

DIPS openEHR suite

An introduction

Status

Bjørn Næss

31.05.2023



ENABLING EFFICIENT HEALTHCARE

History

Date	Version	Author	Description
31.05.2023	1.1	Bjørn Næss/Kjetil Jørgensen	Update with openEHR ETL Service
20.09.2021	1.0	Bjørn Næss	Written as part of the reply to the Catalonian RFI

Approval

Dato	Versjon	Godkjent av	Stilling
31.05.2023	1.1	Bjørn Næss	Section manager Bifrost
20.09.2021	1.0	Bjørn Næss	Section manager Bifrost

© 2023 DIPS AS.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of DIPS AS.

DIPS AS
Postboks 1435
8037 Bodø
Norway
dips.no
+47 75 59 20 00

Innhold

1	Preface	3
2	Introduction	3
3	DIPS EHR Store	4
3.1	DIPS EHR Store Architecture	4
3.1.1	HTTP/Rest Interface	5
3.1.2	OpenEHR Reference Model	5
3.1.3	Database	5
3.1.4	Apache Solr	5
3.1.5	Deployment	6
3.2	Arena Knowledge Repository	7
3.2.1	VAQM	8
3.2.2	Charts definitions	8
3.2.3	Folder definitions	8
3.3	EHR Store Administration (dashboard)	8
4	DIPS Forms	8
4.1	Forms Specification	9
4.1.1	Form Extensions	9
5	Arena Rules	11
5.1	GDL – Guideline Definition Language	11
5.2	Functions	11
6	DIPS EHR Craft	12
6.1.1	Software requirements	12
6.2	Clinical modelling	12
7	DIPS openEHR ETL	13

1 Preface

This document is written as a summary of the different parts of the products, tools and components DIPS has developed according to the openEHR specification. The RFI asks for information regarding the components in different sections. This document is written to make an overview of the platform/suite for openEHR provided by DIPS.

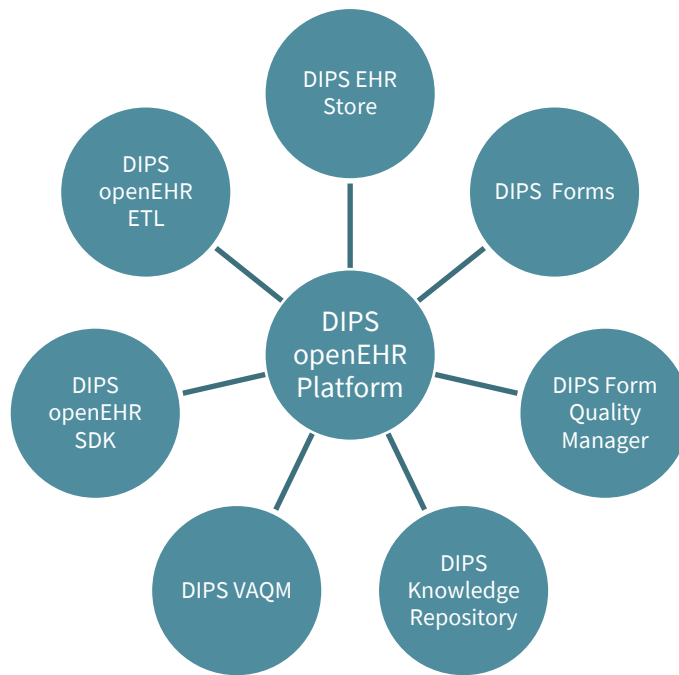
2 Introduction

Since 2010 DIPS has developed software and solutions based on openEHR specifications and information models (archetypes/templates). The primary target has been DIPS Arena; the hospital information system (HIS) used by our Norwegian customers. Since October 2014 openEHR has been in production in Norway. The software, competence and skills have evolved continuously from this. Today DIPS can present a well-proved openEHR CDR, mature software libraries and good tooling for no/low code development of clinical applications.

DIPS Arena is a complete hospital information system covering patient administrative features (resources, scheduling, appointments, users, and access control), laboratory, medication and of course EHR. DIPS EHR was the enabler for the world's first paperless hospital back in 2001.

For this request for information, we chose to focus on the core openEHR solutions, the CDR and the tooling. We will not describe the clinical applications built on the openEHR platform and neither the proprietary solutions making use of openEHR inside the application domain. More details on these might be added in an eventual procurement depending on the needs from the customer.

DIPS openEHR Suite is a concept and product for a computable health information platform. The purpose of the platform is to give developers tools, components, and services to develop functionality fast, consistent, and standardized. The platform contains several modules as shown in the figure below.



3 DIPS EHR Store

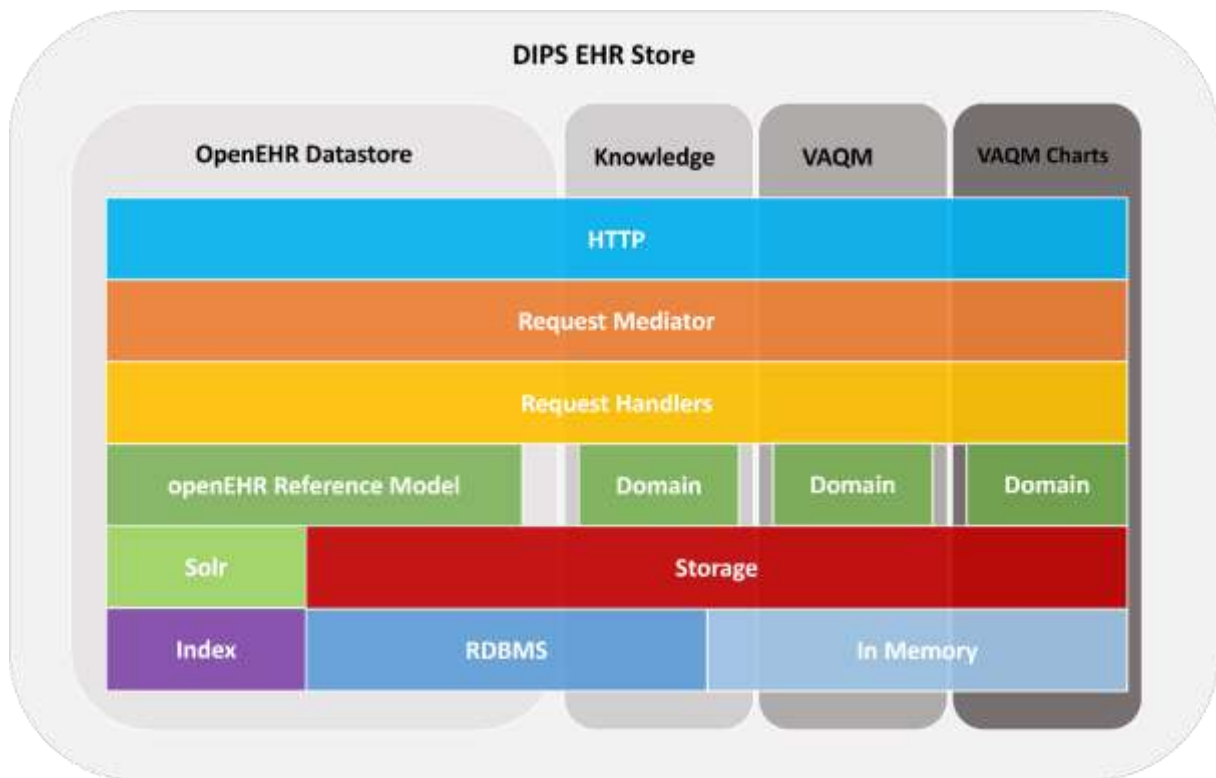
DIPS EHR Store is an openEHR server implementation. It is the key component of DIPS openEHR Platform. The central function of the openEHR server is to manage Compositions. The composition endpoint has functions for basic CRUD operations on entries. All compositions in an openEHR system are bound to a subject of care, which is the EHR object with the primary identifier of the subject of care. The EHR service implements operations to create and retrieve EHR information.

One of the central elements of the openEHR server is the ability to query for structured data across compositions and EHRs. The archetype query language (AQL) is a powerful language to express queries to retrieve data from saved compositions. The query service accepts AQL queries as argument to retrieve and filter data.

On top of the OpenEHR specifications we have implemented a virtual archetype query model (VAQM). VAQM can be compared to *stored queries* but includes more features and enables simplified application development.

3.1 DIPS EHR Store Architecture

In the following, we provide an overview of the architecture of DIPS EHR Store. This is extract of the full documentation, which will be provided on delivery.



3.1.1 HTTP/Rest Interface

Interaction with DIPS EHR Store is through the REST interfaces. The primary REST services are EHR, Composition and Query. These service endpoints are compatible with the better and new openEHR service specification: http://www.openehr.org/releases/ITS/latest/ehr_restapi.html.

3.1.2 OpenEHR Reference Model

DIPS EHR Store and its components are compliant with openEHR Reference model in current version. DIPS is an active member in the openEHR community, and we update the implementation as the specification evolves.

3.1.3 Database

DIPS EHR Store is database agnostic. We have most experience using Oracle. DIPS Arena uses Oracle and Oracle functions. EHR Store has implemented support for SQL Server.

For a developer or clinical modelling setup, DIPS EHR Store may also use filesystem as database. This is of course not recommended for production data.

3.1.4 Apache Solr

DIPS EHR Store uses Solr for indexing of data. For production data it is necessary to have running Solr instance so that the server functions correctly.

There are two possibilities for configuring connections to Solr:

- Solr Standalone – just configure the server with the URL to the Solr server.
- Solr Cloud – uses Apache Zookeeper to enable highly reliable distributed coordination

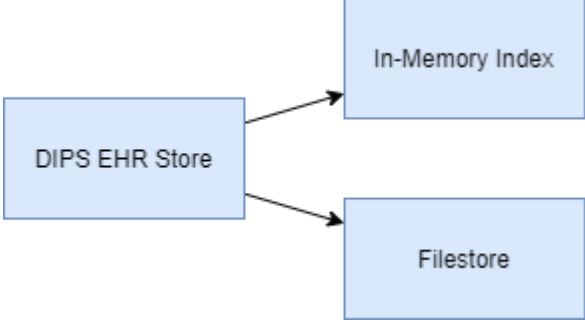
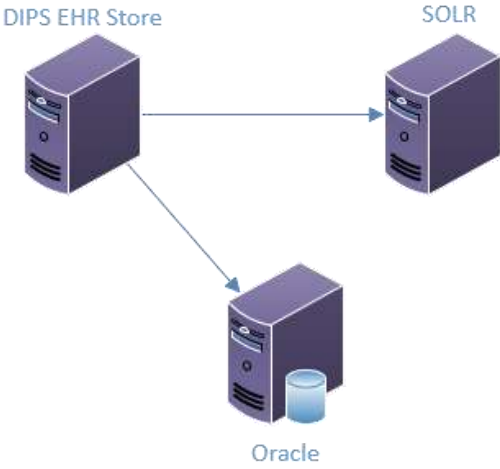
Apache Solr is not transactional, so it is necessary to do some bookkeeping on inflight transaction to handle failure regarding indexing in Solr. In normal, non-failure, operations the indexing to Solr is synchronous in terms of the client call. If indexing in Solr fails DIPS EHR Store switches to a BASE consistency model, which basically means that it is eventually consistent.

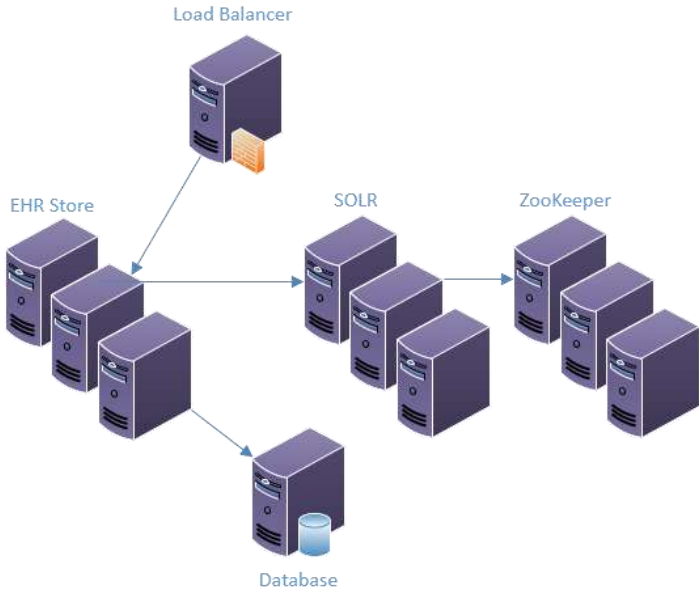
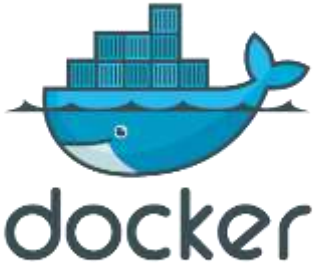

For large scale deployment it is needed to handle index synchronization. The index synchronization feature in DIPS EHR Store was created to handle a multi-environment setup.

The detailed description on Solr indexing and index synchronization is provided on delivery.

3.1.5 Deployment

DIPS EHR Store is a modular server implementation. It is designed to function from an in-process usage and into the most complex deployments in the largest hospitals. Below we provide some examples. The specific deployment model for this delivery must be worked out in cooperation.

<p>In-process mode</p> <p>For simple deployment like developer machine or embedded in EHR Craft we may run DIPS EHR Store as an in-process server. Data is then stored on filesystem.</p>	
<p>Simple deployment</p> <p>One host may have all processes installed.</p> <p>When load is high and there is lots of data customer may deploy to several hosts.</p>	

<p>Complex deployment</p> <p>For large-scale installation, DIPS EHR Store support more complex deployment with several datacentres and load balancing the traffic between several nodes.</p>	
	<p>DIPS EHR Store supports Docker.</p>
<p>CLOUD</p> 	<p>DIPS EHR Store have been tested on Microsoft Azure. The deployment is then based on Docker containers for each component; DIPS EHR Store, Apache Solr and Database.</p>

3.2 Arena Knowledge Repository

Arena Knowledge Repository is a module in DIPS EHR Store to manage different clinical models. Archetypes, Templates and Guidelines (GDL) are models that are edited by using tools available from openEHR.org. In addition, we support a set of other knowledge resources, described below:

- Forms (see separate chapter on DIPS Forms)
- VAQM
- Folder definitions

The REST API for Arena Knowledge Repository is a superset of openEHR definitions REST API.

3.2.1 VAQM

VAQM (Virtual Archetype Query Model) is a model of a structured query and presentation model on top of openEHR data. The model is designed with two main purposes in mind:

- Be able to do define structured and reusable queries for simplified application development
- Compact definition of the presentation of openEHR data

VAQM defines a set of paths that may be used to express the presentation of the data. Below is an example of VAQM expression for blood pressure. This expression is compact and human readable, which makes it easy to build applications on the openEHR reference models.

```
$bp.systolic/magnitude / $bp.diastolic/magnitude $bp.systolic/units
```

VAQM editor is a tool to create and edit definitions. This editor is embedded in EHR Craft.

A full description of the VAQM format will be provided on delivery.

3.2.2 Charts definitions

Based on VAQM we have developed a model to define chart datasets.

3.2.3 Folder definitions

FOLDER is a class defined in openEHR. DIPS have developed a specification to be able to create and commit folders into the openEHR server. EHR Craft has an embedded renderer for FOLDER instances.

3.3 EHR Store Administration (dashboard)

DIPS EHR Store Administration is a lightweight module with a user-interface and services to do basic system configuration and system status operations. The admin module contains functionality to monitor system status, logging, memory usage, response time, etc. in the system.

All functions for system status and response time are also available in the REST interface provided by DIPS EHR Store.

4 DIPS Forms

The Forms layer provide a quick and scalable solution to create input forms with advanced functionality without writing code. This is especially useful when creating several similar, but not identical forms for different clinical situations. The forms provide a possibility for extremely simple and cost-effective prototyping as well as development and testing in close collaboration with end-users.

The Forms module consist of a form designer and a form renderer.

DIPS Form Designer lets users without programming skills create advanced forms with dependencies and calculations based on OpenEHR templates.

DIPS Form Renderer is the component that renders a user-interface with layout, dependencies and validation based on the form definition created with Form Designer. DIPS maintain two Form Renderer:

- A .NET/WPF renderer which is embedded in DIPS Arena
- A WEB based renderer which is integrated in DIPS Wall and DIPS Mobile

The format of the forms is currently not a formal part of the openEHR specification but as vendors, both Better and DIPS use the same basic format for design and rendering of forms. The form specification is in process to be made open source.

4.1 Forms Specification

The following is a short description of the form specification. A form is defined by the following parts:

- Form description – the structural definition of the content. This is where the validation, user-input components, language, etc. is defined
- Form layout - defines how the components from form description should be placed in the form
- Form dependencies - defines the dependencies between elements in the Compositions.
- Form manifest – metadata with information about the current form. Includes name, version, timestamp, etc.
- Form scripts – defines the form logic as JavaScript.

The different parts are defined in files that are packaged in a ZIP archive.

4.1.1 Form Script API

The API for DIPS Form Script is open-source and available here:

<https://github.com/DIPSAS/ehrcraft-form-api> , and there is also a generator/compiler to transform typescript into vanilla JS to be run inside the Form renderer (<https://www.npmjs.com/package/generator-ehrcraft-script>).

4.1.2 Form Extensions

Tags and Annotations are attributes where the application adds extension to the basic form definitions. Tags are simple string-based keywords. Annotations are key/value-based attribute definitions. The semantic of Tags and Annotations is “only” known by the specific application using the form.

The development of Arena EHR forms extension is based on the experience of using the platform to create no-code applications as part of our own EMR suite and provides a large set of extra features on top of the OpenEHR specification and joint forms definition. This makes it possible for customers to expand the form definitions with functions needed for a specific and complex application, without breaking the interoperability on template and archetype level.

To illustrate the properties for DIPS openEHR Platform’s form functionality, some examples of Tags and Annotations that governs form functionality in Arena EHR Platform are listed below. Currently a total of 28 annotations, 10 tags and 103 functions are supported. The total list can be shared on request.

Tag	Description
IncludelnMasterDetail	If ShowAsMasterDetail set on a container then this tag will define which elements to include in the table. If no IncludelnMasterDetail is set then all elements in the container will be columns in the table.
ShowAsMasterDetail	Shows a container (CLUSTER/ENTRY) as a master-detail view. The active container is displayed as defined in the form, and the other entries are displayed in a table below. This is useful to make a compact form with many entries of the same type.
ShowAsWizard	When applied the form will be presented for the use one container at a time, with buttons for next and previous, the keys for this annotation is edit and "null". Edit will present the form as a wizard only when editing

4.1.2.1 Annotations

Annotation	Description
alertType	Defines how a container should be displayed. Normally this is used to change background color. Allowed values are: TIP, NOTE, INFO, INFORMATION, IMPORTANT, WARNING, SUCCESS, ERROR, EXCEPTION, TITLE
aql	Defines the AQL for reuse operations. Se SELECT annotation below.
calc	Use in fields for the purpose of showing results of functions (See function chapter)
constraint.func	Defines the constraint of a field. This could for instance be a past or future date. When using this annotation, the value could be a calculation that sets the constraint.
select	For reuse of data based on the AQL annotation above. Syntax {#variable:<path to instance>}
showAsWizard	Form works like a wizard with one container for each step. Renderer displays buttons for next and previous
stepInterval	Controls the precision level of any step interval in a forms element containing a DV_QUANTITY or DV_COUNT. Could be an integer or double.

5 Arena Rules

5.1 GDL – Guideline Definition Language

GDL is deprecated in DIPS EHR Store

DIPS implemented the openEHR GDL¹ specification and exposed it through a REST service provided by DIPS EHR Store. We used the GDL editor, available from openEHR.org, to create and edit Guides (GDL files), import the guides to DIPS EHR Store and execute the guides using the REST interface either directly or by using EHR Craft.

We deprecated this solution and saw a need for more process-oriented decision support like the Task plan specification. For decision support we currently use script logic within Forms and expression logic in VAQM.

5.2 Functions

Form Renderer has an embedded calculation/function engine. Using “Excel style” functions, the form editor can inject calculations into the form through annotation. This functionality is used widely in DIPS EMR system.

Visit openEHR WIKI for a list of DIPS functions (currently 103)
(<https://openehr.atlassian.net/wiki/display/spec/openEHR+Rules+specification>).

The underlying engine is very flexible for extensions. We are adding new functions when needed. Some of the last functions is calculation of openEHR DURATION. The table below gives this example:

Function	Description
DURATION_TO_MINUTES (iso8601_duration_text)	Returns the numeric value of the duration in minutes.
DURATION_TO_DAYS (iso8601_duration_text)	Returns the numeric value of the duration in days.

¹ <http://www.openehr.org/releases/CDS/latest/GDL.html>

6 DIPS EHR Craft

DIPS EHR Craft is a desktop application developed to support the needs for clinical modelling and application development. The application embeds the tools and components described above. As a result, it's like an openEHR IDE which simplifies the building and testing of clinical models like forms, queries, VAQM, charts, etc. EHR Craft is the tool made for developers and clinical modellers can be easily distributed and is automatically updated through a web-service. This allows customers to distribute the development IDE to users outside the network domain. EHR-Craft can be used on top of both the internal REST API from DIPS EHR Store and openEHR REST API.

EHR Craft contains lots of functionality. Here we will list some:

- Debugger for Forms
- Test manager for Forms
- Embedded file-storage for Compositions to let test data follow source code
- Composition generator from Operational Templates
- Loading and running form-script from file
- AQL editor
- AQL query client which connects to any openEHR REST API
- openEHR ETL Definition generator
- VAQM editor

6.1.1 Software requirements

The following software components is required to run the DIPS EHR Store:

- Windows Server 2008R2 or newer
- Database - Oracle Database Server (no client needed) or SQL Server
- Apache Solr 8.4.2 or newer (standalone or cloud)
- .NET Core 3.1 (.NET Runtime 3.1)

EHR Craft (the IDE) needs Windows 10.

6.2 Clinical modelling

DIPS use the modelling tools available from openEHR. When developing clinical applications, we will always use reviewed archetypes. We work close with the National Editorial Committee for Archetypes in Norway in this process.

The following tools is used:

- Archetype Editor (Ocean Informatics, freeware)
- Template Editor (Ocean Informatics, freeware)
- CKM (arketyper.no, openehr.org/ckm)

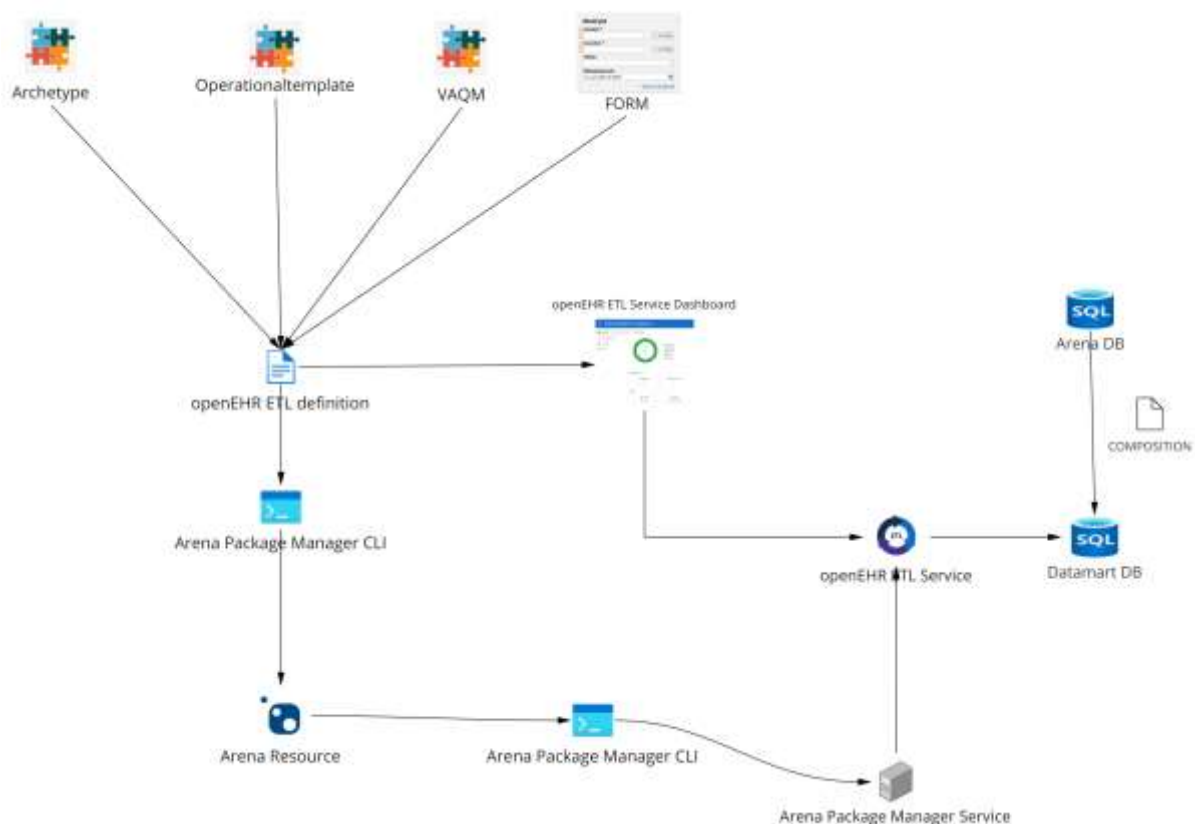
7 DIPS openEHR ETL

openEHR is the best approach to model clinical data for use within application and to carry the semantic flexibility required in health and care. Structured clinical data is an essential for quality improvements, research, and other secondary use of data in the EHR. AQL is the query language close to the clinical models. Still, it has a lack of features related to analytics and reporting. To meet this requirement DIPS has developed a service for dynamic ETL using annotated AQL definitions.

DIPS openEHR ETL service is the latest component in our openEHR platform, and it is currently being deployed to our Norwegian customers.

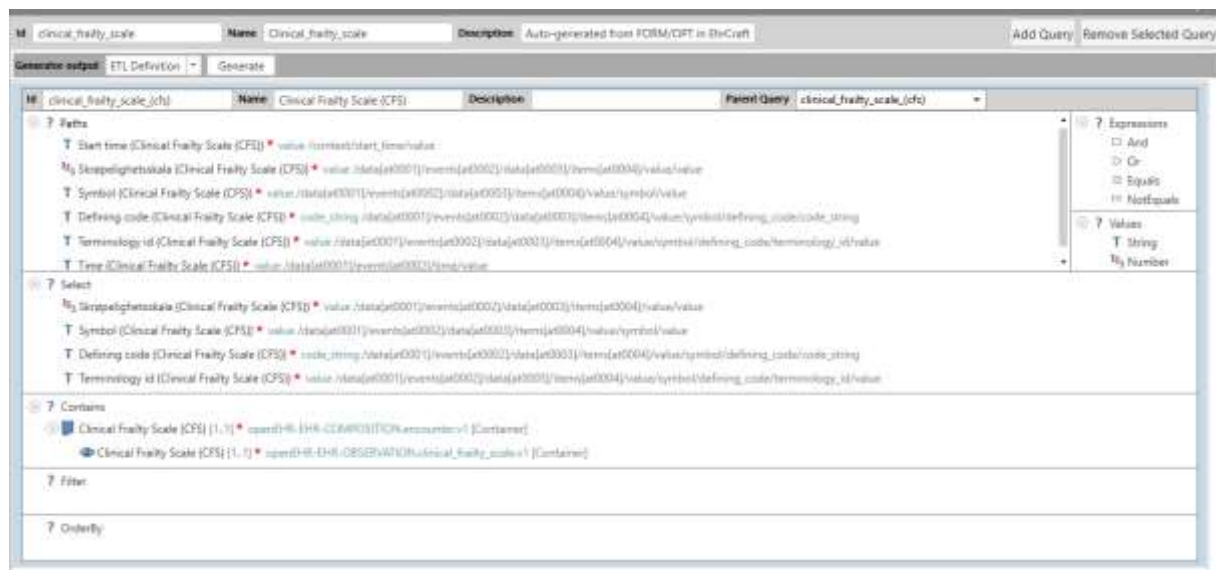
The service is a standalone component using openEHR API with AQL to extract data from the CDR, transform data into relational structures and finally load the target database. To make this developer friendly we have created an annotated syntax above AQL definitions – “openEHR ETL definitions”. The purpose of this syntax is to define the transformation as close to the AQL query as possible, and still be compatible with standard AQL. With tooling such definitions can be created from different sources like archetypes, operational templates, VAQM and openEHR forms.

Using our package management system, the definitions might be deployed into the openEHR ETL Service. The illustration below gives a summary of the setup.



EHR Craft Studio has integrated tooling to generate openEHR ETL definitions. There is a no-code UI to generate AQL and openEHR ETL definitions.

To give illustrate this we will in the following give a simple example using the Clinical Frailty Scale.



Below is the generated definition from the UI.

```
--@etl.definition(schema_version=1, id=no.dips.clinical_frailty_scale, version=1.0.0,
namespace=at_clinical_frailty_scale, name="Clinical_frailty_scale", description="Auto-generated from FORM/OPT in
EhrCraft", authors="DIPS AS", copyright="DIPS AS", tags="ehrcraft-generated")
--@etl.query (id=clinical_frailty_scale_, name="Clinical Frailty Scale (CFS)", description="Auto-generated from
FORM/OPT in EhrCraft", target_name=clinical_frailty_scale)
SELECT
--@etl.query.column (type=Integer, target_name=skropelighetsskala_s, index=false)
o/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value/value,
--@etl.query.column (type=String, target_name=symbol_s, index=false)
o/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value/symbol/value,
--@etl.query.column (type=String, target_name=defining_code_dc, index=false)
o/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value/symbol/defining_code/code_string,
--@etl.query.column (type=String, target_name=terminology_id_ti, index=false)
o/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value/symbol/defining_code/terminology_id/value
FROM
COMPOSITION c[openEHR-EHR-COMPOSITION.encounter.v1]
CONTAINS OBSERVATION o[openEHR-EHR-OBSERVATION.clinical_frailty_scale.v1]
```