what parts are proprietary? Describe what open source license you use.

Text field

Medblocks CDR = HAPI JPA FHIR + EHRbase - Open Source, Apache 2.0
Medblocks UI - Open Source, Apache 2.0
Medblocks Ignite - Proprietary
Medblocks Connect - Proprietary

b. Describe any prebuilt products or EHR-modules based on the platform that you can provide, for instance end-user applications for surgery, emergency wards, medications, or primary care. Also describe any provided "portal" functionality or similar that can easily be configured to different use cases where e.g. clinical end users can browse, read and enter openEHR-based data. Also briefly describe the pricing model for these.

Text field

Medblocks has built a multitude of apps for organizations and different specialties; however, we do not offer them as packaged apps to be installed as is just yet (it's on our roadmap). Instead, we reuse similar patterns and put together an application quickly for a specific use case. Here are a few of the specialties we have covered and built for: Ophthalmology, Surgery, Emergency Medicine, Insurance Adjudication, Tele-Medicine, Primary Health Care and Community Medicine.

Medblocks Ignite provides easy to customize and build dashboards with the use of widgets. It also provides a unified portal for both practitioners as well as patients to enter records and launch other apps.

c. Describe your integration support, tooling and experience, including but not limited to the list items i-vii below. Clearly indicate which list item the answer refers to.

i) Software development kits (SDK:s) for developing and integrating towards your API:s etc.

ii) Publish/subscribe patterns

iii) HL7 FHIR

iv) API standards (such as HL7 v2, IHE, ODBC, OpenAPI) and other interoperability and connectivity standards

v) Integrations with medical imaging standards such as DICOM

vi) OMOP and other standards used for research

vii) Existing EHR systems in Sweden (if so, please state which)

Text field

i) SDKs: Medblocks UI and Medblocks UI VSCode extension provides components that can be used to put together applications that work with openEHR and FHIR APIs in hours. We also provide a testing library for Playwright that can load the forms with multiple permutations and combinations of input to simulate the user and to check if edge cases still result in a valid composition being generated.

ii) Publish/subscribe patterns: Medblocks Connect is based on Kafka Connect and provides source and sink connectors to move data into, out of and between CDRs. This can extensively be used for a multitude of use cases including Real-time data import from legacy systems, Real-time export to analytics databases, Federating organizational CDRs into a single parent multitenant CDR and establishing bidirectional data flows, mapping FHIR and openEHR resources based on a Profile. Along with our Kafka Connectors, Kafka Streams and KSQL can be leveraged to build and deploy custom streaming workloads easily.

iii) HL7 FHIR: Most of our stack natively uses FHIR to store and retrieve non-clinical information. Medblocks UI can be used to build FHIR-based forms as well. Clinical information is streamed to and from FHIR CDRs from openEHR and other external systems using Medblocks Connect. We also provide SMART on FHIR capability on top of any compliant FHIR API by leveraging Medblocks Ignite.

iv) API standards:
HL7v2: Medblocks Connect supports HL7 over HTTP and can ingest messages into the CDR using our destination sink connectors.
IHE: We do not implement any IHE profiles specifically, however apps on top of the stack may choose to do so.
ODBC: Medblocks CDR (especially EHRbase) relies specifically on Postgres and does not support ODBC. Medblocks Ignite supports ODBC as a data persistence mode.
OpenAPI: All of Medblocks's internal APIs, including internal APIs, are exposed according to the OpenAPI specification along with Swagger UI for the same.

v) Integrations with medication imaging standards such as DICOM: Medblocks Ignite can integrate with apps that operate on DICOM and DICOMWeb. For example, a DICOM link in an EHR can be registered to open with a specific "preferred application" that can launch the DICOM viewer. We have tested this capability with OHFI (https://ohif.org/), Orthanc Web Viewer, and apps using Cornerstone.js (https://www.cornerstonejs.org). Currently, Medblocks does not provide a default implementation of the DICOM viewer or a DICOM server.

vi) OMOP and other standards used for research: Medblocks Connect can export, in real time data from openEHR and FHIR into an anonymized, normalized OMOP dataset and send it to a destination SQL system for analytics.

However, the code systems, relationships and value sets in OMOP have to be populated in a separate batch process since Medblocks Connect does not yet support pulling data from the Terminology Server to populate OMOP.

d. Describe how an external terminology server can be connected to the Solution and used both for term selection in forms/GUI and for validation of incoming COMPOSITIONs via API.What terminology server standards or products have been successfully tested and used with the Solution?

Text field

Forms/GUI: Medblocks UI can connect to arbitrary APIs to pull terminology to show when performing a search. The search component is optimized to work with multiple filters and constraints from multiple common code systems. Prebuilt plugins for Terminology APIs like Hermes, and FHIR Terminology are provided.

Validation of compositions against an external FHIR Terminology server is possible with EHRbase.

We have tested our terminology implementation against Hermes (https://github.com/wardle/hermes), HAPI FHIR and Ontoserver.

e. Describe if/how the openEHR demographic model specification is supported by your Solution now, and your future roadmap for such support.

Text field

We do not currently support the openEHR demographics model. We rely on FHIR APIs for demographics. We rely on EHRbase for all of our openEHR APIs; so if they implement a demographics API, we should be able to utilize the same.

Medblocks Ignite can work with an openEHR demographics server if it were to expose FHIR APIs for Patients.

f. Describe query mechanisms in your Solution. Clearly indicate which list item the answer refers to.

i) Describe what version of the AQL specification the CDR supports and if something from the specification is not yet supported.

ii) What parts of the RM can be reached and used as selectors and filters in queries in addition to more “normal” COMPOSITION content? For example, how can FEEDER_AUDIT, LINK, FOLDER (including the FOLDER.details ITEM_STRUCTURE) and TAGs be used to select and filter content through AQL syntax (extensions) and/or via context information like API call parameters?

Text field

We rely on the underlying CDR for AQL support.

g. Describe if and how you support use of openEHR’s TAG and FOLDER classes and mechanisms, including for what API endpoints (such as …/composition and …/query) they can be used to for example show/hide data based on if data belongs to certain FOLDERs (or it’s subfolders) or not, or based on the presence or absence of certain TAG keys and TAG values.

Text field

We rely on the underlying CDR for AQL support.

2.1.5 Tools

a. Does the Solution provide integrated version control tool support (for example Git/Github integrations) for easy retrieval and storage of assets, such as archetypes, templates, forms, and queries? If yes, please describe it briefly.

Text field

We do not provide direct integration with Git in any of our products. However, a GitHub Source Kafka Connector can be used with Medblocks Connect to pull Templates and Stored Queries from a Git repository on GitHub.

We also encourage all app developers building on top of Medblocks to use Git to store the templates along with code and make version changes to templates and code together. This enables the Medblocks UI VSCode extension to detect discrepancies between the template and the code pushed. We also provide a testing tool that can automatically test the form for errors and invalid compositions on every code push/pull request.

b. Describe how/if your products include tool support, and how well they comply with specifications, for openEHR archetype/template lifecycle management and related form lifecycle management.

Text field

The Medblocks UI VScode extension supports openEHR web templates. They provide an indicator of what has changed in the template and what has not yet been updated in the codebase.

c. Describe how your Solution supports multilingual openEHR models in data and end user interfaces. How do you provide workarounds for OPT 1.4 multilingual limitations? Describe if tool-interfaces are multilingual and can be translated and localized to Swedish.

Text field

All of our tooling works on UTF-8 character encoding and can work with multiple languages. However, we expect the web template to be exported in a single language before beginning to code (thereby circumventing the OPT 1.4 issue).

Our VSCode extension as well as Medblocks Ignite can be localized to Swedish.

d. To what extent do you support combining your Solution with components from other openEHR vendors? Describe successful tests you have done regarding this.

Text field

We take the idea of vendor-neutral applications very seriously and have built most of our solutions to work with other openEHR vendors. We will soon be testing with Cambio and establishing compatibility matrixes for all vendors.

Medblocks UI has been tested against the following openEHR CDRs:
EHRbase
Vitagroup HIP CDR
Better CDR
Solit Clouds' EHDB

Medblocks Ignite has been tested against the following openEHR CDRs:
EHRbase (full support)
Vitagroup's HIP CDR (full support)
Better CDR (full support)
Solit Cloud's EHRDB (full support)
EHRServer (openEHR AQL compatibility)

Medblocks Connect has been tested against the following openEHR CDRs:
EHRbase (full support)
Vitagroup's HIP CDR (full support)
Better CDR (openEHR API compatibility)
Solit Cloud's EHRDB (openEHR API compatibility)
EHRServer (openEHR API compatibility)

e. Describe how/if your Solution includes tool support for (ad-hoc and stored) AQL management and use, and how well they comply with (and possibly extend) specifications, for instance the examples in the list items i-iv below. Clearly indicate which list item the answer refers to.

i) Nested and/or joined AQL queries

ii) Development and testing of variables in parametric queries

iii)AQL tools and environments for authoring queries, presentation, export and visualization of AQL responses

iv) Built in configurable/programmable pre- and/or post-processing of queries and results (server and/or client side)

Text field

We rely on the underlying CDR for AQL support.

f. Describe how/if your Solution includes tool support for templates, and how well it complies with specifications for the examples in the list items i-iii below. Clearly indicate which list item the answer refers to.

i) Support for nested/embedded templates

ii) What template tools that have been tested and found compatible with your Solution

iii) Support for templates based on ADL 2

Text field

i) Medblocks UI supports any valid openEHR web template. This can have embedded templates as well.
ii) We have tested compatibility with Better’s Archetype Builder and LinkEHR.
iii) We rely on the underlying CDR for template support.

g. Describe how/if your Solution includes tool support for the examples in the list items i-v below. Clearly indicate which list item the answer refers to.

i) Developing GUI:s

ii) Data management

iii) Import, export, and migration of data, metadata and system configuration, in open well documented formats.

iv) SMART on FHIR integration

v) Mapping and conversion support other standards such as HL7/FHIR

Text field

i) Developing GUIs: Medblocks UI provides excellent support for designing any custom UIs that export simplified openEHR compositions. We also include a VSCode extension and a testing suite for the same.

ii) Data management: Authorization and Access Control can be set up in Ignite using fine-grained policies and a policy language called Rego. Incoming data from foreign systems can be validated and manually signed before being ingested into the CDR using Medblocks Connect.

iii) Import, export, and migration of data, metadata and system configuration, in open well documented formats: Medblocks Connect can be configured to pull and push data from any arbitrary system. There are reusable, well-defined mappings for certain systems like FHIR, OMOP, etc.

iv) SMART on FHIR integration: Medblocks Ignite enables running any SMART on FHIR application with support for all launch methods.

v) Mapping and conversion support other standards such as HL7/FHIR: Medblocks Connect can be configured to map data to and from HL7 FHIR and openEHR archetypes.

h. Describe how/if your Solution includes tool support for creation and use of entry forms based on openEHR templates. Clearly indicate which list item i-ii the answer refers to.

i) Which form rendering tools have been tested and found compatible with your CDR/platform?

ii) Do you supply a form builder and renderer? If yes, please briefly describe its features, for instance drag-n-drop, smart pictures (allowing annotations, term binding, graphs), low code/no code, conditional expressions.

Text field

i) We have not tested any other form rendering tools apart from Medblocks UI
ii) Medblocks UI provides a VSCode extension that can act as a form builder for people who know how to code. With Medblocks UI, you can build anything that can be built using normal HTML, CSS and JS in your app, including conditional rendering, API calls and graphs. We make it easy to use certain terminologies via plugins too. Normal javascript graphing libraries can be used alongside Medblocks UI to provide expressive and interactive charts.

With ChatGPT and GitHub Co-pilot, we are confident that soon coding itself will become the new “no-code” and anyone who has an idea can leverage Medblocks UI to build their dream applications.

i. Describe how/if your products include tool support, and how well they comply with any open specifications, for log management, such as alarms and access logs.

Text field

All of our products are instrumented. They output logs to STDOUT and provide Tracing and Metrics compatible with OpenTelemetry and Prometheus. This can be leveraged by infrastructure managers to set up alarms.

2.1.6 IT and Information Security

a. Describe what kind of IT security features are implemented in your Solution, for instance support for securing API, data at rest, data in transport, data in operation, data removal, and logging and audit.

Text field

API Security is handled by the policy decision engine and no request goes in through the infrastructure otherwise. This can be configured using Rego.

All data is stored in a Postgres database, and this can be configured to be encrypted at rest with most major cloud providers like AWS, Azure and Google.

Data in Transport always is encrypted via HTTPS (for REST APIs) or SASL (for Kafka).

Logging and auditing: All requests are logged, and both EHRbase and HAPI JPA provide ATNA logging.

b. State if there are any relevant IT security certifications for your Solution, such as ISO27001, ISO27018.

Text field

We have obtained WASA (Web Application Security Audit) and Safe to Host certificates on multiple clouds for our stack.

c. Describe what kinds of authentication, authorization and access methods your Solution supports, for instance external IDP, role-based access control, privileged users control, just-in-time access.

Text field

Authentication methods supported: Cookie authentication, OAuth2.0 (including RFC7523 JWT profiles), and Rotating Bearer Token authentication.
For Authorization, our solution uses Open Policy Agent - https://openpolicyagent.org/. The policy decision language Rego can be used to configure the most common access control mechanisms like RBAC, ABAC while also providing the ability to call external APIs like an IDP before granting access. Both REST APIs and Kafka APIs are secured with Open Policy Agent.

d. Do you use supply chain risk management strategies/tools, such as SBOM? Describe how you mitigate risks associated with development, maintenance, acquisitions and, sunsetting of systems/components and/or services? How are risks and mitigating actions documented and what is your strategy for enforcing compliance?

Text field

We use Docker SBOM when building a container to scan for security vulnerabilities. All container images also use Content Trust Tags for security in transit to the cluster that'll deploy the image.

Our risk management approach involves:
Identifying and assessing potential risks.
Developing mitigation plans for high-priority risks.
Continuously monitoring and reviewing risks and mitigation efforts.
Documenting risks and actions in a risk register.
Enforcing compliance through policies, audits, and training.
This process ensures the resiliency of our systems/components and services throughout their lifecycle.

2.1.7 Training, documentation and consultant services

a. Describe the availability of course or on-line training for administrators, technicians, tool users, software developers, EHR end-users (if you provide modules/products for end-users).

Text field

We offer an openEHR course covering clinical modelling, openEHR REST API, Template and Compositions, and querying data using AQL. The course also provides hands-on exercises and practical examples to reinforce learning and instruction on setting up Medblocks CDR with EHRbase in the cloud. Participants will learn how to create openEHR templates from real-world hospital forms, and post compositions to openEHR servers, and use Archetype Query Language (AQL) to query this data back. Link: https://learn.medblocks.org/

Our public documentation can be viewed at https://docs.medblocks.org/ where we cover Medblocks UI, and Medblocks CDR setup with Docker, along with tutorials and guides on how to build apps using Medblocks UI.

We also provide extended API documentation, and reference guides for our customers to build and deploy their solutions.

b. Describe which kind of product documentation you provide, for instance user manuals, installation guides, system administration guides.

Text field

We provide installation and system administration guides for Medblocks CDR, Medblocks Ignite, and Medblocks Connect. For Medblocks UI, we provide extensive guides and user manuals for building apps on our documentation page.

c. Do you offer consultant services for implementation, configuration and/or development?

Text field

Yes, we offer consultant services for implementation, configuration, and development. We offer a wide range of services to healthcare organizations and developers looking to implement openEHR, FHIR, and other terminology standards.

A) Medblocks Setup and Configuration:
Setup, deployment and administration of Medblocks stack on private/public cloud.
Configuring Medblocks Ignite with the right access control, workspaces and applications.
Integration pipelines leveraging Medblocks Connect, Kafka Streams and KSQL.
Analytics Dashboards on top of external SQL representations of data exported by Medblocks Connect

B) General openEHR Services:
openEHR clinical modeling and template designing
Building and deploying openEHR applications based on Medblocks UI
Writing and optimizing AQLs
Mapping openEHR archetypes to profiled FHIR resources

C) General FHIR Services:
Building, Deployment or Modifying SMART on FHIR applications
Writing Implementation Guides and Profiling FHIR Resources
Integration with HL7v2, CCDA to receive data from existing EHRs that support these standards
Mapping profiled FHIR resources to openEHR archetypes

D) General Terminology Services on SNOMED CT, RxNorm, ICD10 and LOINC:
Mapping internal value sets to standard value sets
Identifying and refining value sets
Setting up and deploying Terminology Servers with custom value sets

3. Part 2: Demonstration

3.1 Demonstration sessions

The second part of this RFI consists of a demonstration session where selected respondents, that meet the qualification criteria described below, are invited to demo their Solution.

3.1.1 Qualification and prioritization criteria

To be qualified for a demo time you will need to demonstrate a Solution that is helpful when creating applications, capturing or storing clinical data based on openEHR standards, that is, not just general integration or CDR products. If there is competition for available presentation/demo slots, the written responses to above listed questions will be used as prioritization criteria.

A maximum of six (6) suppliers will get an invite to a demo session.

3.1.2 Purpose

The purpose of the demo is to show how your Solution meets the needs of the stated target groups and the user stories described below.

3.1.3 Dates

The demo sessions are held on May 31, June 1 and June 2. June 5 is reserved as an extra date for back-up purposes. Each demo is limited to two (2) hours.

3.1.4 Format

The demo is an online two (2) hour session via Zoom. The sessions are recorded and made public on Youtube when all suppliers have held their sessions.The purpose of publication is to help other organizations interested in openEHR systems.

A demo session is on the following format:

- Short introduction of company and Solution and what is going to be presented in the demo (maximum 2 minutes)
- Demo based on target group descriptions and user stories
- Discussion with questions and answers (minimum 30 minutes)
- Optionally and on request, the recording can be stopped for the last 15 minutes of the discussion, if there are parts that should not be made publicly available.

Additional county councils may later join the RFI and attend the demonstrations as listeners.

3.1.5 Instructions

To reach business impact goals and purposes, it is essential that a procured solution meets the needs and expectations of the different target groups that will use the openEHR Solution. A number of essential target groups are identified – Platform administrator/technician, Application and content developer/administrator, Super user, External actor, Application end-user, and Newbie.

Each target group has a description and some of them have one or several user stories that highlight aspects of the target group that we think would be interesting for a demo. Use these descriptions and user stories as a basis for your demo. You are not expected to demo everything.

During the demo session, please refer to which target groups/user stories you are demonstrating.

3.1.6 Application and Content Developer/Administrator

This is an informatician, a software developer or a system/content manager. She develops applications, builds integrations, does information modeling and form building, and designs queries for information retrieval. She is also responsible for maintenance of applications, information structures and content. She gives technical support and help to other users of the openEHR tools. When functions that are more complicated are needed in an openEHR-based application, the application and content developer/administrator takes care of it. She is an advanced user with high demands on smart functions in the development tools.

User stories based on Application and content developer/administrator:

1. As an informatician I want to connect an external terminology service to make sure that the terms within the data are consistent with appropriate terminology standards and valuesets/subsets.
2. As a healthcare system developer I want to integrate software to be able to store and retrieve medical data in an openEHR EHR system alongside other healthcare system vendors.

3. As a healthcare developer working on a SmartOnFhir application I want to be able to access part of the openEHR information as standard FHIR API.

4. As an administrator or developer I want to configure or be able to create solutions for collecting IoT device measurements from patients. This includes

   a) data from medical devices that we as healthcare providers have provided, support and collect data from.

   b) data from patients’ privately purchased devices (smartwatches, blood pressure meters etc) that they may have connected to apps in their Android and iOS devices - this transfer may be initiated by the patient without being actively requested by healthcare (e.g. before a visit). Such data should when stored be possible to identify as patient reported so that it can be logically separated from other data.

   c) where the data was created and by which person and device.

5. As an administrator or developer I want to configure or be able to create solutions for collecting data from patient-reported forms, photos, and videos.

6. As an administrator I want to be able to referens see Appendix A

   a) create/define metadata attributes to personal data so the Solution can be configured to meet our needs.

   b) add/update metadata for a specific piece of personal data.

   c) add/update metadata to personal data as a bulk update, e.g. for all compositions created at a certain organizational unit.

   d) use metadata to create functions managing what information a user has access to e.g. in an overview of an encounter of a patient who received specialist care.

3.1.7 Platform Administrator/Technician

This person works in the IT department, has a technical education and a few years working experience. It is his job to ensure that the platform and the development tools are sound and up and running. The platform administrator/technician is an advanced user that needs powerful tools for administration of the openEHR platform. He wants to have full control and overview, and efficient configuration and error handling and system diagnostics tools. The openEHR platform is not his only responsibility at work; there are many other systems as well, so he values extensive system documentation. Sometimes he needs support, and he is grateful that he gets it quickly.

User stories based on Platform administrator/technician:

1. As a server-admin, I want to use supporting functions so that I can carry out technical troubleshooting.

2. As a first line support tech, I want to view the system's operational status via web-UI so that I can at a glance check if there are any issues.

3. As an administrator I want to manage access-rights, e.g. configuring rules, roles and access control policies, so that I can restrict access to information based on user context and information attributes.

3.1.8 Super user

The super user is a nurse, a physician or a researcher at a healthcare unit and is interested in how new technical solutions can be used to improve the patient care, working processes, and gaining new medical knowledge. The super user maintains existing forms and templates in the openEHR-based applications that the department uses. The super user really prefers to be able to solve problems himself if possible. But in rare cases it gets a bit too complicated, for instance when programming skills are necessary or when a new template is needed, and then the super user contacts application and content developer/administrator for help and they cooperate on the solution. The super user also generates reports from the healthcare systems that the care department needs; often it is standard reports that are generated repeatedly, but sometimes a special report is needed.

The super user does not use the openEHR tools on a daily basis, but is more of a "burst" user where intense use is combined with periods of little use or no use at all. This pattern of use means that he might not ever be fluent in how to use the tools.

Since the super user does not have deep technical knowledge it is important that the tools he uses to update forms and templates are easy to use. It is also important for the super user that it is easy to get an overview of which templates and forms that the clinic is using, that version handling is easy and straightforward, and that efficient search and filtering tools are available. The super user also needs a comprehensible report generation tool.

User stories based on Super user:

1. As a clinician, I want to build and design a dynamic form, based on existing templates, with conditional form field display logic and automatic calculations, for structured documentation.

2. As a researcher, I want to create reusable methods to search, collect and present data, for example regarding a certain patient group/diagnosis and only for a specific gender at a certain age.

3. As a clinician, I want to design and generate ad hoc reports, from data collected through a form.

4. As a new employee (or occasional “burst” user) I need user friendly, and intuitive easy to use tools and graphical user interfaces.

3.1.9 Application End-User

Application end-user is a healthcare clinician or a citizen. He wants to enter and retrieve information from and to the health record system. The application end-user has no interest in the technical aspects of the applications they use; the important thing is that the applications support what they want to do in a smooth way. This may include that the applications are always available, or that only information that is relevant in the particular context is shown. In some situations, it may be of interest for the application end-user to switch language in an application. Since he could be any citizen, it might be the case that he has some kind of disability, for instance impaired vision, and is in need of things like enlarged text or textual descriptions of images. Thus, his needs concern the results of using the openEHR platform and development tools; as long as the resulting applications are stable and good, he is happy.

User stories based on Application end-user:

1. As a clinician, I want to have a Clinical Decision Support and process support functionality, to improve the quality of care and reduce risks.

3.1.10 External Actor

External actor is a company, a student, another healthcare region, or a researcher. The external actor delivers applications or content. The external actor has no direct access to the internal systems and uses her own development tools. It is important for her that a full range of REST APIs is available, and she values extensive system documentation. It could be convenient for her to use openEHR tool licenses for a limited period when developing on behalf of a healthcare region.

3.1.11 Newbie

The Newbie is a nurse or a physician at a hospital, but may also be an informatician or a software developer. Newbie has a few years working experience but no or little knowledge of openEHR. Now is the first time Newbie takes part in maintaining existing forms and templates or in developing a new openEHR-based solution. It is important for the Newbie that the tools for developing forms are easy to learn and that the user documentation is pedagogical and covers all common use cases and functions. Some kind of introductory training to get started would help Newbie a lot.