D5.8

Design of the Data Integration Platform - v2

ABSTRACT

This document describes the fundamental software components - together referred to as *Data Integration Platform* - that provide the technological basis for the semantic conversion and translation of electronic health records across languages and across local and national healthcare standards.

Delivery Date	July 9th, 2021
Work Package	WP5
Task	T5.3
Dissemination Level	Public
Type of Deliverable	Report
Lead partner	UniTN

InteropEHRate project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826106.

This document has been produced in the context of the InteropEHRate Project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826106. All information provided in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose.

CONTRIBUTORS

	Name	Partner
Contributors	Gábor Bella	UniTN
Contributors	Simone Bocca	UniTN
Contributors	Alessio Zamboni	UniTN
Reviewers	Thanos Kalligeris	ISA
Reviewers	Shayan Amini	A7

LOG TABLE

Version	Date	Change	Author	Partner
0.1	2020-08-01	Created as copy of V1 release, added section 2	Gábor Bella	UniTN
0.2	2020-08-20	Provided updates on the platform implementation	Alessio Zamboni	UniTN
0.3	2020-08-20	Updated document according to platform implementation changes	Simone Bocca	UniTN
0.4	2021-02-01	Updated with IHS information	Simone Bocca	UniTN
0.5	2021-05-20	Reviewed and updated API endpoint details	Simone Bocca	UniTN
0.6	2021-06-10	Provided details w.r.t. communication with the hospital information system	Gábor Bella	UniTN
0.7	2021-06-15	Added specs on input knowledge formats	Gábor Bella	UniTN
0.9	2021-06-15	Version ready for internal review	Gábor Bella	UniTN
0.91	2021-06-17	First internal review	Shayan Amini	A7
0.92	2021-06-25	Internal review	Thanos Kalligeris	ISA
0.99	2021-07-01	Ready for final QC and submission	Gábor Bella	UniTN
1.0	2021-07-04	Quality check	Argyro Mavrogiorgou	UPRC
Vfinal	2021-07-09	Technical and final review for submission	Francesco Torelli Laura Pucci	ENG

ACRONYMS

Acronym	Term and definition
API	Application Programming Interface
CSV/TSV	Comma-Separated Values / Tab-Separated Values: simple machine-readable tabular file formats.
EHR	Electronic Health Record information system (e.g., as provided by a hospital).
IHS	InteropEHRate Health Services: a high-level software component (a collection of libraries) that provides high-level EHR translation and conversion services to end-user applications.
IHSI	IHS Interface: the application programming interface offered by the InteropEHRate Health Services to hospital information systems.
IHT	InteropEHRate Health Tools: a set of interactive helper tools that are used by hospital employees (data scientists) to set up and maintain the EHR data integration system. These tools are outside the scope of the project deliverables and thus are not fully specified in any deliverable document.
JSON	JavaScript Object Notation: standard, computer-readable, tree-structured file format.
LEI	Legacy EHR Interface: the API provided by a hospital information system to be called by the InteropEHRate Health Services.
OWL	Web Ontology Language: a standard representation format for computer-processable knowledge.
Smart Health Data	The interoperable, multilingual, standard, FHIR-based representation of health data, as defined and used by the InteropEHRate project.
XML	Extensible Markup Language: machine-readable tree-structured file format.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Scope of the document

This deliverable provides the specifications of the *InteropEHRate Health Data Integration Platform* (abbreviated as *Platform* in the rest of the document). The Platform is a base system that provides low-level and mid-level functionalities for a *semantic* - i.e. meaning-level - integration, conversion, and translation of electronic health records of patients. Other InteropEHRate deliverables define how these functionalities are used by higher-level *conversion* and *translation services* to convert and translate EHRs [D5.10] and how such converted EHRs are presented to healthcare professionals [D5.5].

1.2. Intended audience

The Platform is a low-level software component typically used by higher-level health IT services rather than directly by end users. Consequently, this deliverable is primarily aimed at technical audiences, such as the IT managers/staff of a hospital or third-party developers of semantic services on top of EHR data, who wish to understand:

- how InteropEHRate Health Services work internally;
- how to build services on top of the Platform.

The deliverable focuses on the *architecture*, components, and interfaces of the Platform. While it also presents how the Platform can be technically deployed within the context of a healthcare institution, it does *not* present in detail the process by which the Platform is filled with formal health knowledge in order to implement specific data conversion and translation tasks. In order to understand the detailed process, the reader is referred to [D5.10] (data conversion) and [D5.11] (data translation).

1.3. Structure of the document

Section 2 motivates the Data Integration Platform with respect to the InteropEHRate scenarios. Section 3 presents the high-level architecture of the Platform, its components, and its interfaces both on the input and the output side. Sections 4 and 5 present the two layers of the Platform: the Knowledge Layer and the EHR Data Layer, respectively. The two sections explain the role of each layer, its principal components, and the principal API endpoints that they provide to the services using the Platform. Section 6 presents the IHS Controller component, which is an adaptation wrapper around the Platform, and the APIs it provides. Section 7 describes the health record conversion process using the functionalities provided by the Platform and the IHS Controller. Finally, section 8 provides conclusions and next steps.

1.4. Updates with respect to previous version (if any)

On the whole, the overall design of the Platform has not changed since the first version of this deliverable. Improvements have been made to the specifications by adding more details wherever necessary, and by improving the overall quality of the document:

- **Section 2:** this newly introduced section situates the need for the Data Integration Platform within the usage scenarios covered by the InteropEHRate project, providing motivation for the contents of this deliverable.
- **Sections 4 and 5** were updated according to a partial redesign of the Platform APIs, that was necessary in order to make the exposed services better suited to the needs of its client software components (such as the HCP App and the Conversion and Translation services).
- The **new Section 4.4** contains detailed specifications of the knowledge importing data formats.
- **Section 6** was newly introduced: it was moved here from [D5.10] and has subsequently been improved. It describes a wrapper component around the Platform that provides high-level services that are adapted to the context of the healthcare organisation where the Platform is deployed. Therefore, its presence in the Platform deliverable is better justified than in the deliverable describing data mappings and conversions, where it used to be.

Small changes have also been made in other sections in order to add missing details with respect to the previous version of the document.

2. RELEVANCE TO THE INTEROPEHRATE SCENARIOS

The Platform described in this deliverable is used in all scenarios of the InteropEHRate project. All scenarios require health records to be converted to the FHIR-based Smart Health Data representation. As explained in [D2.3], before each scenario, smart health records are downloaded (either directly or indirectly) from a hospital information system where they undergo a deep data conversion and translation process, consisting of low-level data integration operations carried out by the Platform. For the research scenario (scenario 3) in particular, it is mandatory to perform a deep, "semantic" conversion of health record data values to the representations defined by the Interoperability Profile [D2.8]. Otherwise, the evaluation of research enrolment criteria and research queries would not be feasible due to value-level heterogeneity. The necessary value conversions are made possible by the data cleaning, parsing, and semantic conversion capabilities of the Data Integration Platform that are set up as part of the *Data Scientist Scenario 4* as described in [D2.3].

3. PLATFORM ARCHITECTURE

The role of the Platform is to provide fundamental syntactic and semantic data integration and query functionalities to the *InteropEHRate Health Services* (IHS). Figure 1 below shows the high-level architecture of the Platform, as well as its relation to the IHS and to the users of the IHS: on the one side, *Hospital Information Systems*, and on the other side, applications and systems that consume or store integrated smart health data, such as the *S-EHR App*, the *HCP App*, or the *S-EHR Cloud*.

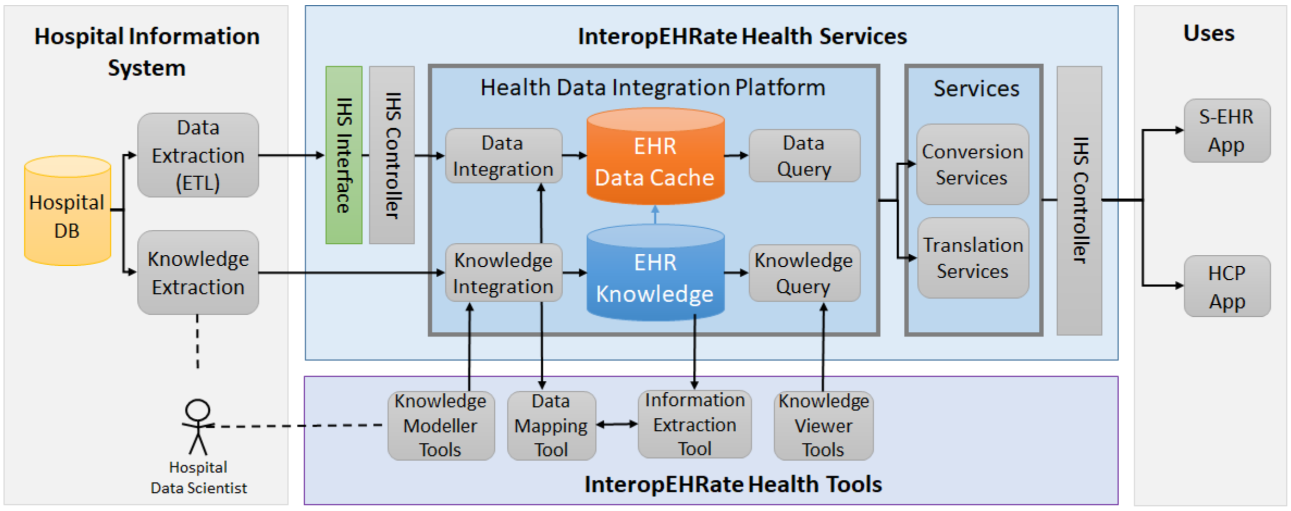

Figure 1 - The Data Integration Platform and its use by outside components

The Platform is composed of two horizontal layers:

1. the **Knowledge Layer** that is responsible of patient-independent healthcare knowledge, such as data structures, encodings, and terminologies;
2. the **Data Layer** that is responsible for patient-specific EHR data and that uses integrated knowledge from the Knowledge Layer.

Vertically, the Platform is divided into three phases:

1. the **Integration phase** that takes informally or semi-formally (i.e., non-semantically) defined knowledge and data from data providers (i.e., hospitals) and converts them to a formal semantic representation;
2. the **Storage phase** that stores the integrated semantic representations;
3. the **Query phase** that serves integrated knowledge and/or EHR data to consumers according to their requirements with respect to language (e.g., French or Italian), terminology (e.g., ICD-9 or ICD-10), and structure (e.g., FHIR).

The process through which the Platform is used - that is filled in with knowledge and EHR data and subsequently queried - is described in detail in the deliverable [D5.10].

4. THE KNOWLEDGE LAYER

The Knowledge Layer of the Platform takes as input *informal* or *semi-formal health knowledge*, such as human-readable or machine-readable descriptions of:

- **EHR data structures** as provided by a given hospital;
- **coded values** as used within EHRs of the hospital;
- **terminology** and **natural language** words as used by the hospital.

The goal of the Knowledge Layer is to formalise such health knowledge into a machine-readable form that the Platform can use to automate the integration of EHRs. While the full health knowledge on which a hospital implicitly relies is vast, only a small subset of it needs to be formalised as required for the representation of EHRs.

4.1. EHR Knowledge Storage

The central storage component of the Knowledge Layer (depicted as the blue DB in the figure above) maintains formal knowledge that describes both local hospital-specific knowledge (terminology and coded values) and their mappings to international knowledge as used by InteropEHRate, such as international codes and FHIR data structures. For each hospital, the Storage component comes pre-loaded with formalised international knowledge:

- FHIR data structures;
- international terminology such as SNOMED CT, LOINC, or ICD-10;
- if available, the lexicalisations of all of the above in the local language, as well as in any other language that the local institution is willing and capable of supporting (e.g., in multilingual regions of Europe the support of more than one language may be required);
- international data instances ("entities") such as richly described pharmaceutical substances as provided by WHO ATC.

4.2. Knowledge Integration

The initial *Knowledge integration* phase consists of formalising local, hospital-specific knowledge, as well as its mapping to international knowledge, into a *formal knowledge representation* that the Platform can use in order to automate the conversion and translation of health records. Knowledge integration is always a manual or semi-automated process that is governed by persons provided by the hospital having the following roles:

- a **data scientist** whose role is to understand the human-readable descriptions of informal health knowledge, and subsequently to use interactive tools to convert these descriptions into formal knowledge. In particular this person has to know the format of the data in the hospital database as well as the medical knowledge to understand the data.
- a **software developer** who, if necessary, assists the data scientist in case the formalisation activity is automatable and would otherwise be too onerous to implement manually, such as in the case of integrating large terminologies with tens of thousands of entries.

It is possible that no full documentation of all aspects of knowledge above is available at a given hospital, but some of the knowledge is implicit within the hospital's information system (e.g., embedded within tools and local data structures). The manual *knowledge extraction* process, executed by the data scientist, is supposed to identify such implicit forms of knowledge and make them explicit, e.g., in the form of a report.

Knowledge integration produces the following results:

- **spreadsheets** (of simple spreadsheet/CSV format) that formally describe the terminology and coded data values used by the hospital, as well as how they map to international terminology as used by InteropEHRate. These spreadsheets may either be produced manually by the data scientist (if their contents are small) or automatically by scripts written by the software developer.
- **direct intervention on the EHR Knowledge Store** by the data scientist using the *Knowledge Modeller Tools*, for the purpose of smaller updates on knowledge.

For both of these knowledge integration modalities above, RESTful *importing APIs* are used, exposed by the EHR Knowledge Storage component.

Endpoint	Description
importConcepts	Imports a spreadsheet that contains concepts and their corresponding words (lexicalisations) in one or more languages.
createTypes	Creation of new data structures (so-called *entity types*) using concepts already imported in the EHR Knowledge Storage component. This functionality is offered by the *Knowledge Modeller Tool* which is able to access directly to the knowledge resources stored in the platform.
importTypes	Imports data structures (so-called *entity types*) in OWL format. In this case the data structures are defined externally (using external tools) specifying the same concepts stored in the EHR Knowledge Storage component, and then imported using the *Knowledge Modeller Tool.*

Table 1 - Principal endpoints of the Platform knowledge importing API

These endpoints are mainly used by *InteropEHRate Health Tools* for knowledge management.

4.3. Knowledge Query

Integrated knowledge can be queried through RESTful APIs for the following purposes:

- browsing and visualisation of formal hospital-specific knowledge, international IEHR knowledge, and their mappings;
- the automation of data mappings, conversions, and translations.

The following query API endpoints are exposed by the Platform (the list is not exhaustive and contains only the most important endpoints). All endpoints are RESTful and return their output in JSON.

Endpoint	Input	Output	Example call URI	Description
GET /vocabularies	knowledgeBase, [languageCode]	vocabularyID	/vocabularies?knowledgeBase=1&languageCode=fra	Returns the numerical identifiers of a single language given as input (as a 3-letter ISO code, same as “locale” below), or of all languages supported

				(e.g., English, Italian, or French).
GET /concepts	knowledgeBase, wordPrefix, locale	list of concept IDs	/concepts?knowledgeBase=1&wordPrefix=tachycardia&locale=eng	Allows querying the numerical IDs of concepts by corresponding word prefix and three-letter language code. Concepts are language-independent units of meaning that formalise healthcare terms such as diseases, procedures, or lab tests and their results.
GET /concepts/{id}	id, locale	list of words	/concepts?id=1234&locale=eng	Allows querying the words in a given language corresponding to a concept with the given numerical identifier.
GET /words	wordPrefix, vocabulary	list of words	/words?wordPrefix=xyz&vocabulary=1	Returns the lexicalisation ("translation") of all the words having the specified prefix, in a given language (provided as a numerical vocabulary ID, as returned by the GET /vocabularies endpoint).
GET /types	typeId, locale	type description	/types?typeId=1234&locale=eng	Returns the descriptions of the type specifying the FHIR data structures (resources) supported and of their attributes, in a given language (locale).
GET /data/ exportTypesRDF	knowledgeBase, locale	all data schemas	/data/exportTypesRDF?knowledgeBase=1&locale=eng	Returns in OWL format all data schemas defined in the knowledge base, with attribute names provided in the language indicated by the locale parameter (as a three-letter language code).

Table 2 - Principal endpoints of the Platform knowledge query API

These endpoints are mainly used by the *Conversion Services* and *Translation Services* of the *InteropEHRate Health Services* for converting EHRs to the FHIR-based representation, to map the coded values contained within, and translate their textual across languages. These services are described in the deliverable [D5.10].

4.4. Knowledge Input Formats

This section defines the file formats expected by the importConcepts and importTypes endpoints of knowledge integration, as described above in section 4.2.

4.4.1. Linguistic and Terminological Knowledge and its mappings (importConcepts)

Beyond concepts, the *importConcepts* endpoint imports knowledge of multiple forms:

- **supra-lingual concepts** expressing the meaning of healthcare terms, classification items, codes, and data attributes (e.g. the meaning of "cerebral infarction");
- the actual **codes** that designate the term or classification item (e.g. "ICD10_I69.X");
- **natural-language terms** the lexicalise the concepts above, including attribute names;
- **descriptions** of concepts in a given language (e.g. "infarto cerebrale", which is the Italian description of the concept above);
- **terminological mappings** between codes and meanings in general (e.g. "ICD10_I69.3" *more_specific_than* "ICD10_I69.X" or "ICD10_I69.8" *equivalent_to* "ICD9:43889".

The definition of all of the information above is carried out by filling in, either manually or automatically, a spreadsheet, exported in XLS format, that is expected by the *importConcepts* endpoint. The spreadsheet, a template of which is provided at the InteropEHRate code repository [XLS Template], contains the following:

- **senses:** describes all kinds of information above except for "more_general_than" and "more_specific_than" mappings;
- **relations:** describes mappings of the kind "more_general_than" and "more_specific_than";
- **concepts_move:** not used in the context of InteropEHRate;
- **gaps:** not used in the context of InteropEHRate;
- **etypes:** not used in the context of InteropEHRate;
- **domains:** not used in the context of InteropEHRate.

The senses tab. Each row in the senses tab corresponds to a *term*. Each term consists of a concept, its lexicalisation in a given language, as well as metadata describing the term. Each term is defined through the following spreadsheet columns.

Column	Meaning	Example values
Cased Word Lemma	The (cased) word, multiword expression, or code that lexicalises a concept in a given language. For codes and attribute names, it is good practice to apply a prefix that indicates the underlying standard, e.g. "icd10_" or "fhir_".	cerebral infarction, ICD10_I69.3, fhir_DiagnosticReport.status

Word Forms	Not used in the context of InteropEHRate, to be left empty.	-
Concept UK ID	A unique numerical ID that will internally identify the concept. Three kinds of numerical IDs are allowed: ● positive integer: refers to an existing concept, in which case the ADD operation (see below) cannot be used; ● negative integer: the system will allocate a new concept ID dynamically. If two rows contain the same negative concept ID, it means that they refer to the same concept (i.e. dynamic allocation will happen only at the first occurrence); ● resource part reference: an already existing concept is referenced through its resource part reference as opposed to its internal concept ID (see below).	123, -1, SCTID_432504007
Word Sense Rank	Serves the purpose of ranking word meanings by importance. It is recommended to use the inbuilt dynamic allocation mechanism by providing decremented negative integer values for every row (e.g. the first row containing "-1", the second "-2", the third "-3" and so on).	-1 -2 -3 -4
Concept Word Rank	Serves the purpose of ranking synonymous words (lemmas linked to the same concept) by importance. It is recommended to use the inbuilt dynamic allocation mechanism by providing decremented negative integer values for every row (e.g. the first row containing "-1", the second "-2", the third "-3" and so on).	-1 -2 -3 -4
Description	Provides a textual description for a concept in a given language.	"The status of the diagnostic report text information"
Operation	One of "ADD", "DELETE", or "UPDATE". For the creation of new knowledge, "ADD" should be used.	ADD
Language	Three-letter ISO 693-3 language codes that specify the language of the term or description.	eng ita
Url Reference	Provenance in the form of a URL. If not empty, then User Reference, Resource Reference, and Resource Part Reference must be empty.	http://somedomain.com/path/
User Reference	Provenance in the form of a person name and email address responsible for providing the information, using the syntax: name<SPACE>(email@address) If not empty, then URL Reference, Resource Reference, and Resource Part Reference must be empty.	Mario Rossi (mario.rossi@example.com)

Resource Reference	Provenance in the form of a resource name or URL, in case the information was retrieved from an existing resource. If not empty, then URL Reference and User Reference must be empty. This field should be used to specify which version of which existing resource served as the source of the information.	FHIR v4.0.1 Specifications
Resource Part Reference	Provenance in the form of an external identifier used in an existing resource. If not empty, then URL Reference and User Reference must be empty. This field should be used to specify the precise entry in an external source that served as the origin of the information, such as a SNOMED CT term identifier when importing SNOMED CT terms. If at a later point the SNOMED term needs to be modified (e.g. it becomes obsolete), it can be done using the external ID as opposed to indicating the (dynamically allocated) internal concept ID.	SCTID_432504007
Note	Optional textual note accompanying the entry.	"Created for testing purposes."

Table 3 - Structure of the terminological knowledge to be imported

An example of the contents of the senses spreadsheet tab is shown below:

Cased Word Lemma	Concept UK ID	Word sense rank	Concept Word Rank	Description	Operation	Language	Resource Reference	Resource Part Reference
ICD10_I63	-1	-1	-1	Cerebral infarction, incl.: occlusion and stenosis, excl.: sequelae	ADD	eng	ICD10	
ICD10_I63.X	-1	-2	-2		ADD	eng	ICD10	
cerebral infarction	-1	-3	-3		ADD	eng	ICD10	
infarto cerebrale	-1	-4	-4	Infarto cerebrale	ADD	ita	ICD9 Italian Translation	

SCT_432504007	-1	-5	-5		ADD	eng	SNOMED CT International 2021-01-31	SCTID_432504007
fhir_DiagnosticReport.status	-2	-6	-6	The status of the diagnostic report text information	ADD	eng	FHIR v4.0.0	

In the example above, two concepts are created, one corresponding to the term "cerebral infarction" and the other to "diagnostic report status". Five distinct lemmas (lexicalisations) are defined for the first (the ICD10 code expressed in two standard manners, an English lexicalisation, an Italian lexicalisation, and the equivalent SNOMED CT code) and one lemma expressing the embodiment of the attribute within the FHIR standard (*fhir_DiagnosticReport.status*).

The relations tab. The purpose of the relations tab is to define broader - narrower relationships between concepts. This allows us to reproduce, for example, the SNOMED CT or the ICD hierarchies, as well as to define broader - narrower relationships across hierarchies. The structure of the relations tab is as follows.

Column	Meaning	Example values
Parent Concept UK ID	Identifier of the source concept of the relation (see senses tab for details).	123, -1, SCTID_432504007
Parent Concept Label	Optional label that provides a human-readable meaning for the source concept (ignored during importing).	Cerebral infarction
Child Concept UK ID	Identifier of the source concept of the relation (see senses tab for details).	124, -2, SCTID_230693009
Child Concept Label	Optional label that provides a human-readable meaning for the source concept (ignored during importing).	Anterior cerebral circulation infarction
Relation Kind	Identifies the meaning of the relation. Available relation types: IS_A PART_OF SUBSTANCE_OF	IS_A

	MEMBER_OF	
Operation	Same as in the senses tab.	ADD
Language	Same as in the senses tab.	eng ita
Url Reference	Same as in the senses tab.	http://somedomain.com/path/
User Reference	Same as in the senses tab.	Mario Rossi (mario.rossi@example.com)
Resource Reference	Same as in the senses tab.	FHIR v4.0.1 Specifications
Resource Part Reference	Same as in the senses tab.	SCTREL_1234567890
Note	Same as in the senses tab.	"Created for testing purposes."

Table 4 - Structure of the terminological knowledge relations to be imported

The following is example content for the relations tab:

Parent Concept UK ID	Parent Concept Label	Child Concept UK ID	Child Concept Label	Relation Kind	Operation	Language	Resource Reference	Resource Part Reference
-1	Cerebral infarction	-3	Anterior cerebral circulation infarction	IS_A	ADD	eng	SNOMED CT International 2021-01-31	SCTREL_1234567890

4.4.2. Data Schemas (importTypes)

While the Knowledge Management Tools contain an Entity Type Modeller Tool that allows the interactive definition of data schemas (target FHIR data schemas to be used for data mapping), the Platform also allows the batch importing of schema definitions in OWL format, as well as their exporting from the Platform, using the *importTypes* and the *data/exportTypesRDF* endpoints, respectively. The detailed specifications of the precise OWL format used by the Platform are beyond the scope of this deliverable. However, the FHIR data models used in the project, as defined by the InteropEHRate Interoperability Profiles, are downloadable in OWL format from the InteropEHRate code repository [FHIR_OWL].

5. THE EHR DATA LAYER

The purpose of the Data Layer is:

1. to **integrate** local EHRs, i.e., convert them from their local representations into a formal, supra-lingual and international representation;
2. to **store** these representations temporarily in an *EHR Data Cache* (the time of storage being customisable by the hospital);
3. to **query and retrieve** these representations in standard serialised FHIR format and in a given language, ready to be used by FHIR-compliant applications such as the *HCP App*, the *S-EHR App*, and the *S-EHR Cloud*.

5.1. EHR Integration

The integration is a semi-automated process that is executed in two phases:

- **semi-automated setup phase:** in this manually executed phase a data scientist from the hospital defines the rules, based on a small- or medium-sized corpus of local EHRs, by which the local EHR data structures and data values will be mapped to FHIR: this set of rules or "recipe" is called the *data integration model*;
- **automated data integration phase:** in this fully automated phase the Platform converts EHRs automatically from their local representation into their FHIR-based international representation, using the "recipe" defined by the data scientist in the setup phase.

In the setup phase, the data scientist uses the interactive *Data Mapper Tool* (DMT) from the *InteropEHRate Health Tools*. The output of the tool is the data integration model (recipe) that helps automate the integration process.

In the production phase, an automated data integration service uses the recipe from above to automate the conversion of local EHRs. The output of the production phase, namely the converted health record, is directly written into the *EHR Data Cache*.

In both phases, the input consists of one or more EHRs, each EHR physically represented as a set of files. Each file corresponds to a "data table" or data structure as provided by the hospital. The files must be expressed in a table-structured CSV/TSV (comma-separated/tab-separated) format or in a tree-structured XML or JSON format.

5.2. EHR Data Cache

The *EHR Data Cache* is logically a graph database that stores EHRs as knowledge graphs. The nodes of the knowledge graph are qualified using the *EHR Knowledge* defined in the Knowledge Layer. The graph uses a language-independent representation of the EHR that can subsequently be queried and retrieved in any language provided by the EHR Knowledge.

5.3. EHR Data Query

The EHR Data Cache provides a search and query API that allows the retrieval of both entire health records and portions of health record data. The principal output formats provided are:

- for entire health records or FHIR resources: a serialised FHIR format (XML or JSON);

- the Platform-internal and JSON-based *EML (Entity Markup Language)* format that preserves the original graph structure and can thus be used for the transfer of rich graph data across systems;
- JSON snippets for individual data values and fine-grained data.

The table below provides a summary of the principal API endpoints that we foresee to be used within the project. All endpoints are RESTful and return their output in JSON.

Endpoint	Input	Output	Example URI call	Description
GET /instances	instanceID, lang	entity described through its attributes	/instances?instanceID=1234&lang=eng	These low-level calls return data instances, called also *Entities* (corresponding to the single FHIR resources of a health record bundle) or individual attribute values, expressed in the language given as input. The output format is either the Platform's internal EML graph data format or RDF. This service can be used to help the transfer of resources between different instances (i.e. different Hospitals) of the data integration platform, without the need to transform the data in the formats used internally by the platform (EML, RDF).
Simple Search: **GET /search**	entityBase, query, queryType=SIMPLE	list of entities	/search?entityBase=1&queryType=SIMPLE&query=hemorrhage	Returns data values corresponding to a conventional text-based search. This functionality can be used to verify the presence of a given string within a health record.
Semantic Search: **GET /search**	entityBase, query, queryType=SIMPLE, parseSemantics=true	list of entities	/search?entityBase=1&queryType=SIMPLE&parseSemantics=true&query=hemorrhage	Returns data values corresponding to a "semantic" search where the search is formulated not through words but through concepts. It returns data values that are equivalent or more specific than the query concept. For example, a search for the concept of "hemorrhage" will return EHRs with all kinds of cerebrovascular diseases mentioned within, such as "cerebral infarction". This functionality can be used by the Research Scenario.
Semantic Query:	entityBase, query, queryType=	list of entities	/search?entityBase=1&queryType=EQL&parseSemantics=true&q	Returns data values corresponding to a "semantic" query. The query language is EQL, Entity Query Language, a semantic and

GET /search	EQL, parseSemantics=true		uery='SELECT	entity-centric extension of SQL. Prior semantic analysis of natural-language data values allows semantic queries to retrieve EHRs based on the meaning of the text contained within. "Semantic" queries allow for semantic reasoning over data values, e.g., "SELECT * WHERE disease is_a *'cerebrovascular_disease'* " would return all EHRs with a specific type of cerebrovascular disease, e.g., cerebral infarction. This functionality can be used by the Research Scenario.

Table 5 - Principal endpoints of the Platform EHR data query API

6. THE IHS CONTROLLER AND INTERFACE

The IHS Controller (see Figure 1 as well as Figure 2 below) acts as a middleware between calls from external applications (such as the HCP App) and health data conversion and translation functionalities. Its role is to adapt the latter operations to the specific requirements of the local hospital context: the local language, the local standards used, the local IT system, etc. Therefore, through the *IHS Interface* (IHSI) it offers to hospitals, it receives high-level calls that are agnostic of local specificities.

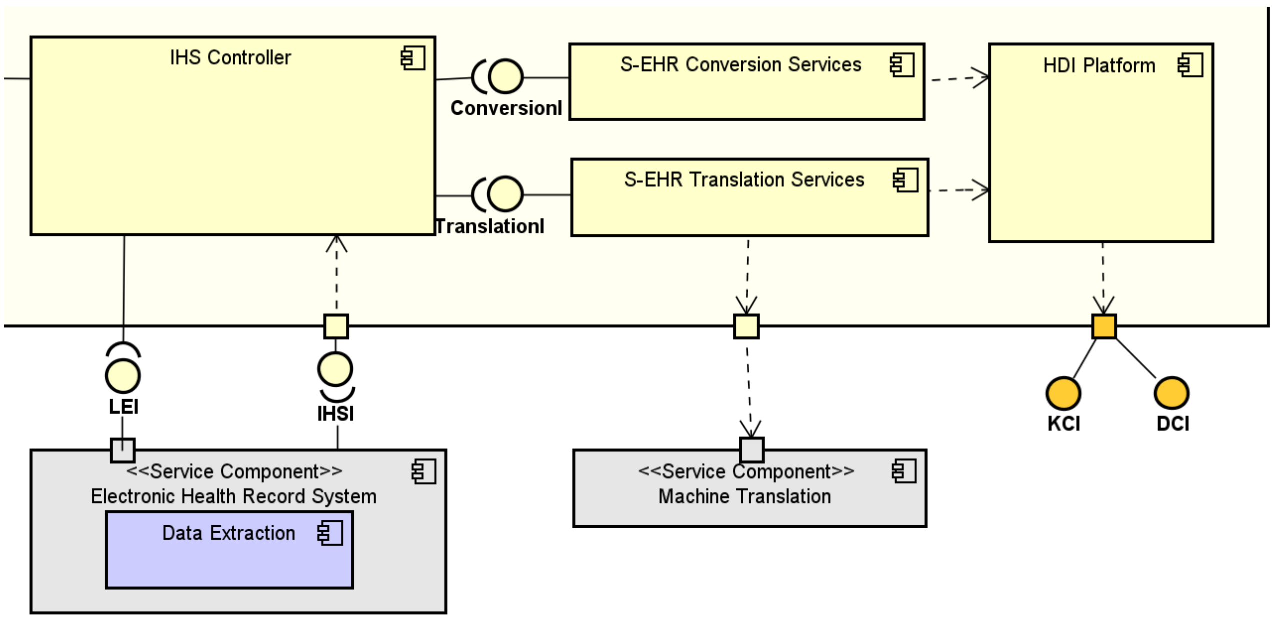

Figure 2 - Relevant detail from the IHS component diagram showing the interfaces and components discussed in this section

6.1. IHS Interface Datatypes

The following table provides descriptions of the non-standard datatypes appearing in the IHS Interface.

Datatype	Description
FHIRResource	Represents a FHIR resource or entire bundle, in one of its standard serialised forms, as supported by the interface.
ModelString	In the *codeConversion* call described in section 6.2 below, the *model* parameter appears. This parameter is a simple String, composed of semicolon-separated pairs "X-Y", where X and Y are data standards used inside health data. The pair indicates that the codes of the data standard X have to be converted into the respective codes of the data standard Y. For example, "icd9-icd10;aic-atc" means that ICD-9 codes should be converted into ICD-10, and Italian AIC codes should be converted into their international ATC counterparts.
CompressedFile	A ZIP-compressed bundle of a set of files representing an entire hospital-specific patient health record, or a part of it. The files may be of different

	types (e.g. unstructured PDFs, structured XML files, spreadsheets): using its internal configuration, the IHS Controller uses the filenames and extensions to choose the appropriate conversion models to be applied. In other words, the filenames and extensions used inside the bundle must remain consistent across health records.

Table 6 - IHS Interface Specific Datatypes

6.2. IHS Interface Endpoints

The IHSI exposes a set of endpoints that includes the methods called by the HCP App, in order to translate and/or convert the content of health records, such as *requestConversion*, *translateFHIRResource*, as well as other methods provided by internal Interfaces (such as TranslationI and ConversionI) that can be called individually. The table below describes the RESTful IHSI endpoints.

IHSI Endpoint	Input	Output	Example call	Description
POST /requestConversion	Compressed File comp, requestID	void	/requestConversion	Sends for conversion a single ZIP file containing one or more files that constitute the health record of a patient. The ZIP file need not be complete: multiple calls can be made with different parts of the health record, which the IHS system will automatically link together. The call is asynchronous: it requires a requestID input parameter in order to later associate the converted health record to the right conversion request.
POST /translateFHIRResource	FHIRResource resource, [String targetLang]	FHIRResource translatedResource	/translateFHIRResource	Translates natural language labels inside a FHIR resource (or bundle) in input into the target language. The parameter targetLang may be omitted, in which case translation will happen towards the local language, as defined by the local IHS configuration.
POST /conceptTranslation	FHIRResource resource, String targetLang	FHIRResource translatedResource	/conceptTranslation?targetLang=it	This call allows the translation at concept level (the translation of code descriptions and terms identified as concepts on the base of the domain knowledge collected). It takes as input a FHIR resource and provides as output the same resource with the concepts translated in the language indicated in input (targetLang). This call is used within the more general

				endpoint extendWithTransltion.
POST /machineTranslation	FHIRResource resource, String targetLang	FHIRResource translatedResource	/machineTranslation?targetLang=it	This call allows the translation of specific text attributes only, within a FHIR resource. It takes as input a FHIR resource and provides as output the same resource with the attributes translated in the language indicated in input (targetLang). This service is also used by the more general endpoint extendWithTransltion.
POST /extendWithTranslation	FHIRResource resource, String targetLang	FHIRResource translatedResource	/extendWithTranslation?targetLang=it	This call allows the translation of an entire FHIR resource (i.e. Bundle), calling internally both the machineTranslation and conceptTranslation methods. It takes as input a FHIR resource and provides as output the same resource with the attributes translated in the language indicated in input (targetLang). This service is also used by the more general endpoint translateFHIRResource.
POST /codeConversion	FHIRResource resource, ModelString model	FHIRResource convertedResource	/codeConversion?model=icd9-icd10	Converts all the coded values within a FHIR resource from one standard to another. It takes as input a FHIR resource and the model string specifying the conversion, and provides as output the same resource with the codes converted in the specified standards. It is assumed that the underlying Data Integration Platform has been provided the formal knowledge necessary to carry out the conversion requested here.
POST /convertFHIRResource	FHIRResource resource fhirResource	FHIRResource resource convertedFhirResource	/convertFHIRResource	This call allows to convert a FHIR resource (i.e. HCP update on Citizen's SHER) produced by the HCP (using HCP App), in order to be later uploaded on the S-EHR app, and readable by the Citizen.

Table 7 - IHSI endpoints

6.3. Hospital Information System Necessary Endpoints

This interface, implemented by local institutions, provides access to the local (legacy) EHR information system in a controlled way. It provides the following endpoints:

LEI Endpoint	Input	Output	Example call URI	Description
POST /onDataConverted	convertedHealthData, requestID	-	/onDataConverted	Sends the converted health record as an entire FHIR bundle to the hospital information system. Alongside the bundle, the request ID corresponding to the original conversion request is also returned.

Table 8 - LEI endpoints

6.4. IHS Controller Configuration

At deployment time, the IHS Controller has to be configured in order to work correctly in the local hospital context. This initial setting consists in defining local configuration parameters needed for the conversion and translation operations (apart from setting up the so-called *local knowledge* for knowledge-based data integration, as described in [D5.10]). The configuration parameters are stored in a JSON file representing the IHS configuration file, and they are described as follows:

- **applicationPort:** the number of the server's port used to run the IHS component.
- **DIPlatformHost:** the host name of the server where the Data Integration Platform is deployed.
- **DIPlatformPort:** the number of the server's port used to run the Data Integration Platform API.
- **vocabularies:** a list of JSON elements describing the languages supported by the IHS. This parameter includes for each language: the name of the language, the ISO 639-1 code and the ISO 639-2 code.
- **mappingModelPath:** the path of the location where the mapping tool's models are stored, in order to be retrieved to perform the correct mapping within the conversion of each data resource managed.
- **dataSchemaPath:** the path of the location where the interoperable data schema is stored. This is the schema generated using the knowledge resources previously collected.
- **FHIRExportParameter:** this is a list of FHIR resource names, specifying the resources to be considered when a health record has to be exported in FHIR from the EHR data cache (allowing a selective export of the health record contents).
- **process:** an ordered list of data extraction and conversion operations that specifies in what order and how each file of the input health record should be converted (see below for details).
- **Information Extraction Configuration:** the configuration parameters for Information Extraction will be provided in the corresponding deliverable [D5.12].

The **process** configuration attribute above is an ordered list of operations to be executed automatically over the health record arriving in input. The health record typically consists of multiple files of different types (PDF documents, XMLs, spreadsheets), some of which may need to be preprocessed (e.g. extraction of text from a PDF or splitting of a complex XML document into multiple parts). After preprocessing, the files need to be converted into a knowledge graph (an operation referred to as “data mapping”), from which the Data Integration Platform produces the FHIR-based output. Both the extraction and the mapping operations are described in the recipe as a list of operations:

FILENAME PROCESSOR MODEL

where “FILENAME” is the name of the input file from the original health record (possibly using wildcards if necessary), “PROCESSOR” refers to a specific executable in charge of preprocessing or mapping, and “MODEL” is the preprocessing or mapping model.

For example, in the case of a simple input bundle of two health record files:

- `discharge.pdf`: a discharge report in PDF format;
- `ips.xml`: an International Patient Summary file in XML format;

the process parameter may define a PDF extraction preprocessing operation, an XML splitting preprocessing operation, and a data mapping operation as follows:

```
process: [
      { “discharge.pdf”,          “PDFEXTRACTOR”,  “” },
      { “discharge.csv”,          “DATAMAPPER”,    “DISCHARGE.TTL” },
      { “ips.xml”,                “XMLSPLITTER”,   “IPS_SPLITTER” },
      { “ips_patient.xml”,        “DATAMAPPER”,    “IPS_PATIENT.TTL” },
      { “ips_visit.xml”,          “DATAMAPPER”,    “IPS_VISIT.TTL” },
      { “ips_medication.xml”,     “DATAMAPPER”,    “IPS_MEDICATION.TTL” }
]
```

Here, the `PDFEXTRACTOR` generates a `discharge.csv` file as output (no model needed), which is then mapped into FHIR using the `DATAMAPPER` tool and `DISCHARGE.TTL` as the mapping model. The `ips.xml` file, in turn, is split into three parts by the `XMLSPLITTER`, namely `ips_patient.xml`, `ips_visit.xml`, and `ips_medication.xml`. Each of them is then mapped using its corresponding mapping model.

7. USE OF THE PLATFORM FOR HEALTH RECORD CONVERSION AND TRANSLATION

Deliverables [D5.10], [D5.11], and [D5.12] provide a detailed description of all steps that a hospital needs to implement in order to deploy and configure the conversion and translation process. In the following, we provide a more technical and developer-oriented description that concentrates on the specific configuration steps and service calls that the hospital needs to make to the IHS Controller in order to obtain a FHIR-based converted and/or translated version of a health record.

In order to automate the conversion and translation of health records, both the IHS Controller and the underlying Data Integration Platform need to be properly configured and pre-loaded with the necessary knowledge. The configuration of the IHS Controller is described in Section 6.3 above, while the knowledge setup process for the Data Integration Platform is described in [D5.10].

Once the configuration and setup are done, the hospital is ready to receive requests to serve FHIR-based health records and/or to perform translations into the local language or into the language of the patient. In the scenarios supported by the InteropEHRate project, such requests originate either from the S-EHR Apps of patients who wish to upload their health records onto their mobile devices, or from the HCP Apps used in hospitals that need to perform translation or information extraction.

7.1. Conversion and translation upon request from the S-EHR App

In this subsection we consider the use case where a patient wishes to download from a hospital his/her health record in FHIR format, and in his/her own language, onto his/her S-EHR App. This involves a single-step conversion+translation of the health record from the format used within the hospital into FHIR and into the patient's language. The main software components participating in the process are:

- the **S-EHR App** that initiates the request;
- the **Hospital R2D Server** to which S-EHR Apps connect and that forwards the requests to the hospital's information system;
- the **Hospital EHR System** that retrieves the corresponding health record and calls the InteropEHRate Health Services in order to convert (and possibly also translate) them;
- the **InteropEHRate Health Services** in charge of conversion and translation.

Upon request of a health record by the S-EHR App from the hospital's R2D Server, the R2D Server verifies if an up-to-date converted version of the health record already exists in its cache, in which case it serves it directly to the S-EHR App. If a cached version does not exist, the R2D Server requests the health record from the hospital's EHR information system. The process, depicted in Figure 3, is as follows.

1. The R2D Server requests the health record from the hospital's EHR information system (`requestHealthData`).
2. If the converted data is not already available, the hospital retrieves the health data and requests their conversion to the IHS, by sending the data to the `POST /requestConversion` endpoint.
3. The hospital can make multiple conversion requests to the IHS on different parts (files) of the same health record. Each single conversion is performed by calling the internal call `POST /convertHealthData`, provided by the Conversion Service [D5.10]. This may be necessary in

case a single health record is too large to process it in a single call. The IHS is able to “merge” the parts automatically and silently, provided that the patient is clearly identified (e.g. through a patient ID) within each file provided.

4. Upon completion of all conversion calls, related to the same health record, the IHS retrieves the full health record as a single FHIR bundle through the `GET /retrieveConvertedFHIR` internal call, provided by the Conversion Service [D5.10]. In case the S-EHR App also requested translation into the patient’s own language, the language parameter can be provided to the call.
5. The IHS forwards the FHIR bundle, obtained from the Conversion Service, to the hospital’s EHR information system, calling the `POST /onDataConverted` call provided by the EHR itself. As an alternative, the `onDataConverted` endpoint may also be implemented by the R2D Server, in which case the IHS has the option of directly sending the conversion results to the R2D Server, skipping the EHR Middleware. This alternative solution is not shown on the diagram below.

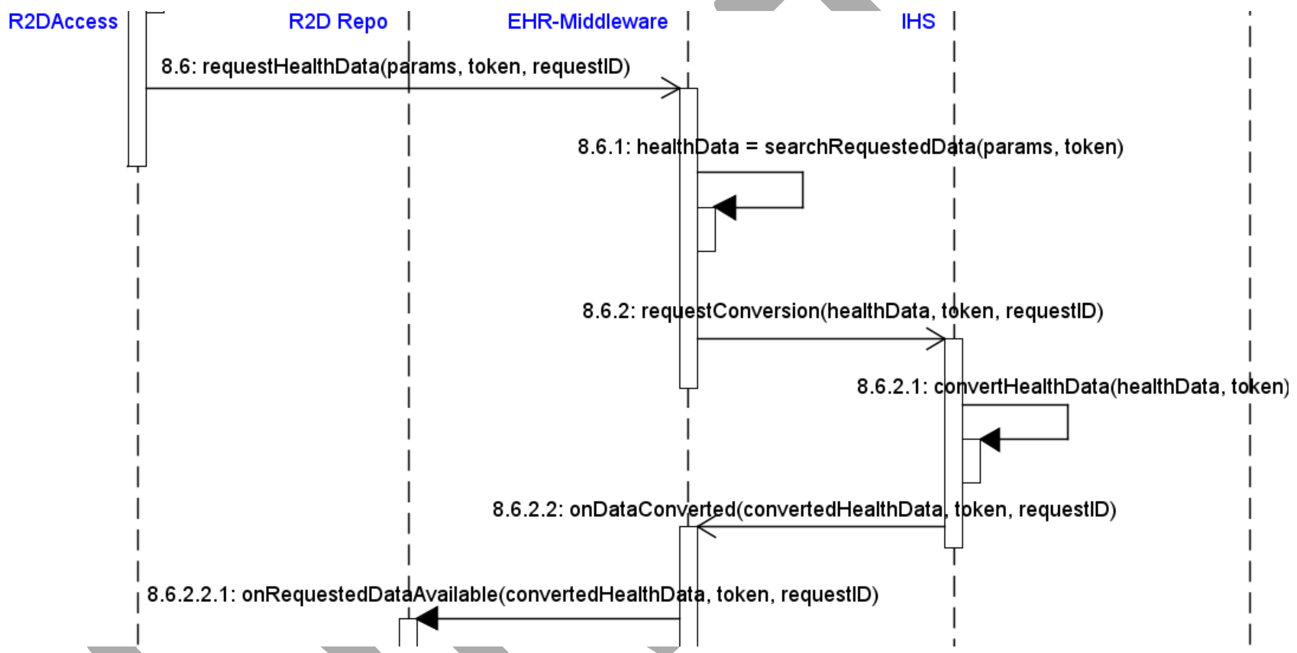
R2DAccess
R2D Repo
EHR-Middleware
IHS
8.6: requestHealthData(params, token, requestID)
8.6.1: healthData = searchRequestedData(params, token)
8.6.2: requestConversion(healthData, token, requestID)
8.6.2.1: convertHealthData(healthData, token)
8.6.2.2: onDataConverted(convertedHealthData, token, requestID)
8.6.2.2.1: onRequestedDataAvailable(convertedHealthData, token, requestID)

Figure 3 - Retrieval of a health record from a hospital by a S-EHR App

7.2. Conversion and translation upon request from the HCP App

In this section we consider a use case where an HCP App within a hospital:

- needs to display a FHIR-based health record (retrieved either from a patient’s S-EHR App or from the S-EHR Cloud) in the local language of the hospital;
- needs to convert new data entered by the HCP into the HCP App into FHIR format or translate the same data into the patient’s own language.

Even though the HCP App is natively capable of producing FHIR-based data, conversion may still be necessary in order to transform free text entered by the HCP into formal concepts, terms, and codes. The process involves the following systems:

- the **HCP App** that initiates the request;
- the **InteropEHRate Health Services** in charge of conversion and translation.

The initial translation process, depicted on the top of Figure 4, consists of the following steps:

1. In order to translate the FHIR-based health record contents into the local language, the HCP App calls the `translateFHIRResource` endpoint of the IHS. A single call over the entire FHIR bundle is sufficient. The HCP App does not need to indicate the target language, as it is stored as a local configuration parameter inside the IHS.
2. The IHS applies concept-based translation and free-text-based machine translation (possibly using a third-part machine translation service) to the health record contents.
3. The IHS returns the result to the HCP App.

After the editing of the health record by the HCP, the final conversion and translation operations invoked by the HCP App are as follows:

1. the HCP App calls the `convertFHIRResource` method of the IHS over each FHIR resource that was generated or modified by the HCP.
2. The IHS performs the conversion in an asynchronous manner, applying concept extraction to the resources.
3. The HCP App makes a second call to the IHS, this time (optionally) asking for translation into the patient's own language. The `extendWithTranslations` method is called, as the goal is to obtain FHIR resources that contain data both in the original language (in which the HCP entered the data) and in the patient's language. The patient's language needs to be indicated explicitly to the method as an input argument.
4. After performing both concept and machine translation, the IHS returns the result to the HCP App.

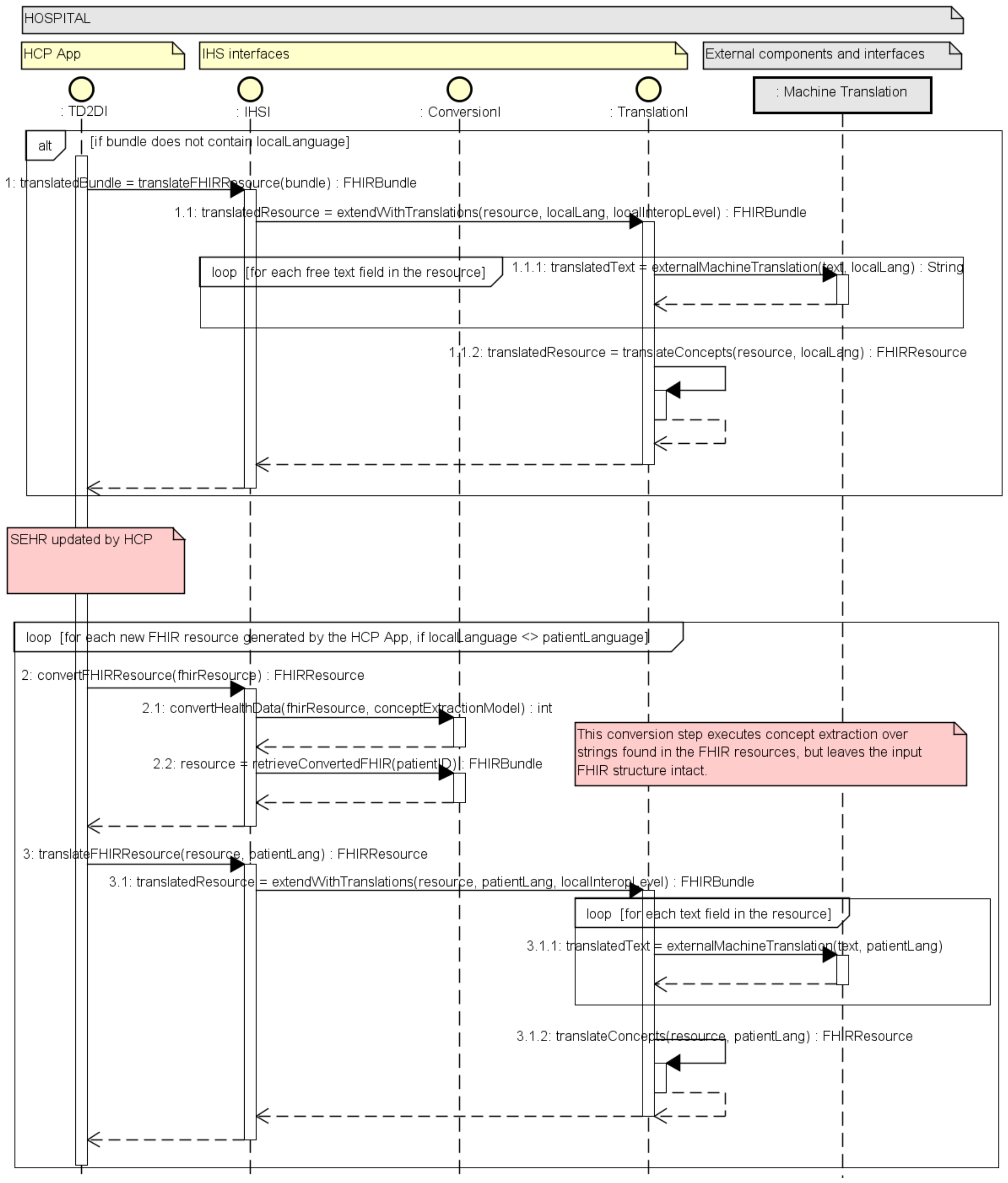
HOSPITAL
HCP App
IHS interfaces
External components and interfaces
: Machine Translation
: TD2DI
: IHSI
: ConversionI
: TranslationI
alt
[if bundle does not contain localLanguage]
1: translatedBundle = translateFHIRResource(bundle) : FHIRBundle
1.1: translatedResource = extendWithTranslations(resource, localLang, localInteropLevel) : FHIRBundle
loop [for each free text field in the resource]
1.1.1: translatedText = externalMachineTranslation(text, localLang) : String
1.1.2: translatedResource = translateConcepts(resource, localLang) : FHIRResource
SEHR updated by HCP
loop [for each new FHIR resource generated by the HCP App, if localLanguage <> patientLanguage]
2: convertFHIRResource(fhirResource) : FHIRResource
2.1: convertHealthData(fhirResource, conceptExtractionModel) : int
2.2: resource = retrieveConvertedFHIR(patientID) : FHIRBundle
This conversion step executes concept extraction over strings found in the FHIR resources, but leaves the input FHIR structure intact.
3: translateFHIRResource(resource, patientLang) : FHIRResource
3.1: translatedResource = extendWithTranslations(resource, patientLang, localInteropLevel) : FHIRBundle
loop [for each text field in the resource]
3.1.1: translatedText = externalMachineTranslation(text, patientLang)
3.1.2: translateConcepts(resource, patientLang) : FHIRResource

Figure 4 - Conversion and translation of a FHIR-based health record initiated by the HCP App

8. CONCLUSIONS AND NEXT STEPS

This document provided specifications for the IEHR Data Integration Platform and for its deployment and configuration within a hospital information system environment. The Platform is used as a domain-agnostic technological basis to implement high-level data mapping, conversion, translation, and information extraction services. These services are to a large extent controlled by domain knowledge - formal definitions of terminologies, coding standards, data structures, and their mappings - that needs to be uploaded into the Platform for it to be adapted to local needs. This adaptation process is described in detail in the deliverable [D5.10] for data mapping and conversion, as well as in [D5.11] (and, later, [D5.12]) for data translation and information extraction.

REFERENCES

- **[D2.3]** InteropEHRate Consortium, InteropEHRate Requirements v3 (deliverable D2.3), 2021. https://www.interopehrate.eu/resources/#dels
- **[D2.6]** InteropEHRate Consortium, InteropEHRate Architecture v3 (deliverable D2.6), 2021. https://www.interopehrate.eu/resources/#dels
- **[D2.8]** InteropEHRate Consortium, Interoperability Profile v2 (deliverable D2.8), 2020https://www.interopehrate.eu/resources/#dels
- **[D5.5**] InteropEHRate Consortium, Design of an integrated EHR web app for HCP - V2 (deliverable D5.5), 2020. https://www.interopehrate.eu/resources/#dels
- **[D5.10]** InteropEHRate Consortium, Design of the Data Mapper and Converter to FHIR - v2 (deliverable D5.10), 2021. https://www.interopehrate.eu/resources/#dels
- **[D5.11]** InteropEHRate Consortium, Design of information extractor and natural language translator v1 (deliverable D5.11), 2020. https://www.interopehrate.eu/resources/#dels
- **[D5.12]** InteropEHRate Consortium, Design of information extractor and natural language translator v2 (deliverable D5.12). https://www.interopehrate.eu/resources/#dels
- **[XLS_Template]** Example spreadsheet template for terminology import. http://iehrgitlab.ds.unipi.gr/interopehrate/health-services/fhir-import/blob/master/Knowledge%20import%20resources/KnowledgeImportFileTamplate.xls
- **[FHIR_OWL]** Platform-specific OWL representation of the FHIR data models used in the project. http://iehrgitlab.ds.unipi.gr/interopehrate/health-services/fhir-import/blob/master/Knowledge%20import%20resources/FHIRSchema_21-06-2021.owl