D4.2

Specification of remote and D2D protocol and APIs for HR exchange - V2

ABSTRACT

This report describes the second version of the open specification of the device-to-device (D2D) and the remote-to-device (R2D) protocols, defined (and to be experimented) in the context of the InteropEHRate project. These protocols will be used by software applications facilitating health data exchange between the citizens and the Healthcare professionals. The D2D protocol defines the technology, data structures and operations to be offered by the interacting applications, allowing the exchange of health data between the aforementioned stakeholders without the usage of internet. On the other side, the R2D protocol defines the technology, operations and structure of data used for enabling the exchange of health data between an Electronic Health Record (EHR) system of a single organization, either a National EHR or to a S-EHR Cloud (a remote cloud system for the storage of encrypted personal health data) and the citizen with the usage of internet.

Delivery Date	April 30th, 2020
Work Package	WP4
Task	T4.1
Dissemination Level	Public
Type of Deliverable	Report
Lead partner	UPRC

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	Thanos Kiourtis, Argyro Mavrogiorgou, Dimitris Laliotis, Aikaterini Aggeli	UPRC
	Alessio Graziani	ENG
	Martin Marot, Julien Henrard	A7
	Nicu Jalba	SIVECO
Reviewers	Simone Bocca	UNITN
	Chrysostomos Symvoulidis	BYTE

LOG TABLE

Version	Date	Change	Author	Partner
0.1	2020-02-18	First draft of ToC	Thanos Kiourtis, Argyro Mavrogiorgou	UPRC
0.2	2020-02-18	Updated sections related with the Introduction, D2D protocol specification, and Conclusions/Next Steps	Thanos Kiourtis, Argyro Mavrogiorgou, Konstantinos Vidakis	UPRC
0.3	2020-02-24	Updates sections regarding the Bluetooth Profiles to be used for Android and iOS devices, regarding the D2D protocol.	Thanos Kiourtis, Argyro Mavrogiorgou	UPRC
0.4	2020-03-20	Updated section of R2D with new specifications, revised common section about conceptual data model	Alessio Graziani, Francesco Torelli	ENG
0.5	2020-04-10	Added section about R2Cloud, moved section about eHDSI in annex, revised technical section about FHIR	Alessio Graziani, Francesco Torelli	ENG
0.6	2020-04-22	Finalized missing parts regarding the updates with the	Thanos Kiourtis, Argyro	UPRC

		previous version of the deliverable, updated references	Mavrogiorgou	
0.7	2020-04-22	Sent for internal review	Thanos Kiourtis, Argyro Mavrogiorgou	UPRC
0.8	2020-04-27	Internal reviewers' comments received and changes were made accordingly	Thanos Kiourtis	UPRC
0.9	2020-04-28	Sent for QA	Thanos Kiourtis	UPRC
1.0	2020-04-28	Quality review	Argyro Mavrogiorgou	UPRC
VFinal	2020-04-30	Final check and submission	Laura Pucci	ENG

ACRONYMS

Acronym	Terms and definition
D2D	Device-to-Device
R2D	Remote-to-Device
EHR	Electronic Health Record
S-EHR	Smart Electronic Health Record
NFC	Near Field Communications
RFID	Radio Frequency Identification
WBAN	Wireless Body Area Networks
WPAN	Wireless Personal Area Networks
WLAN	Wireless Local Area Networks
VAN	Vehicle Area Networks
WSN	Wireless Sensor Networks
EMR	Electronic Medical Record
BLE	Bluetooth Low Energy
IEEE	Institute of Electrical and Electronics Engineers
EDR	Enhanced Data Rate
GATT	Generic Attribute Profile
LTE	Long-Term Evolution
AoA	Angle of Arrival
AoD	Angle of Departure
P2P	Peer-to-Peer
ISM	Industrial, Scientific, and Medical
GUI	Graphical User Interface
UI	User Interface
UX	User Experience

ITU	International Telecommunication Union
MD2D	Mobile Device-to-Device
TD2D	Terminal Device-to-Device
MD2DI	Mobile Device-to-Device Interface
TD2DI	Terminal Device-to-Device Interface
HCO	Healthcare Organization
CEF	Connecting Europe Facility
eHDSI	eHealth Digital Service Infrastructure
NCP	National Contact Points
API	Application Programming Interface
FHIR	Fast Healthcare Interoperability Resources
UML	Unified Modeling Language
SQL	Structured Query Language
R2DI	Remote-to-Device Interface
UUID	Universally Unique Identifier
SDDB	Service Discovery DataBase
SPP	Serial Port Profile
RFCOMM	Radio Frequency Communication
LMP	Link Manager Protocol
L2CAP	Logical Link Control and Adaptation Protocol
OSI	Open Systems Interconnection
GSM	Global System for Mobile Communication
SDP	Service Discovery Protocol

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Scope of the document

The most characteristic aspect of the InteropEHRate project is the fact that the EHR of a Citizen resides on her mobile device. The Citizen, using the dedicated mobile app (S-EHR app) is able to coordinate the exchange of health data between her mobile device and the governmental eHealth systems that, depending on the case, may act as sender or receiver of health data. Every exchange of health data among software applications adhering to InteropEHRate specifications, must be realized using the two data exchange protocols defined by the InteropEHRate platform: the D2D protocol for short range distance transmission, and the R2D protocol for remote (long range) transmission over the internet. Both the D2D and the R2D protocols will be used for standardizing how software applications exchange health data with the mobile application (S-EHR app) of the citizen, including additionally - in the case of the D2D protocol - the application (HCP app) of the healthcare practitioner.

The main focus of this document is to define the technical specifications of these two protocols, providing also detailed descriptions of their contexts of use, outlining a detailed description of their functionality, accompanied by an explanation of their separate purposes of existence. Moreover, the deliverable describes the research that was undertaken for each different field regarding short range and remote, data - and health data - exchange, for specifying each corresponding protocol.

The D2D is based on a short range communication protocol (i.e. Bluetooth), and it specifies a series of bluetooth messages regarding the information that is being exchanged (e.g. in terms of successful or failed data exchange) and healthcare related data between a healthcare practitioner (utilizing a web application (HCP app)) and a citizen (utilizing a mobile application (S-EHR app)), without the usage of internet connection. Having in mind that there exist several short-range communication techniques that could be used for the process of exchanging data without using internet connection (e.g. Wi-Fi Direct, Bluetooth Low Energy, Bluetooth Classic), upon specific research, the second version of the D2D protocol is still specified based on the Bluetooth Classic. The reason for this choice is explained in Section 3, and consequently any developer that would like to use the D2D protocol can be based on the listed operations and exchanged messages, in order to implement the corresponding data exchange process.

Differently from the D2D protocol, the R2D protocol works over the internet and defines the set of operations for acquiring the health data of a citizen from an EHR (National EHR, or EHR of hospitals). Moreover, it is used for initial import of health data to the mobile device and for the subsequent periodic synchronization operations. It should be noted that the R2D protocol is not only used for importing data from an EHR, but also to upload data to the S-EHR Cloud, a platform available to the citizens for periodic backup of their encrypted data. Data available in the S-EHR Cloud may also be available to authorized hospitals in case of emergency, if the citizen is unable to interact with the healthcare practitioner.

To this end, it should be also mentioned that the current document received as input the reference architecture as elaborated in deliverable D2.5 - InteropEHRate Architecture - V2 [D2.5], and the user requirements that were analysed and presented in deliverable D2.2 - User Requirements for cross-border HR integration - V2 [D2.2].

1.2. Intended audience

The current document is mainly intended for developers, and manufacturers that are interested in designing and building either mobile or web applications that need to exploit and reuse either the D2D or the R2D protocols, or both of these two protocols, in the context of their applications. As a result, this audience will be able to offer the aforementioned separate functionalities in their developed applications, since both the D2D and the R2D protocols can be easily integrated and reused by other systems and applications. Apart from this, the document is intended to researchers as well, as they may be interested in understanding the way that these two protocols exchange messages (and their content), be influenced by them, and possibly extending and updating their specification.

1.3. Structure of the document

The current document is organized in the following Sections:

- Section 1 (the current section) introduces the overall concept of the document, defining its scope, intended audience, and relation to the other project tasks and reports.
- Section 2 deals with the vision and the common parts of the D2D and the R2D protocols, showcasing the common data model in which both protocols are based upon for the overall data exchange process.
- Section 3 outlines the short range health data exchange protocol (D2D), describing in deep detail the purpose of the existence of the protocol, whereas stating the research that was undertaken for concluding into the design of the protocol, and the overall description of it.
- Section 4 describes the remote health data exchange protocol (R2D), explaining in deep detail the purpose of the existence of the protocol, whereas stating the research that was undertaken for concluding into the design of the protocol, and the overall description of it. It also describes the R2D protocol and its interaction with the Cloud, providing description of the APIs, the conceptual data model and some sample interactions for storing data on the cloud and retrieving data from the cloud.
- Section 5 outlines the conclusions of the current document, including the updates and the future development plans for the two protocols.

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

With regards to the previous version of the deliverable (D4.1 Specification of remote and D2D protocol and APIs for HR exchange V1 [D4.1]), the following changes have been performed for each separate section of the current document.

Section 1 - Introduction:

- Scope of the document: a minor text update has been provided regarding the overall purpose and usage of the D2D and R2D protocols.
- Structure of the document: The structure has been updated since two (2) new chapters have been introduced, regarding the D2D and R2D Protocols vision and common parts, as well as the R2D Cloud.

Section 2 - D2D and R2D Protocols vision and common parts:

- This section is a completely new chapter where the document describes the overall vision of the two protocols, and a conceptual common data model in which the two protocols are based upon for specifying and identifying the data that is being exchanged.

Section 3 - D2D Protocol: Short Range Health Data Exchange:

- D2D Protocol Scope: a minor text update has been provided regarding the scope of the D2D protocol
- D2D Protocol Description:
  - The section of the Conceptual D2D has been updated for the current version of the D2D protocol specification, including updated interfaces on both the data that their operations are taking as parameters and the data they are returning, based upon the conceptual data model.
  - The section of the D2D interface interactions has been updated for the current version of the D2D protocol specification, including new interactions between the involved interfaces, where the connection interactions are simplified and the overall interactions are split based on their functionality.
  - The section of the D2D over Bluetooth has been updated for the current version of the D2D protocol specification, where apart from the section that describes the Bluetooth Profile that is being used for performing the connection on top of Android Operating System, a new section is added for describing the Bluetooth Profile that is being used for performing the connection on top of Apple devices Operating System.
- D2D Commands & Replies Specification: a major update has been provided on the commands and replies that are being exchanged for the purposes of the D2D protocol, based upon the conceptual data model

Section 4 - R2D Protocol: Remote Health Data Exchange:

- Related Work: The state of the art has been updated adding projects or initiatives like Argonaut, IHE, IPSP and IPA.
- R2D Protocol Conceptual Description:
  - Conceptual R2D API: In this section some operations have been revised and the operation nextPage has been removed.
  - R2D over FHIR: In this section, the technical specifications have been revised due to the definition of the FHIR [HL7 FHIR] InteropEHRate Interoperability profile. Moreover, the format of the specifications has been completely changed, and the whole section has been re-written. New specifications regarding search narrowing and raised exceptions have been added.
  - R2D Cloud Conceptual Description:This section is a completely new addition where the document describes the R2D Cloud APIs, the conceptual data model and some sample interactions for storing data on the cloud, while also retrieving data from the cloud.

Section 5 - Conclusion and Next Steps:

- Minor updates have been provided based on the current conclusions and our future goals for the specification of the D2D and R2D protocols.

2. D2D AND R2D PROTOCOLS VISION AND COMMON PARTS

"True healthcare interoperability means that information can flow wherever it is needed, across borders and health systems" **[IHE IPS]**.

In the InteropEHRate standard architecture (i.e. the European architecture for health data exchange proposed by the project) [D2.5], data flows from an EHR system of a healthcare organisation in Country A to the EHR of another healthcare organization in Country B, with the mediation of the citizen. All data transfers are performed over the two InteropEHRate protocols: R2D and D2D, while the operations depicted in Figure 1, do not happen simultaneously. Firstly, the citizen is importing (using R2D) her health records stored within the EHR of one country (e.g. the National EHR system of her Country) to her smart mobile device. Only after this operation has been executed, she is able to transfer her health records (using D2D) to an HCP in Country B or execute a backup to the InteropEHRate Cloud (using R2D Cloud).

All the operations are coordinated by the citizen using the S-EHR app, while the citizen is under the total control of her health data: data exchange with R2D is possible only if the citizen provides a valid eIDAS identity, while data exchange with D2D is possible only after the citizen has been identified by the HCP receiving her and the citizen has given the needed consent to the healthcare organization of the HCP.

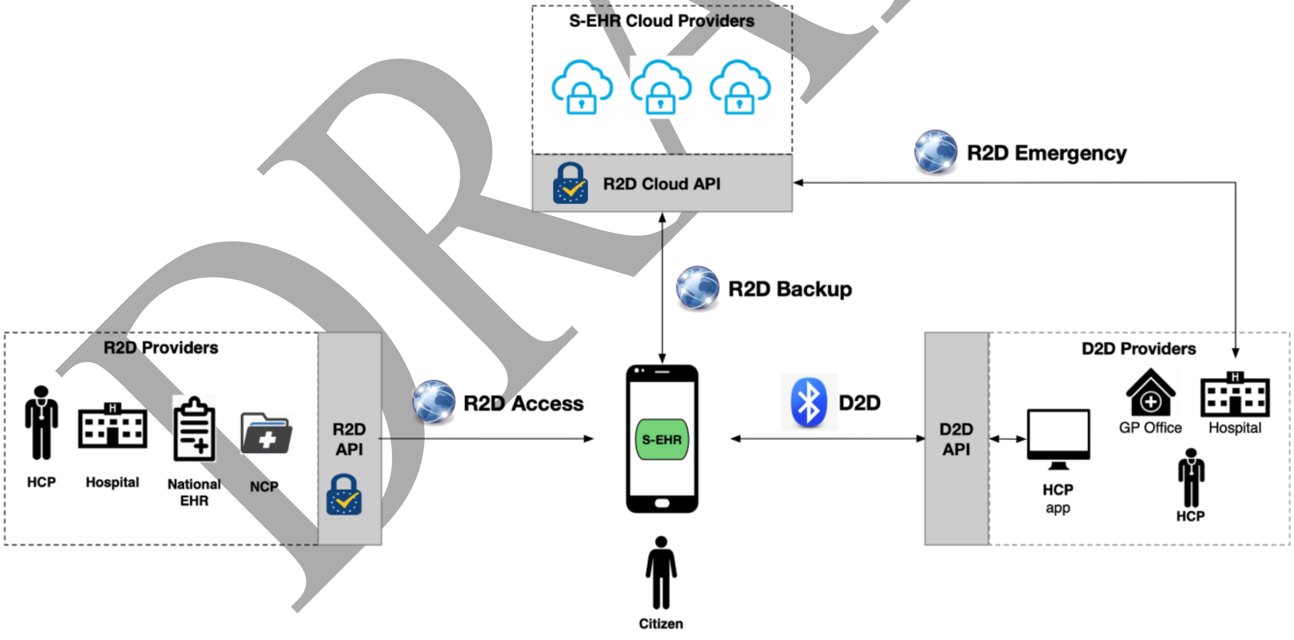
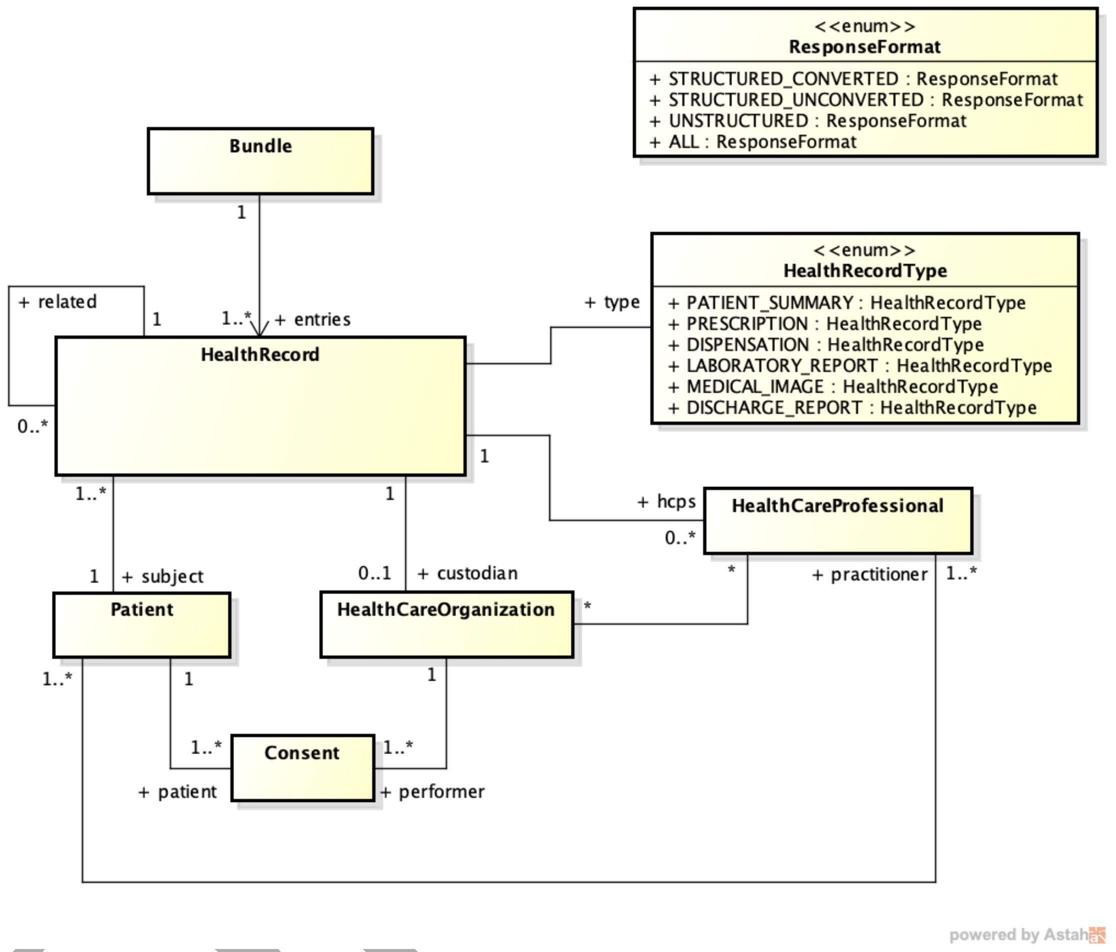


Figure 1 - Usage of the D2D and the R2D protocols

It must be mentioned that the D2D and the R2D protocols deal with similar types of health data, both protocols share the same data model, not only from a conceptual point of view, but also from a technical point of view. This aspect is fully described in the specification sections of the two protocols, however it is useful to anticipate a conceptual description of the common data model, to facilitate the understanding of the protocols.

2.1. Conceptual D2D and R2D Protocol Data Model

This section describes the conceptual data model used by both the D2D and R2D protocols. The following UML class diagram (Figure 2) depicts this data model, depicting the main classes and their relationships (each class is fully described in the specific table):



<<enum>>
ResponseFormat
+ STRUCTURED_CONVERTED : ResponseFormat
+ STRUCTURED_UNCONVERTED : ResponseFormat
+ UNSTRUCTURED : ResponseFormat
+ ALL : ResponseFormat
Bundle
1
1..*
+ entries
+ related
1
0..*
HealthRecord
+ type
<<enum>>
HealthRecordType
+ PATIENT_SUMMARY : HealthRecordType
+ PRESCRIPTION : HealthRecordType
+ DISPENSATION : HealthRecordType
+ LABORATORY_REPORT : HealthRecordType
+ MEDICAL_IMAGE : HealthRecordType
+ DISCHARGE_REPORT : HealthRecordType
1
+ hcps
0..*
HealthCareProfessional
1..*
1
1
+ subject
0..1
+ custodian
Patient
HealthCareOrganization
*
*
+ practitioner
1..*
1..*
1
1
1..*
Consent
1..*
+ patient
+ performer
powered by Astah

Figure 2 - Conceptual data model

The two central classes of the data model are (i) the Patient class that represents the citizen, and (ii) the HealthRecord class that represents the health data of one of the following types: patient summary, prescriptions, dispensation, laboratory report, medical image, or discharge report (health data types indicated by EU guidelines about European Electronic Health Record exchange format **[EUCBEHR REC]**).

In the following table (Table 1), a brief description of each class of the data model and their relationships are provided:

Name	Description
HealthRecord	It represents any possible health data of the patient (Patient Summary, laboratory analysis, set of medical images, etc.). Relationships:

	● *Patient*: this mandatory relationship is used to link an instance of HealthRecord to the patient whose health data refers to. ● *HealthCareProfessional*: this optional relationship links an instance of HealthRecord to all the instances of HealthCareProfessionals that have (or had) an active role as care providers in the life cycle of this HealthRecord. ● *HealthCareOrganization*: optional relationship defining the organization that is the custodian of an instance of HealthRecord. ● *HealthRecord*: optional relationship (named related) linking an instance of HealthData with another instance of HealthData (belonging to the same patient). ● *HealthRecordType*: mandatory relationship linking an instance of HealthRecord to an instance of HealthRecordType. This relationship is used to specify the type of an HealthRecord.
HealthRecordType	An enumeration representing the set of defined kinds of health data. Values of HealthRecordType have been defined in order to match the baseline of the EU Cross Border Health Data Exchange recommendations. Its values are: ● PATIENT_SUMMARY ● PRESCRIPTION ● LABORATORY_REPORT ● MEDICAL_IMAGES ● DISCHARGE_REPORT It will likely be extended in the future to support any health data that may also be represented using the FHIR standard.
Patient	It represents an individual receiving care or other health-related services. Relationships: ● *HealthRecord*: this mandatory relationship links an instance of Patient to his / her set of HealthRecords. ● *HealthCareProfessional*: this mandatory relationship links an instance of Patient to the set of nominated care providers. ● *Consent*: this relationship links a Patient to the set of Consent that she has given.
HealthCareProfessional	It represents an individual providing care or other health-related services to a patient. Relationships: ● *HealthCareOrganization*: this mandatory relationship links an instance of HealthCareProfessional to the set of HealthCareOrganizations instances where he/she provides care. ● *HealthRecord*: this optional relationship links an instance of HealthCareProfessional to an instance of HealthRecord (this link is necessary for reasons of care providing and really depends on the kind of HealthRecord). ● *Patient*: this mandatory relationship represents the link between an HealthCareProfessional and the set of Patients under his/her care.

HealthCareOrganization	It represents any organization offering health-related services. Relationships: ● *HealthCareProfessional*: this relationship represents the link between an HealthCareOrganization and the set of HealthCareProfessional that collaborates (or have collaborated) with the organization. ● *HealthRecord*: this relationship represents the link between an HealthCareOrganization and the set of HealthRecord where the organization acts as custodian. ● *Consent*: this relationship represents the link between an HealthCareOrganization and the set of Consent that have been given (by patients) to the organization itself.
Consent	It is used to express a consent regarding healthcare, given by a patient (grantee) to an healthcare provider (grantor). Relationships: ● *Patient*: this mandatory relationship links a Consent with the grantee (a Patient). ● *HealthCareOrganization*: this mandatory relationship links a Consent with the grantor (an HealthCareOrganization).
Bundle	It is a container for sets of instances of HealthRecord class. Relationships: ● HealthRecords: this mandatory relationships, links a Bundle to the set of contained HealthRecord.
ResponseFormat	It represents the format required by the client S-EHR of health data to be returned: ● *STRUCTURED_CONVERTED*: returned data are represented only in the common format (FHIR based or CDA based depending if the FHIR realization or the eHDSI realization is chosen) and using only semantic codes prescribed by the InteropEHRate profile [D2.7] (possibly obtained from the conversion of the codes (if any) in the original source data or by means of a semi-automatic information extraction process). Semantic codes belonging to the InteropEHRate profile will allow the client S-EHR to show the semantically annotated content in the natural language of the user. ● *STRUCTURED_UNCONVERTED*: returned data are represented in FHIR (without any specific restriction), using the same semantic codes (if any) as the original source data (no conversion of semantic codes is performed). ● *UNSTRUCTURED*: returned data are represented in a human readable document format (e.g. PDF). No information extraction or conversion of semantic code is performed. The exact content of the returned document is not defined by the R2D protocol. ● *ALL*: data are returned both in the same format returned by UNSTRUCTURED value and in the format STRUCTURED_CONVERTED (if the server is able to convert), or STRUCTURED_UNCONVERTED (if

	the server is not able to convert the semantic codes, but is able to transform the content in FHIR format). If the server is not able to perform any data transformations, only UNSTRUCTURED values are returned.

Table 1 - Description of the data model classes

Providing a concrete representation of this abstract data model is not an objective of this deliverable, but of deliverable [D2.7], R2D and D2D protocols defines only the set of APIs to invoke their services.

3. D2D PROTOCOL: SHORT RANGE HEALTH DATA EXCHANGE

The D2D protocol defines the set of operations that allow the exchange of health data between a S-EHR App and a short-range distance HCP App (i.e. in the context of a distance of up to 100m long) without the usage of internet connection (in opposition to the R2D protocol that is the protocol to exchange health data remotely in a long-range distance, including the usage of internet connection).

The next section of this document provides a detailed description on the technologies that have been studied and implemented towards the realization of the D2D protocol, the context in which the D2D protocol will be used, and of all the conditions that have influenced the decisions taken in the process carried out to define the D2D protocol specification.

3.1. D2D Protocol Scope

The purpose of the D2D protocol is to propose a series of Bluetooth messages regarding the information that is being exchanged (e.g. in terms of successful or failed data exchange) and healthcare related data, between a healthcare practitioner and a citizen, without the usage of internet connection. This protocol is based on short-range wireless technologies and in particular Bluetooth, to be adopted at EU level, for the secure exchange of health records between a smart mobile device and a health information system in the form of a web application. The smart device will use the S-EHR app (i.e. application that serves the purposes of the D2D from the side of the citizen), while the health information system will use the HCP app (i.e. application that serves the purposes of the D2D from the side of the healthcare practitioner), that will be used by the citizen and the healthcare practitioner accordingly (Figure 3).

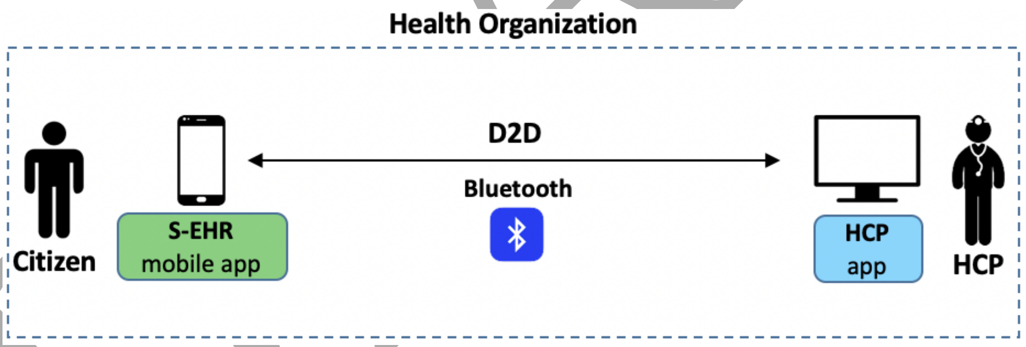


Figure 3 - D2D protocol scope

Bluetooth technology is most commonly associated with exchanging data between two bluetooth enabled devices in short-range distance (±100 meters), through which a bluetooth-enabled device as soon as it receives the initialization advertisement message from another bluetooth-enabled device, it establishes a connection to it, being thus able to exchange and display information between them, without needing any other technologies or types of connection (e.g. internet connection). Adopting a similar paradigm, the proposed D2D protocol will facilitate the information exchange between citizens (i.e. through smart mobile devices) and healthcare practitioners (i.e. through a computer with a bluetooth adapter), without the usage of central cloud services or internet connection.

In order to accomplish all the aforementioned tasks, the D2D protocol will define the Bluetooth services (represented by the interface TD2DI) to be offered by a healthcare organization to share health data contained in their EHR with the S-EHR app, as well as the Bluetooth services (represented by the interface MD2DI) to be offered by the S-EHR app for receiving requests from the HCP app of the Healthcare Organization Information System. The D2D protocol will also exploit D2D Security protocol (which interfaces are part of the R2DI and MD2DI) for performing Identity Management, Consent Management, and Authorization Management, that will be integrated into the overall functionality of the D2D protocol.

3.2. Related Work

3.2.1. Short Range Wireless Communication

Short-range wireless communications systems have to do with a wide range of scenarios, technologies and requirements. There is not any specific definition of such systems though one can always classify short-range wireless systems according to their typical reach or coverage. Henceforth, short-range wireless communications can be defined as the systems that aim to provide wireless connectivity within a local sphere of interaction. Short-range wireless systems deal with transferring of data from millimetres to a few hundreds of meters, which are not only providing wireless connectivity in the immediate proximity, but in a broader area they can be defined as technologies that are used to construct service access in local areas. Together with wide/metropolitan area cellular systems, short-range wireless systems represent the two main developing directions in today’s wireless communications scene having certain common parts as well as various differences from cellular systems. The main goal is to maximize the supported data throughput for both types of wireless networks, and as a result a detailed comparison between them is not easily implemented. There is a great deal of cooperation between short-range and cellular networks, and in many cases exploiting their full characteristics results in very efficient solutions.

The short-range wireless systems include NFC for very close connectivity, RFID ranging from centimetres up to a few hundred meters, WBAN providing wireless access in the close vicinity of a person, WPAN offering users in their surroundings of up to ten meters or so, WLAN, the de facto local connectivity for indoor scenarios covering typically up to a hundred meters around the access point, VAN involving distances of up to several hundred meters and WSN, reaching even further. Short-range wireless communication systems include a great diversity in possible air interface solutions, topologies as well as achievable data throughput and supported mobility. In general, short-range networks have ad hoc distributed architectures allowing direct and multi-hop connectivity among nodes. Centralized access is also possible, as in WLAN, which in fact supports both centralized and distributed topologies. Data throughput requirements for short-range wireless systems are very broad, from some hundred bits per second in simple RFID systems up to 10 Gbps and beyond in WLAN systems, for instance.

One of the key requirements for short-range wireless systems is low power consumption, particularly taking into account that transceivers are in many cases battery-driven. Particular attention is necessary when designing short-range wireless devices while minimizing the power consumption at the three basic conditions of the transceiver, namely transmitting, receiving and idle states. Low cost is another important factor to take into consideration when designing short-range wireless systems. Furthermore, minimizing size and weight are also quite often imperative engineering principles that need to be applied by the designer of such communication systems. Another important aspect of short-range wireless communications networks is their relationship to other existing wireless networks and their possible interaction.

These days, there is more wireless technology in use than ever before. Wireless technology is portable, easy to install, flexible and eliminates the cost of expensive wiring. With the explosion of available wireless devices, there has also been a big increase of wireless protocols and standards to support all of that technology. These include several short-range wireless communication technologies that transmit shorter distances than other long range technologies, making them great for certain applications. Below is a short list of the most commonly used short-range wireless communication standards and technologies.

ANT+ [ANT+]

ANT and ANT+ are sensor network technologies used for collecting and transferring sensor data. This short-range wireless communication technology is a type of personal-area network (PAN) that features low power consumption and long battery life. It divides the 2.4 GHz band into 1 MHz channels and accommodates multiple sensors. It is primarily used for short-range, low-data-rate sensor applications such as sports monitors, wearables, wellness products, home health monitoring, vehicle tire pressure sensing and in household items that can be controlled remotely such as TVs, lights and appliances. It can be configured to spend long periods in a low-power "sleep" mode (consuming of the order of microamps of current), wake up briefly to communicate (when consumption rises to a peak of 22mA (at -5dB) during reception and 13.5mA (at -5 dB) during transmission) and return to sleep mode. Average current consumption for low message rates is less than 60 microamps on some devices. Each ANT channel consists of one or more transmitting nodes and one or more receiving nodes, depending on the network topology. Any node can transmit or receive, so the channels are bi-directional. ANT accommodates three types of messaging: broadcast, acknowledged, and burst.

- Broadcast is a one-way communication from one node to another (or many). The receiving node(s) transmit no acknowledgement, but the receiving node may still send messages back to the transmitting node. This technique is suited to sensor applications and is the most economical method of operation.
- Acknowledged messaging confirms receipt of data packets. The transmitter is informed of success or failure, although there are no retransmissions. This technique is suited to control applications.
- Burst messaging is a multi-message transmission technique using the full data bandwidth and running to completion. The receiving node acknowledges receipt and informs of corrupted packets that the transmitter then re-sends. The packets are sequence numbered for traceability. This technique is suited to data block transfer where the integrity of the data is paramount.

Bluetooth/ BLE [BLUETOOTH]

Bluetooth is covered by the IEEE 802.15.1 standard. Originally created as an alternative to cabled RS-232, Bluetooth is now used to send data from PANs and fixed and mobile devices. This plug-and-play technology utilizes the 2.4 -2.485 GHz band and has a standard range of 10 meters, but it can be extended to 100 meters at maximum power with a clear path. BLE has a simpler design and is a direct competitor of ANT+, focusing on health and medical applications.

Bluetooth is a packet-based short-range wireless communication technology with a master/slave architecture, where the devices can switch roles, by agreement, and the slave can become the master. One master may communicate with up to seven slaves in a piconet. All devices share the master's clock. Packet exchange is based on the basic clock, defined by the master, which ticks at 312.5 µs intervals. Two clock ticks make up a slot of 625 µs, and two slots make up a slot pair of 1250 µs. In the simple case of single-slot packets, the master transmits in even slots and receives in odd slots. The slave, conversely, receives in even slots and transmits in odd slots. Packets may be 1, 3 or 5 slots long, but in all cases the master's transmission begins in even slots and the slave's in odd slots. The master chooses which slave device to address by switching rapidly from one device to another in a round-robin fashion. Since it is the master that chooses which slave to address, whereas a slave is supposed to listen in each receive slot, being a master is a lighter burden than being a slave. Being a master of seven slaves is possible; being a slave of more than one master is possible. The specification is vague as to required behaviour in scatternets. To use Bluetooth wireless technology, a device must be able to interpret certain Bluetooth profiles, which are definitions of possible applications and specify general behaviours that Bluetooth-enabled devices use to communicate with other Bluetooth devices. These profiles include settings to parameterize and to control the communication from the start. Adherence to profiles saves the time for transmitting the parameters anew before the bi-directional link becomes effective. There are a wide range of Bluetooth profiles that describe many different types of applications or use cases for devices. Bluetooth has different versions, with either minor or major differences among them, based on their implementation and usage. The version history of Bluetooth is depicted below:

- Bluetooth 1.0 and 1.0B: This version had many problems, and manufacturers had difficulty making their products interoperable. They included mandatory Bluetooth hardware device address transmission in the Connecting process, which was a major setback for certain services planned for use in Bluetooth environments
- Bluetooth 1.1: This version was ratified as an IEEE Standard 802.15.1–2002. It included many error fixes from the v1.0B specifications, including the possibility of non-encrypted channels, and Signal Strength Indicator for the received data.
- Bluetooth 1.2: This version included faster Connection and Discovery, adaptive frequency-hopping spread spectrum, which improved the resistance to radio frequency interference by avoiding the use of crowded frequencies in the hopping sequence. It also included higher transmission speeds than in v1.1, and improved quality of audio links by allowing retransmissions of corrupted packets, and increasing audio latency to provide better concurrent data transfer.
- Bluetooth 2.0: This version of the Bluetooth Core Specification was released in 2004. The main difference is the introduction of an EDR for faster data transfer. The bit rate of EDR is 3 Mbit/s, although the maximum data transfer rate is 2.1 Mbit/s. EDR can provide lower power consumption through a reduced duty cycle. The specification is published as Bluetooth v2.0 + EDR, which implies that EDR is an optional feature. Aside from EDR, the v2.0 specification contains other minor improvements, and products may claim compliance to "Bluetooth v2.0" without supporting the higher data rate.
- Bluetooth 2.1: This version of Bluetooth Core Specification includes secure simple pairing, improving the pairing experience for Bluetooth devices, while increasing the use and strength of security. It also allows improvements, including extended inquiry response, which provides more information during the inquiry procedure to allow better filtering of devices before connection, and sniff subrating that reduces the power consumption in low-power mode.
- Bluetooth 3.0: This version of the Bluetooth Core Specification was adopted in 2009. Bluetooth v3.0 provides theoretical data transfer speeds of up to 24 Mbit/s, though not over the Bluetooth link itself. Instead, the Bluetooth link is used for negotiation and establishment, and the high data rate traffic is carried over a collocated 802.11 link. The main new feature is Alternative MAC/PHY, the addition of 802.11 as a high-speed transport. The high-speed part of the specification is not mandatory, and hence only devices that display the "+HS" logo actually support Bluetooth over

802.11 high-speed data transfer. A Bluetooth v3.0 device without the "+HS" suffix is only required to support features introduced in Core Specification Version 3.0 or earlier Core Specification Addendum 1.

- Bluetooth 4.0: This version of the Bluetooth Core Specification version 4.0 (Bluetooth Smart) has been adopted as of 2010. It includes Classic Bluetooth, Bluetooth high speed and BLE short-range wireless communication technologies. Bluetooth high speed is based on Wi-Fi, and Classic Bluetooth consists of legacy Bluetooth short-range wireless communication technologies. BLE, previously known as Wibree, is a subset of Bluetooth v4.0 with an entirely new short-range wireless communication technology stack for rapid build-up of simple links. As an alternative to the Bluetooth standard short-range wireless communication technologies that were introduced in Bluetooth v1.0 to v3.0, it is aimed at very low power applications powered by a coin cell. Chip designs allow for two types of implementation, dual-mode, single-mode and enhanced past versions. Compared to Classic Bluetooth, BLE is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range. In a single-mode implementation, only the low energy short-range wireless communication technology stack is implemented. The compliant architecture shares all of Classic Bluetooth's existing radio and functionality resulting in a negligible cost increase compared to Classic Bluetooth. Cost-reduced single-mode chips, which enable highly integrated and compact devices, feature a lightweight Link Layer providing ultra-low power idle mode operation, simple device discovery, and reliable point-to-multipoint data transfer with advanced power-save and secure encrypted connections at the lowest possible cost. General improvements in version 4.0 include the changes necessary to facilitate BLE modes, as well as the GATT and Security Manager services with Advanced Encryption Standard.
- Bluetooth 4.1: This version is an incremental software update to Bluetooth Specification v4.0, and not a hardware update. The update incorporates some Bluetooth Core Specifications and adds new features that improve consumer usability. These include increased co-existence support for LTE, bulk data exchange rates, and aid developer innovation by allowing devices to support multiple roles simultaneously. New features of this specification include mobile Wireless Service Coexistence Signalling, Train Nudging and Generalized Interlaced Scanning, Low Duty Cycle Directed Advertising, L2CAP Connection Oriented and Dedicated Channels with Credit-Based Flow Control, Dual Mode and Topology, LE Link Layer Topology, 802.11n PAL, Audio Architecture Updates for Wide Band Speech, Fast Data Advertising Interval, and Limited Discovery Time.
- Bluetooth 4.2: This version includes major areas of improvement in the domain of Low Energy Secure Connection with Data Packet Length Extension. It also includes Link Layer Privacy with Extended Scanner Filter Policies, Internet Protocol Support Profile version 6 ready for Bluetooth Smart things to support connected home, and supports the fact that older Bluetooth hardware may receive 4.2 features such as Data Packet Length Extension and improved privacy via firmware updates.
- Bluetooth 5.0: This version has new features mainly focusing on emerging Internet of Things technology. Bluetooth 5.0 provides for BLE, options that can double the speed (2 Mbit/s burst) at the expense of range, or up to fourfold the range at the expense of data rate, and eightfold the data broadcasting capacity of transmissions, by increasing the packet lengths. The increase in transmissions could be important for Internet of Things devices, where many nodes connect

throughout the whole house. Bluetooth 5 adds functionality for connectionless services such as location-relevant navigation of low-energy Bluetooth connections.

- Bluetooth 5.1: This version supports major areas of improvement including AoA and AoD which are used for location and tracking of devices, Advertising Channel Index, GATT Caching.

EnOcean [ENOCEAN]

This system is self-powered and able to wirelessly transmit data by using ultra-low power consumption and energy collecting technology. Instead of a power supply, EnOcean's wireless sensor technology collects energy from the air. Energy from the environment, such as light, pressure, kinetic motion and temperature differences, is harvested and used to transmit a signal up to 30 meters indoors using a very small amount of energy. In the US, EnOcean runs on the 315 MHz and 902 MHz bands. In Europe, it uses the 868 MHz frequency band and in Japan, it operates on the 315 MHz and 928 MHz frequency bands. EnOcean technology is based on the energetically efficient exploitation of slight mechanical motion and other potentials from the environment, such as indoor light and temperature differences, using the principles of energy harvesting. In order to transform such energy fluctuations into usable electrical energy, electromagnetic, solar, and thermoelectric energy converters are used. EnOcean-based products (such as sensors and light switches) perform without batteries and are engineered to operate maintenance-free. The radio signals from these sensors and switches can be transmitted wirelessly over a distance of up to 300 meters in the open and up to 30 meters inside buildings. Early designs from the company used piezo generators, but were later replaced with electromagnetic energy sources to reduce the operating force (3.5 newtons), and increase the service life to 100 operations a day for more than 25 years. EnOcean wireless data packets are relatively small, with the packet being only 14 bytes long and are transmitted at 125 kbit/s. RF energy is only transmitted for the 1's of the binary data, reducing the amount of power required. Three packets are sent at pseudo-random intervals reducing the possibility of RF packet collisions. Modules optimized for switching applications transmit additional data packets on release of push-button switches, enabling other features such as light dimming to be implemented. The transmission frequencies used for the devices are 902 MHz, 928.35 MHz, 868.3 MHz and 315 MHz. In 2017, EnOcean unveiled a series of light switches utilizing BLE radio (2.4 GHz).

NFC [NFC]

NFC is an ultra-short-range technology created for contactless communication between devices. It is often used for secure payment applications, fast passes and similar applications. Operating on the 13.56 MHz ISM frequency, NFC has a maximum range of around 20 cm, which provides a more secure connection that is usually encrypted. Many smart phones already include an NFC tag. NFC short-range wireless communication technologies established a generally supported standard. When one of the connected devices has Internet connectivity, the other can exchange data with online services. NFC-enabled portable devices can be provided with application software, for example, to read electronic tags or make payments when connected to an NFC-compliant apparatus. Earlier close-range communication used technology that was proprietary to the manufacturer for applications such as stock tickets, access control and payment readers. Like other "proximity card" technologies, NFC employs electromagnetic induction between two loop antennas when NFC-enabled devices—for example a smartphone and a printer—exchange information, operating within the globally available unlicensed radio frequency ISM band of 13.56 MHz on ISO/IEC 18000-3 air interface at rates ranging from 106 to 424 kbit/s. NFC tags contain data and are typically read-only, but may be writable. They can be custom-encoded by their manufacturers or use NFC

Forum specifications. The tags can securely store personal data such as debit and credit card information, loyalty program data, PINs and networking contacts, among other information. The NFC Forum defines four types of tags that provide different communication speeds and capabilities in terms of configurability, memory, security, data retention and write endurance. Tags currently offer between 96 and 4,096 bytes of memory.

The two modes for NFC are:

- Passive: The initiator device provides a carrier field and the target device answers by modulating the existing field. In this mode, the target device may draw its operating power from the initiator-provided electromagnetic field, thus making the target device a transponder.
- Active: Both initiator and target device communicate by alternately generating their own fields. A device deactivates its RF field while it is waiting for data. In this mode, both devices typically have power supplies.

It should be noted that each full NFC device can work in three modes:

- NFC card emulation: It deals with NFC-enabled devices such as smartphones to act like smart cards, allowing users to perform transactions such as payment or ticketing.
- NFC reader/writer: It enables NFC-enabled devices to read information stored on inexpensive NFC tags embedded in labels or smart posters.
- NFC peer-to-peer: It enables two NFC-enabled devices to communicate with each other to exchange information in an ad hoc fashion.

RFID [RFID]

RFID uses small, flat, cheap tags that can be attached to anything and used for identification, location, tracking and inventory management. When a reader unit is nearby, it transmits a high-power RF signal to the tags and reads the data stored in their memory. Low frequency RFID uses the 125-134 kHz band, high frequency RFID uses the 13.56 MHz ISM band and Ultra-high frequency RFID uses the 125-134 kHz band. With multiple ISO/IEC standards available for RFID, this technology has replaced bar codes in some industries. It uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically stored information. Passive tags collect energy from a nearby RFID reader's interrogating radio waves. Active tags have a local power source (such as a battery) and may operate hundreds of meters from the RFID reader. Unlike a barcode, the tag need not be within the line of sight of the reader, so it may be embedded in the tracked object. RFID is one method of automatic identification and data capture.

ZigBee [ZIGBEE]

ZigBee is the standard of the ZigBee Alliance. The path of a message in this network zig-zags like a bee, hence the name. It is a software short-range wireless communication technology that uses the 802.15.4 transceiver as a base and is meant to be cheaper and simpler than other WPANs, like Wi-Fi or Bluetooth. ZigBee is able to build large mesh networks for sensor monitoring, handling up to 65,000 nodes, and it can also support multiple types of radio networks such as point-to-point and point-to-multipoint. It has a data rate of 250 kB/s and can transfer wireless data over a distance of up to 100m. ZigBee can be used for a range of applications including remote patient monitoring, wireless lighting and electrical meters, traffic management systems, consumer TV and factory automation, to name a few.

Zigbee devices are of three kinds:

- Zigbee Coordinator: The most capable device, the Coordinator forms the root of the network tree and might bridge to other networks. There is precisely one Zigbee Coordinator in each network since it is the device that started the network originally (the Zigbee LightLink specification also allows operation without a Zigbee Coordinator, making it more usable for off-the-shelf home products). It stores information about the network, including acting as the Trust Center & repository for security keys.
- Zigbee Router: As well as running an application function, a Router can act as an intermediate router, passing on data from other devices.
- Zigbee End Device: Contains just enough functionality to talk to the parent node (either the Coordinator or a Router); it cannot relay data from other devices. This relationship allows the node to be asleep a significant amount of the time thereby giving long battery life. A Zigbee End Device requires the least amount of memory, and, therefore, can be less expensive to manufacture than a Zigbee Router or Zigbee Coordinator.

The current Zigbee short-range wireless communication technologies support beacon and non-beacon enabled networks.

- In non-beacon-enabled networks, an unslotted CSMA/CA channel access mechanism is used. In this type of network, Zigbee Routers typically have their receivers continuously active, requiring a more robust power supply. However, this allows for heterogeneous networks in which some devices receive continuously, while others only transmit when an external stimulus is detected. The typical example of a heterogeneous network is a wireless light switch: The Zigbee node at the lamp may constantly receive, since it is connected to the mains supply, while a battery-powered light switch would remain asleep until the switch is thrown. The switch then wakes up, sends a command to the lamp, receives an acknowledgment, and returns to sleep. In such a network the lamp node will be at least a Zigbee Router, if not the Zigbee Coordinator; the switch node is typically a Zigbee End Device.
- In beacon-enabled networks, the special network nodes called Zigbee Routers transmit periodic beacons to confirm their presence to other network nodes. Nodes may sleep between beacons, thus lowering their duty cycle and extending their battery life. Beacon intervals depend on data rate; they may range from 15.36 milliseconds to 251.65824 seconds at 250 kbit/s, from 24 milliseconds to 393.216 seconds at 40 kbit/s and from 48 milliseconds to 786.432 seconds at 20 kbit/s. However, low duty cycle operation with long beacon intervals requires precise timing, which can conflict with the need for low product cost.

In general, the Zigbee short-range wireless communication technologies minimize the time the radio is on, so as to reduce power use. In beaconing networks, nodes only need to be active while a beacon is being transmitted. In non-beacon-enabled networks, power consumption is decidedly asymmetrical: Some devices are always active while others spend most of their time sleeping.

Wi-Fi Direct [WIFIDIRECT]

Wi-Fi Direct, initially called Wi-Fi P2P, is a Wi-Fi standard enabling devices to easily connect with each other without requiring a wireless access point. Wi-Fi Direct allows two devices to establish a direct Wi-Fi connection without requiring a wireless router. Hence, Wi-Fi Direct is single radio hop communication, not multihop wireless communication, unlike wireless ad hoc networks and mobile ad hoc networks. Wi-Fi ad hoc mode, however, supports multi-hop radio communications, with intermediate Wi-Fi nodes as packet relays. One advantage of Wi-Fi Direct is the ability to connect devices even if they are from different manufacturers. Only one of the Wi-Fi devices needs to be compliant with Wi-Fi Direct to establish a peer-to-peer connection that transfers data directly between them with greatly reduced setup. Wi-Fi Direct negotiates the link with a Wi-Fi Protected Setup system that assigns each device a limited wireless access point. The "pairing" of Wi-Fi Direct devices can be set up to require the proximity of a near field communication, a Bluetooth signal, or a button press on one or all the devices. When a device enters the range of the Wi-Fi Direct host, it can connect to it, and then gather setup information using a Protected Setup-style transfer. Connection and setup is so simplified that it may replace Bluetooth in some situations. Wi-Fi Direct-certified devices can connect one-to-one or one-to-many and not all connected products need to be Wi-Fi Direct-certified. One Wi-Fi Direct enabled device can connect to legacy Wi-Fi certified devices. The Wi-Fi Direct certification program is developed and administered by the Wi-Fi Alliance, the industry group that owns the "Wi-Fi" trademark. The specification is available for purchase from the Wi-Fi Alliance

Z-Wave [ZWAVE]
Z-Wave is a wireless communications short-range wireless communication technology used primarily for home automation. It is a mesh network using low-energy radio waves to communicate from appliance to appliance, allowing for wireless control of residential appliances and other devices. A Z-Wave system can be controlled via the Internet from a smartphone, tablet or computer, and locally through a smart speaker, wireless keyfob, or wall-mounted panel with a Z-Wave gateway or central control device serving as both the hub controller and portal to the outside. Z-Wave is designed to provide reliable, low-latency transmission of small data packets at data rates up to 100kbit/s. The throughput is 40kbit/s (9.6kbit/s using old chips) and suitable for control and sensor applications, unlike Wi-Fi and other IEEE 802.11-based wireless LAN systems that are designed primarily for high data rates. Communication distance between two nodes is about 30 meters (40 meters with 500 series chip), and with message ability to hop up to four times between nodes, it gives enough coverage for most residential houses. Modulation is by Manchester channel encoding. Z-Wave uses the Part 15 unlicensed ISM band. It operates at 868.42 MHz in Europe, at 908.42 MHz in North America and uses other frequencies in other countries depending on their regulations. This band competes with some cordless telephones and other consumer electronics devices, but avoids interference with Wi-Fi, Bluetooth and other systems that operate on the crowded 2.4 GHz band. The lower layers, MAC and PHY, are described by ITU-T G.9959 and fully backwards compatible. In 2012, the ITU included the Z-Wave PHY and MAC layers as an option in its G.9959 standard for wireless devices under 1 GHz. Data rates include 9600 bps and 40 kbps, with output power at 1 mW or 0 dBm. Devices can communicate to one another by using intermediate nodes to actively route around and circumvent household obstacles or radio dead spots that might occur in the multipath environment of a house. A message from node A to node C can be successfully delivered even if the two nodes are not within range, providing that a third node B can communicate with nodes A and C. If the preferred route is unavailable, the message originator will attempt other routes until a path is found to the C node. Therefore, a Z-Wave network can span much farther than the radio range of a single unit; however, with several of these hops a slight delay may be introduced between the control command and the desired result.

3.2.2. InteropEHRate D2D approach

A comparison study took place with specified criteria, for identifying the most ideal short-range wireless communication technology as a basis to realize the communication aspects of the envisioned D2D protocol. The short-range wireless communication technologies that have been considered include: Wi-Fi direct, Bluetooth v4.0, BLE, and NFC. These criteria included all of our needs and necessities as described in D2.2

deliverable [D2.2], in order to be compliant with the objectives of the D2D protocol, for the interoperable exchange of healthcare data between a S-EHR app and an application on the healthcare practitioner's personal computer (i.e. HCP App). In more detail, the comparison criteria were the following:

- Range (in centimetres/ meters)
- Data Rate (in bps)
- Security (Low, Medium, High based on the short-range wireless communication technology specification)
- Platform Applicability (referring to the mostly used platform operating systems (iOS, Android))
- Ease of use (Low, Medium, High based on the short-range wireless communication technology specification)
- Power Consumption (Low, Medium, High based on the short-range wireless communication technology specification)

	Wi-Fi Direct	Bluetooth v4.0	BLE	NFC
Range	Up to 180 m	Up to 100m	Up to 10m	Up to 4cm
Data Rate	Up to 250Mbps	Up to 25 Mbps	Up to 200kbps	Up to 424kbps
Security	High	High	High	Medium
Platform Applicability	Android: Host Wi-Fi network by Android or PC iOS: Manually join hosted network by the PC	Android: Applicable iOS: Needs certification under MFI program	Android: Applicable iOS: GATT-based API	Android: Applicable iOS: Applicable (data transfer by NFC packets to be understood)
Ease of Use	High	High	High	High
Power Consumption	High	Medium	Low	Low

Table 2 - Comparison between short-range wireless communication technologies

In terms of process, a list of requirements was circulated to the hospitals of the InteropEHRate consortium (i.e. users), in order to identify their needs and specifications. This list (including the users' answers) can be seen in the ANNEX of the current document. Based on the results that are displayed in Table 2, as well as the needs and the criteria that were set, the two most likely candidates short-range wireless communication technologies for the D2D protocol were Bluetooth v4.0 and BLE. In general, one of the primary advantages of Bluetooth is that it is enabled in almost any Android, Windows or Apple iOS device, allowing these devices to transmit data wirelessly. This advantage can be translated to more specific benefits including wirelessly connecting or "pairing" devices to create a wireless personal area network or WPAN, wireless synchronization, as well as conveniently sending and/or receiving files without the trouble of carrying and using cables or other hardware interfacing technology such as the USB standard or

Thunderbolt technology. That is why complementary devices have been developed and marketed because Bluetooth has seemingly become a standard feature of modern computers, specifically laptops and mobile devices. These devices included wireless speakers and headphones, smart devices such as smartwatches and other wearable technologies for monitoring activities, and Bluetooth-enabled smart home appliances and office equipment, among others. Moreover, pairing devices with built-in Bluetooth radio is considerably easy. There is no need to install additional software or drivers to establish communication between Bluetooth-enabled devices. There is also no rigorous setup process for two devices to communicate. The technology simplifies the entire pairing process by making enabled devices readily discoverable to one another as long as their Bluetooth radios are turned on, and they are within the coverage radius. In addition, the technology also includes a protocol for identifying services using the Service Discovery Protocol and Universal Unique Identifier to list down specific services or features of a particular device. These protocols allow another device to readily determine and display the name and class of a device it intends to pair with, as well as its services or features and technical information. Furthermore, Bluetooth technology is relatively energy efficient, thus promoting further the benefits and convenience that come with wireless data transmission. This is particularly true for the BLE or BLE standard. The ultra-low power requirement of BLE makes it ideal for small devices, including wearable technologies, in which minimal battery life requirement and small form factor are critical design and engineering considerations.

However, since both Bluetooth v4.0 and BLE were still two different candidates, this research was continued for identifying the most efficient solution for the InteropEHRate project. It was identified that BLE could be considered ideal for devices that must operate for long periods of time on small energy sources. In general, devices that utilize BLE today range in intelligence from heart rate monitors to smart watches, where BLE is particularly well suited for receiving small data updates, such as the current heart rate, every few seconds. In that case, the "pairing" process is also very simple since BLE devices can advertise themselves (at Peripheral state) and multiple BLE devices can be connected to another device (at Central state), such as a smartphone. However, BLE cannot be considered as the answer for every IoT device or scenario, especially in the case that our goal is to transmit moderately sized files (including texts of a few kilobytes, or images of a few megabytes (Mbytes)). As it can be seen in Table 3, the users demanded from the D2D protocol to exchange - among others, image files of a few megabytes, for the identified types of examinations.

Type of Examination	Weight evaluation by category (Mbytes)
Radiology	30
Ultrasound	10
Mammography	100
TC	300
RM	200
Angiographies	150

Positron Emission Tomography - PET	30
Other Nuclear Medicine examinations	30
Radiotherapy	30
Other	38
Reports	0.1

Table 3 - Size of files that will be exchanged through the D2D protocol

In more detail, with Bluetooth v4.0, throughput is as high as 25 Mbps, so once two devices are paired, sending 100 Mbytes of data would take a very small amount of time to be transmitted. In order to develop applications requiring Bluetooth access in Apple iOS devices, this can be cost-prohibitive, as Apple collects licensing and other fees in exchange for this functionality. It should be noted that this is not an issue on Android, but what was needed was the used devices to be compatible with both major mobile operating systems. In that scenario, BLE could be considered as the most convenient solution, since Apple iOS devices support BLE out of the box, without the need to enrol into the MFI program, and many Android devices are being manufactured with BLE as well. In the case of BLE, in order to send large data files, one needs to break apart the payload into 20-byte chunks, where on the receiving end, these chunks are recombined. Four such chunks can be sent per connection interval, which can vary from one device to the next. Android devices support intervals as low as 7.5 ms, while Apple iOS devices support connection intervals in the 30 ms range. At this rate, with BLE a 500KB image file would take over three minutes to transfer. Some workarounds for reducing this amount of data transfer time could be the usage of data partitioning, serialization, and compression which could help in reducing payload sizes, but in the case of InteropEHRate, this could not satisfy the users' needs and requirements who demand low transfer times and high data transfer rates. Consequently, due to its specifications, Bluetooth v4.0 has been identified as the most convenient and suitable candidate for short-range wireless communication data exchange, for being implemented in the D2D protocol in the InteropEHRate project.

3.3. D2D Protocol Description

The D2D protocol will define the bluetooth operations represented by the interfaces that will be offered by the mobile application and the healthcare's organization application speaking of the S-EHR app and the HCP app accordingly. These interfaces will be included in both applications accordingly, and will be used by the two main actors of the D2D protocol, the citizens and the HCPs. These two actors will be the only involved ones in the overall interaction, for exchanging the consent of accessing each one's personal data, the healthcare related data, and the evaluation data (i.e. healthcare data in the form of evaluation data, which are created after the examination of the HCP) accordingly.

In order for the D2D protocol to realize the communication and interaction with the two parties, two different interfaces will be designed and realized. The first interface (MD2DI) will be responsible for offering the bluetooth operations and services by the S-EHR app for interconnecting, exporting messages and receiving requests from the HCP app, while the second interface (TD2DI) will be responsible for offering the bluetooth operations and services by the HCP app for similar types of tasks from the S-EHR app. As discussed in Section 3.2, the overall communication will be based on the Bluetooth short-range wireless communication technology, hence the initial step of the overall D2D protocol will be the two involved actors to pair and bond their devices, prior to exchanging any messages. Figure 4 displays the overall connection between a citizen and an HCP, including the aforementioned interfaces, as well as the involved applications.

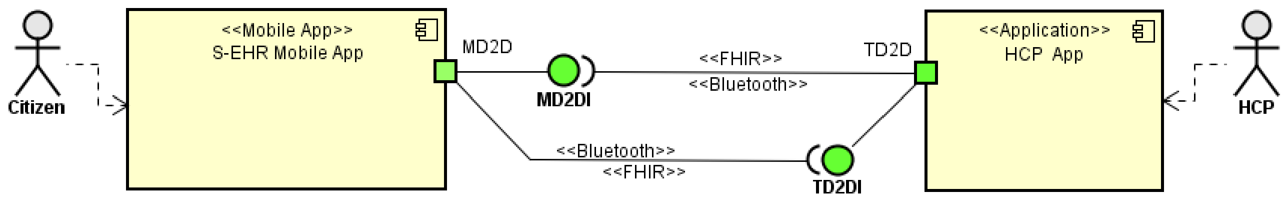


Figure 4 - D2D protocol interfaces

The rest of this section provides technical specifications (using UML language) of the D2D protocol. The section is structured in sub-sections, each one focused on a specific topic:

- Involved Applications: This includes a short description of the involved applications (S-EHR app and the HCP app), referring to the purpose and to a high-level description of their architecture.
- Conceptual D2D: This includes the conceptual description of the D2D protocol, defining the basic operations that this protocol should contain.
- D2D Interface Interactions: This includes the overall interactions between the involved interfaces for both the S-EHR app and the HCP app through a Sequence Diagram.
- D2D over Bluetooth: This includes the concrete description of the D2D protocol, regarding the Bluetooth profiles used for the connection and pairing process of the involved applications, and the messages that are exchanged according to the applications' interactions.
- D2D Commands & Replies: This includes the messages that will be exchanged within the context of the D2D between the involved applications, and the rules in which the exchanges messages should be compliant with.

3.3.1. Involved Applications

This section describes and lists the involved applications from the side of the S-EHR app and the HCP app.

3.3.1.1. S-EHR Application

A S-EHR app is any application installed on a personal mobile device, that is able to store the personal health data of a user in a secure (encrypted) way according to the constraints specified by [D3.3] and that supports the InteropEHRate protocols defined in the current deliverable and in [D4.8]. Different vendors may develop different S-EHRs applications. A S-EHR app contains health data of the user, produced and signed (for traceability and trustability) by the healthcare organization that produces them, but can also contain data directly stored and produced by citizens or by sensors. The provenance and author of each health data is unambiguously persisted on the S-EHR and the principles of integrity and non repudiation are guaranteed. More details on what is supported by a S-EHR app can be found in [D2.5].

3.3.1.2. HCP Application

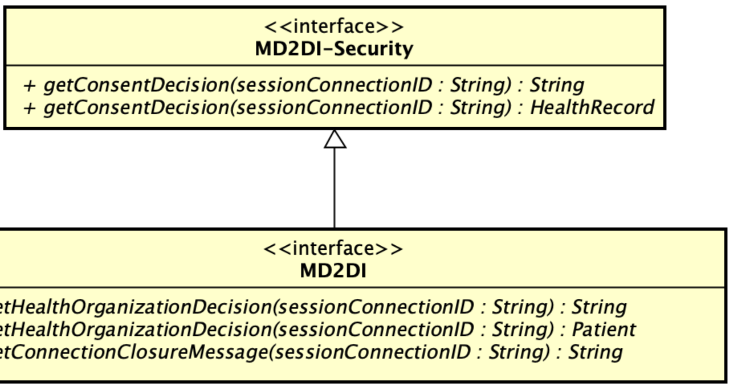
An HCP app is a software application designed to provide medical staff with the ability to access and operate patients' data from S-EHR apps, S-EHR Cloud and EHR of the Healthcare Organization. In other words, the HCP app is an application used by the HCPs to securely exchange health data of their EHRs with any S-EHR app and to read health data stored in S-EHR Cloud using the InteropEHRate protocols. More details on what is supported by an HCP app can be found in [D2.5].

3.3.2. Conceptual D2D

The Conceptual D2D defines the basic operations that the protocol should contain, for the exchange of data between the HCP and the citizen through the HCP app and the S-EHR app accordingly. The following figures (Figure 5, Figure 6) show two UML class diagrams representing the main interfaces of D2D, named MD2DI and TD2DI, from the side of the S-EHR App and the side of the HCP App.

MD2DI

MD2DI and MD2DI-Security are the names of the interfaces that are offered by the S-EHR app regarding the D2D protocol, containing the operations for letting the HCP app to perform tasks related to the S-EHR app, by invoking these operations.



<<interface>>
MD2DI-Security
+ getConsentDecision(sessionConnectionID : String) : String
+ getConsentDecision(sessionConnectionID : String) : HealthRecord
<<interface>>
MD2DI
+ getHealthOrganizationDecision(sessionConnectionID : String) : String
+ getHealthOrganizationDecision(sessionConnectionID : String) : Patient
+ getConnectionClosureMessage(sessionConnectionID : String) : String

Figure 5 - MD2D interface operations

The following tables provide a complete description of all the operations of the MD2DI and MD2DI-Security interfaces.

Operation getHealthOrganizationDecision

Name	getHealthOrganizationDecision
Description	This operation is invoked by the HCP app for getting the decision from the side of the S-EHR app, regarding whether the provided Health Organization identity is approved or not
Arguments	● sessionConnectionID: a string that identifies the connection between the two involved applications
Return Value	This operation will return, one of the two following options: ● A connection closure message (connectionClosureMessage) in the form of a String, indicating that the Health Organization identity was not

	approved, hence the connection will be closed ● The demographic data of the S-EHR app owner in the form of an object (Patient FL7 FHIR Resource as described in the D2D and R2D data model)
Exceptions	● Security exceptions related to the Bluetooth connection ● Network exceptions related to Bluetooth connection failure
Preconditions	● The Bluetooth connection exists in the mobile phone that includes the S-EHR app ● The smart mobile device is enabled with Bluetooth v4.0 and above ● The session is still valid

Operation getConsentDecision

Name	getConsentDecision
Description	This operation is invoked by the HCP app for getting the decision from the side of the S-EHR app, regarding whether the consent for getting the S-EHR app owner's healthcare data has been approved or not
Arguments	● sessionConnectionID: a string that identifies the connection between the two involved applications
Return Value	This operation will return, one of the two following options: ● A connection closure message (connectionClosureMessage) in the form of a String, indicating that the Health Organization identity was not approved, hence the connection will be closed ● The requested healthcare data of the S-EHR app owner (i.e. data that has been approved for sharing) in the form of an object (HealthRecord HL7 FHIR Resource as described in the D2D and R2D data model)
Exceptions	● Security exceptions related to the Bluetooth connection ● Network exceptions related to Bluetooth connection failure
Preconditions	● The Bluetooth connection exists in the mobile phone that includes the S-EHR app ● The smart mobile device is enabled with Bluetooth v4.0 and above ● The session is still valid

Operation getConnectionClosureMessage

Name	getConnectionClosureMessage
Description	This operation is invoked by the HCP app for getting the final message of connection closure, after the overall interaction has successfully ended with the provision of the evaluation data
Arguments	● sessionConnectionID: a string that identifies the connection between the two involved applications
Return Value	This operation will return a connection closure message (connectionClosureMessage) in the form of a String, indicating that the overall interaction was successfully completed
Exceptions	● Security exceptions related to the Bluetooth connection ● Network exceptions related to Bluetooth connection failure
Preconditions	● The citizen has already opened through her smart mobile device the Bluetooth connection ● The smart mobile device is enabled with Bluetooth v4.0 and above ● The session is still valid

TD2DI

TD2DI and TD2DI-Security are the names of the interfaces that are offered by the HCP app regarding the D2D protocol, containing the operations for facilitating the S-EHR app to perform tasks related to the HCP app, by invoking these operations.

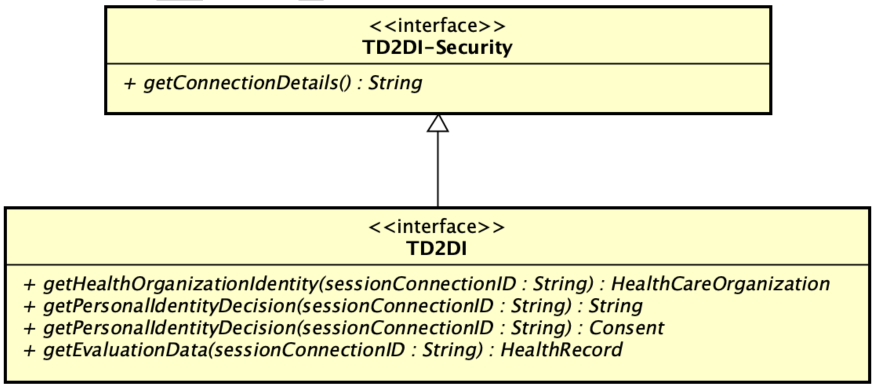

<<interface>>
TD2DI-Security
+ getConnectionDetails() : String
<<interface>>
TD2DI
+ getHealthOrganizationIdentity(sessionConnectionID : String) : HealthCareOrganization
+ getPersonalIdentityDecision(sessionConnectionID : String) : String
+ getPersonalIdentityDecision(sessionConnectionID : String) : Consent
+ getEvaluationData(sessionConnectionID : String) : HealthRecord

Figure 6 - TD2D interface operations

The following tables provide a complete description of all the operations of the TD2DI and TD2DI-Security interfaces.

Operation getConnectionDetails

Name	getConnectionDetails
Description	This operation is invoked by the S-EHR app for getting the Bluetooth connection session identifier, in the form of a String. This String will be used by both sides (S-EHR app and HCP app), for current connection identification purposes
Arguments	-
Return Value	This operation will return a unique session identifier (sessionConnectionID), in the form of a String that will be used by both sides (S-EHR app and HCP app), for current connection identification purposes
Exceptions	● Security exceptions related to the Bluetooth connection ● Network exceptions related to Bluetooth connection failure
Preconditions	● The computer that hosts the HCP app is enabled with Bluetooth v4.0 and above

Operation getHealthOrganizationIdentity

Name	getHealthOrganizationIdentity
Description	This operation is invoked by the S-EHR app for getting the Healthcare Organization identity details, for identifying whether the identity is valid or not
Arguments	● sessionConnectionID: a string that identifies the connection between the two involved applications
Return Value	This operation will return the Health Organization identity in the form of an object (HealthCareOrganization HL7 FHIR Resource as described in the D2D and R2D data model) containing details that identify the Health Organization
Exceptions	● Security exceptions related to the Bluetooth connection ● Network exceptions related to Bluetooth connection failure
Preconditions	● The computer that hosts the HCP app is enabled with Bluetooth v4.0 and above ● The session is still valid

Operation getPersonalIdentityDecision

Name	getPersonalIdentityDecision
Description	This operation is invoked by the S-EHR app for getting the decision from the side of the HCP app, regarding whether the provided Personal identity is approved or not
Arguments	● sessionConnectionID: a string that identifies the connection between the two involved applications
Return Value	This operation will return, one of the two following options: ● A connection closure message (connectionClosureMessage) in the form of a String, indicating that the Personal identity was not approved, hence the connection will be closed ● The temporary consent request of the HCP app owner in the form of an object (Consent HL7 FHIR Resource as described in the D2D and R2D data model), requesting access to specific types of data stored in the S-EHR app
Exceptions	● Security exceptions related to the Bluetooth connection ● Network exceptions related to Bluetooth connection failure
Preconditions	● The computer that hosts the HCP app is enabled with Bluetooth v4.0 and above ● The session is still valid

Operation getEvaluationData

Name	getEvaluationData
Description	This operation is invoked by the S-EHR app for getting the evaluation data from the side of the HCP app (after the examination), in order to be stored to the S-EHR app
Arguments	● sessionConnectionID: a string that identifies the connection between the two involved applications

Return Value	This operation will return the evaluation data of the examination to the S-EHR app, in the form of an object (HealthRecord HL7 FHIR Resource as described in the D2D and R2D data model), containing the results of the evaluation
Exceptions	● Security exceptions related to the Bluetooth connection ● Network exceptions related to Bluetooth connection failure ● Access exceptions related to the ownership of data (only evaluation data of the current S-EHR app owner can be provided)
Preconditions	● The computer that hosts the HCP app is enabled with Bluetooth v4.0 and above ● The session is still valid

3.3.3. D2D Interface Interactions

The interactions between the interfaces of the MD2D and the TD2D are explained and visualized through the Sequence Diagram of Figure 7.

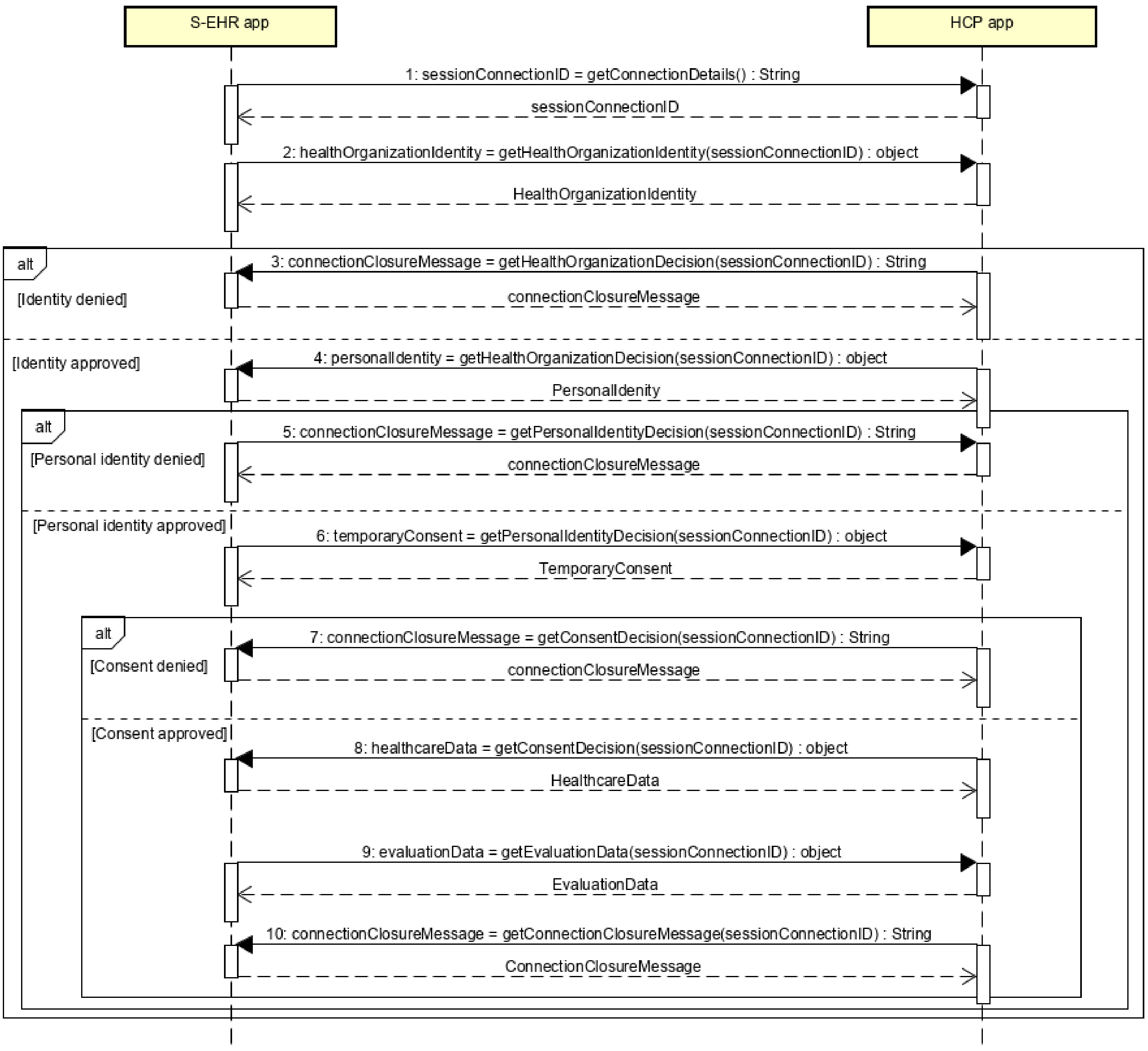

S-EHR app
HCP app
1: sessionConnectionID = getConnectionDetails() : String
sessionConnectionID
2: healthOrganizationIdentity = getHealthOrganizationIdentity(sessionConnectionID) : object
HealthOrganizationIdentity
alt
[Identity denied]
3: connectionClosureMessage = getHealthOrganizationDecision(sessionConnectionID) : String
connectionClosureMessage
[Identity approved]
4: personalIdentity = getHealthOrganizationDecision(sessionConnectionID) : object
PersonalIdenity
alt
[Personal identity denied]
5: connectionClosureMessage = getPersonalIdentityDecision(sessionConnectionID) : String
connectionClosureMessage
[Personal identity approved]
6: temporaryConsent = getPersonalIdentityDecision(sessionConnectionID) : object
TemporaryConsent
alt
[Consent denied]
7: connectionClosureMessage = getConsentDecision(sessionConnectionID) : String
connectionClosureMessage
[Consent approved]
8: healthcareData = getConsentDecision(sessionConnectionID) : object
HealthcareData
9: evaluationData = getEvaluationData(sessionConnectionID) : object
EvaluationData
10: connectionClosureMessage = getConnectionClosureMessage(sessionConnectionID) : String
ConnectionClosureMessage

Figure 7 - D2D protocol interactions

Following, a detailed description of the sequence diagram takes place. It should be noted that these steps are classified into five (5) main categories for the easier understanding of the D2D protocol specification. These categories are the (i) Bluetooth Connection, (ii) Demographic Data Exchange, (iii) Consent Exchange, (iv) Healthcare Data Exchange, and (v) Bluetooth Connection Closure.

(i) Bluetooth Connection:

- Step 1 - getConnectionDetails: The S-EHR app invokes this operation for getting the connection's unique session identifier, in the form of a String. This String will be used by both sides (S-EHR app and HCP app), for the current's connection identification purposes. It should be mentioned that the overall pairing and connection process is based on the Bluetooth short-range distance communication technology, and more specifically on the Bluetooth Serial Port Profile (SPP) (for Android OS devices) and Bluetooth Personal Area Network (PAN) (for iOS (i.e. Apple) devices). As a result, in the case of the D2D protocol specification it is avoided repeating the sequence of the exchanged messages for the Bluetooth (Bluetooth SPP and Bluetooth PAN profiles) pairing and connection process, since the D2D protocol is specified on top of them (i.e. reusing Bluetooth). More details regarding this process are provided in Section 3.3.4, where the involved protocols and interactions are thoroughly described.

(ii) Demographic Data Exchange

- Step 2 - getHealthOrganizationIdentity: The next step is for the S-EHR app to invoke this operation for getting the Healthcare Organization identity, for identifying whether the identity is valid or not.
- Step 3-4 - getHealthOrganizationDecision: As soon as the decision has been made about the Healthcare Organization identity, this operation is invoked by the HCP app for getting the decision from the side of the S-EHR app, regarding whether the provided Health Organization identity is approved or not. This operation will return, one of the two following options:
    - A connection closure message (connectionClosureMessage) in the form of a String, indicating that the Health Organization identity was not approved, hence the connection will be closed.
    - The demographic data of the S-EHR app owner in the form of an object (Patient).
- Step 5-6 - getPersonalIdentityDecision: In the case that the Healthcare Organization identity has been approved, this operation is invoked by the S-EHR app for getting the decision from the side of the HCP app, regarding whether the provided demographic data is approved or not. As in the previous cases, this operation will return, one of the two following options:
    - A connection closure message (connectionClosureMessage) in the form of a String, indicating that the demographic data was not approved, hence the connection will be closed.
    - The temporary consent request of the HCP app owner in the form of an object (Consent), requesting access to data stored in the S-EHR app.

(iii) Consent Exchange

- Step 7-8 - getConsentDecision: In the case that the temporary consent has been requested, this operation is invoked by the HCP app for getting the decision from the side of the S-EHR app, regarding whether the consent for getting the S-EHR app owner's data has been approved or not. As in the previous cases, this operation will return, one of the two following options:
    - A connection closure message (connectionClosureMessage) in the form of a String, indicating that the temporary consent was not approved, hence the connection will be closed.
    - The requested healthcare data of the S-EHR app owner (i.e. data that has been approved for sharing) in the form of an object (HealthRecord).

(iv) Healthcare Data Exchange

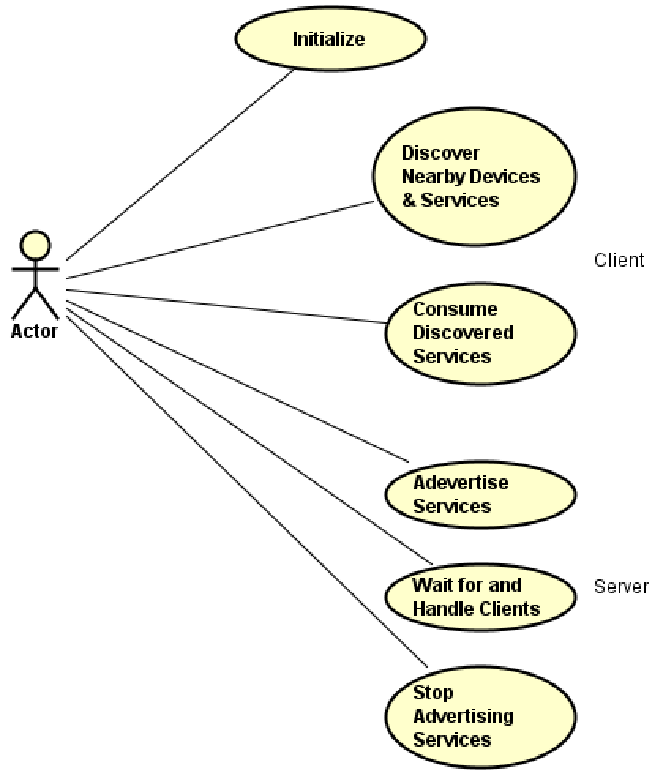
- Step 9 - getEvaluationData: In the case that the healthcare data has been provided, the S-EHR app invokes this method for getting the evaluation data (HealthRecord) from the side of the HCP app (after the examination), in order to be stored to the S-EHR app.

(v) Bluetooth Connection Closure

- Step 10 - getConnectionClosureMessage: The last steps includes this operation that has to be invoked by the HCP app for getting the final message of connection closure, after the overall interaction has successfully ended with the provision of the evaluation data.

3.3.4. D2D over Bluetooth

In order for the Bluetooth pairing and connection between the devices that host the S-EHR app and the HCP app to be realized, a specific process between the client and the server application must be followed. In our case, both the S-EHR app and the HCP app will behave as a true peer-to-peer endpoint by exposing both server and client behaviour. Typically, the pairing process that is being followed is based on the following figure (Figure 8).



Initialize
Discover Nearby Devices & Services
Client
Actor
Consume Discovered Services
Adevertise Services
Wait for and Handle Clients
Server
Stop Advertising Services

Figure 8 - Bluetooth pairing process

In general, in order for Bluetooth-enabled devices to transmit data between each other, they must first form a channel of communication using a pairing process. One device, a discoverable device, makes itself available for incoming connection requests. Another device finds the discoverable device using a service discovery process. After the discoverable device accepts the pairing request, the two devices complete a bonding process where they exchange security keys. The devices cache these keys for later use. After the pairing and bonding processes are complete, the two devices exchange information. When the session is complete, the device that initiated the pairing request releases the channel that had linked it to the discoverable device. The two devices remain bonded, however, so they can reconnect automatically during a future session as long as they're in range of each other and neither device has removed the bond.

This process can be summarized as follows:

1. Initialization: All bluetooth enabled applications must first initialize the Bluetooth stack.
2. Client: A client consumes the remote services. This is done by first discovering any nearby devices, and afterwards for each discovered device it searches for services of interest.
3. Server: A server makes the services available to clients. It registers them in the SDDB [SDDB], in effect advertising them. It then waits for incoming connections, accepts them as they come in, and serves the clients that make them. Finally, when the service is no longer needed, the application removes it from the SDDB.

In general, in order to use Bluetooth, a device must be compatible with the subset of Bluetooth profiles, meaning a set of rules, which are necessary to use the desired services defined by these rules. A Bluetooth profile is a specification regarding an aspect of Bluetooth-based wireless communication between devices that resides on top of the Bluetooth Core Specification and (optionally) additional protocols [BCS]. The way a device uses Bluetooth depends on its profile capabilities while the profiles provide standards which manufacturers follow to allow devices to use Bluetooth in the intended manner.

3.3.4.1. Android-side D2D over Bluetooth

Regarding the devices that run Android OS, since the D2D protocol will be based on the Bluetooth v4.0, a research was performed to the different Bluetooth profiles that are supported by the Bluetooth Core Specification version 4.0 [BSCv4.0], where it was identified that the Bluetooth Serial Port Profile [SPP] was the most suitable Bluetooth profile to be used for the purposes of the D2D protocol for the cases that the S-EHR device is running Android OS. Shortly, the Bluetooth SPP profile defines the requirements for Bluetooth devices that are necessary for setting up emulated serial cable connections using the RFCOMM protocol [RFCOMM] between the two peer devices. The requirements are expressed in terms of services provided to applications, and by defining the features and procedures that are required for interoperability between Bluetooth devices.

In the Bluetooth SPP profile, the following roles are defined:

- Device A (DevA) takes initiative to form a connection to another device (Initiator).
- Device B (DevB) waits for another device to take initiative to connect (Acceptor).

Figure 9 shows the protocols and entities used in the Bluetooth SPP profile in the case of the S-EHR and the HCP app, where the S-EHR app is considered as the Initiator while the HCP app is considered as the Acceptor. These roles can change between the two involved applications. Shortly, the Baseband [Baseband], LMP [LMP] and L2CAP [L2CAP] are the OSI layer [OSI] of the Bluetooth protocols. RFCOMM is the Bluetooth adaptation of GSM TS 07.10 [GSM], providing a transport protocol for serial port emulation, while SDP [SDP] is the Bluetooth Service Discovery Protocol.

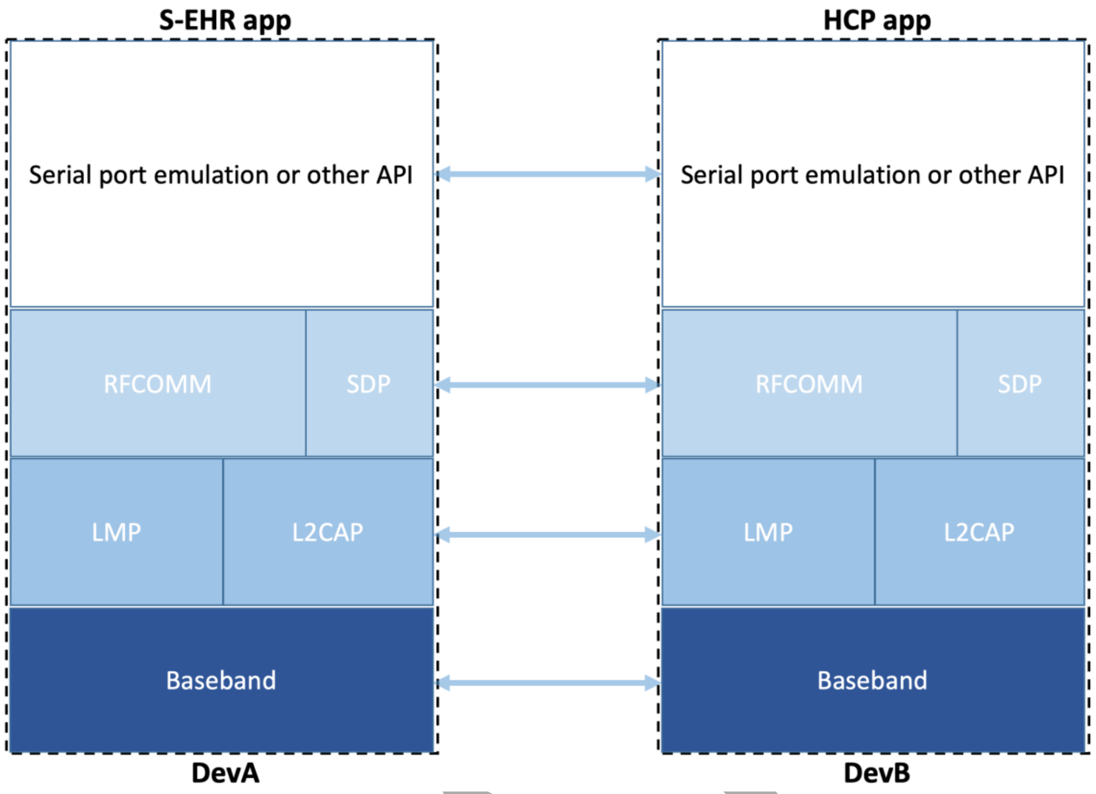


Figure 9 - Protocols and entities of the Bluetooth SPP profile

The required procedures that should be defined in the Bluetooth SPP profile are divided in three different steps, referring to the (i) establishment of a link and the set up of a Virtual Serial Connection, (ii) the acceptance of the link and the establishment of a Virtual Serial Connection, and (iii) the registration of the Service Record in a local SDP database.

(i) Establish Link and Set up Virtual Serial Connection
These steps are defined below:

1. Submit a query using SDP to find out the RFCOMM Server channel number of the desired application in the remote device.
2. Optionally, require authentication of the remote device to be performed. Also optionally, require encryption to be turned on.
3. Request a new L2CAP channel to the remote RFCOMM entity.
4. Initiate an RFCOMM session on the L2CAP channel.
5. Start a new data link connection on the RFCOMM session, using the aforementioned server channel number. After step 5, the virtual serial cable connection is ready to be used for communication between applications on both sides.

In the case that there already exists an RFCOMM session between the devices when setting up a new data link connection, the new connection must be established on the existing RFCOMM session. (This is equivalent to skipping over steps 3 and 4 above.)

(ii) Accept Link and Establish Virtual Serial Connection
These steps are defined below:

1. If requested by the remote device, take part in the authentication procedure and, upon further request, turn on encryption.
2. Accept a new channel establishment indication from L2CAP.
3. Accept an RFCOMM session establishment on that channel.
4. Accept a new data link connection on the RFCOMM session. This may trigger a local request to authenticate the remote device and turn on encryption, if the user has required that for the emulated serial port being connected to (and authentication/encryption procedures have not already been carried out).

(iii) Register Service Record in Local SDP Database
This procedure refers to registration of a service record for an emulated serial port (or equivalent) in the SDP database. This implies the existence of a Service Database, and the ability to respond to SDP queries.
All services/applications reachable through RFCOMM need to provide an SDP service record that includes the parameters necessary to reach the corresponding service/application.

3.3.4.2. iOS-side D2D over Bluetooth

Regarding the devices that run iOS (i.e. Apple's Operating System), a similar research was performed to the different Bluetooth profiles that are supported by the Bluetooth Core Specification version 4.0 [BSCv4.0], where it was identified that the Bluetooth Personal Area Networking Profile [PAN] was the most suitable Bluetooth profile to be used for the purposes of the D2D protocol for the cases that the S-EHR device is running iOS. It should be mentioned that Apple supports a short list of Bluetooth profiles, such as the Hands-Free Profile (HFP), The Phone Book Access Profile (PBAP), or the Advanced Audio Distribution Profile (A2DP), which however cannot serve the purposes of the D2D protocol [APPLE BT]. Among the supported Bluetooth profiles, the PAN profile is considered as the most appropriate for the current D2D protocol specification. Shortly, the PAN profile describes how two or more Bluetooth enabled devices can form an ad-hoc network and how the same mechanism can be used to access a remote network through a network access point. For the PAN profile, two general scenarios are usually discussed: (1) the Network access points, and (2) the Group Ad-hoc Networks. Each of the scenarios has unique network architecture and unique network requirements, but all are various combinations of a PAN profile.
In the case of the D2D protocol, the Group Ad-hoc networking scenario is being considered, as a collection of mobile hosts that cooperatively create an ad-hoc wireless network without the usage of additional networking hardware or infrastructure. In that case, the PAN profile focuses on a simple personal ad-hoc networking scenario consisting of a single Bluetooth piconet with connections between two or more Bluetooth devices. A piconet consists of one Bluetooth device operating as a piconet master, communicating with one or more (max. seven) of active Bluetooth devices operating as slaves. Communications in a piconet are between the master and the slaves, being under the control of the master, either in a point-to-point or a point-to-multipoint fashion.

In the Bluetooth PAN profile, the following roles are defined:

- Group Ad-hoc Network (GN) and GN service can be considered as a Bluetooth device that supports the GN service that is able to forward Ethernet packets to each of the connected Bluetooth devices, speaking about the PAN users, as needed. The Group Ad-hoc Network and the PAN User exchange data using the Bluetooth Network Encapsulation Protocol (BNEP). It should be mentioned that the Group Ad-hoc Networks do not provide access to any additional networks. Instead, Group Ad-hoc

Networks are intended to allow a group of devices to form temporary networks and exchange of information.

- PAN User (PANU) is the Bluetooth device that uses either the NAP or the GN service. PANU supports the client role for both the NAP and GN roles.

Figure 10 shows the protocols and entities used in the Bluetooth PAN profile. The Baseband, LMP and L2CAP are the parts of the Bluetooth protocols that reside in the OSI layer. SDP is the Bluetooth Service Discovery Protocol, while the Management Entity [ME] is the entity that coordinates the procedures for initialization, configuration and connection management.

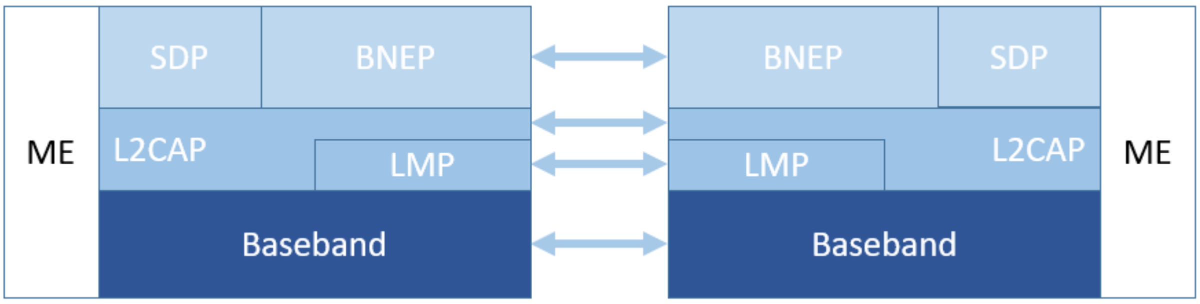


Figure 10 - Protocols and entities of the Bluetooth PAN profile

The required procedures that should be defined in the Bluetooth PAN profile are divided in two categories, following different steps based on whether the PANU wants to connect to a GN or the GN wants to connect to a PANU in order to finally create an ad-hoc network.

(i) Case of PANU initiating the connection

1. The first step is to find another Bluetooth device that is within radio range and is providing the GN service by using baseband inquiries and SDP searches.
2. If there is no existing Bluetooth connection, then the PANU requests a Bluetooth connection with the selected device providing the GN service. A master-slave switch will be required to complete the connection if the GN is in multi-user mode.
3. Once the connection is made, the PANU can create an L2CAP channel for BNEP and use the BNEP control commands to initialize the BNEP connection and setup filtering of different network packet types. The GN would store all network packet type filters in a Packet Filter Database, which would maintain a set of packet filters for each connection.
4. Ethernet traffic can now flow across the link. GNs will not provide networking services and therefore each of the PANUs will perform various tasks to operate without these services. The GN will forward all Ethernet packets to each of the connected PANUs.
5. At any time the PANU or the GN may terminate the connection(s).

(ii) Case of GN initiating the connection

This is only possible if the PANU advertises a PANU service record [Service Record].

1. The first step is to find another Bluetooth device that is within radio range and is providing the PANU service by using baseband inquiries and SDP searches.
2. If there is no existing Bluetooth connection, then the GN requests a Bluetooth connection with the selected device with the PANU service. No master-slave switch will be required.

3. Once the connection is made, the GN can create an L2CAP channel for BNEP. The PANU uses the BNEP control commands to initialize the BNEP connection and set up filtering of different network packet types. The GN would store all network packet type filters in a Packet Filter Database, which would maintain a set of packet filters for each connection.
4. Ethernet traffic can now flow across the link. GNs will not provide networking services and therefore each of the PANUs will perform various tasks to operate without these services. The GN will forward all Ethernet packets to each of the connected PANUs.
5. At any time the PANU or the GN may terminate the connection(s).

3.3.5. D2D Commands & Replies Specification

As soon as the devices have been paired and the connection has been established, the messages that will be exchanged within the context of the D2D protocol between the involved applications, should follow the process defined below, being also consistent to the provided rules, as described in the following tables. Figure 11 illustrates the commands and replies that must be exchanged in the case of the D2D protocol, between any involved applications.

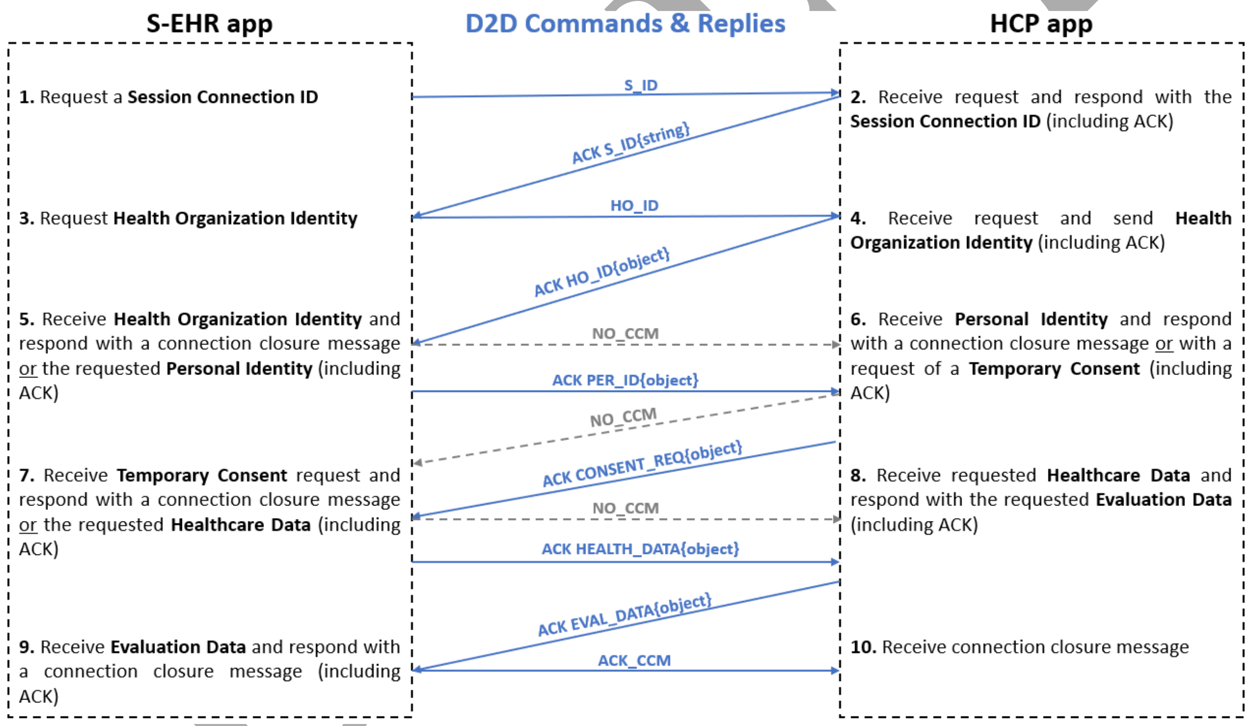

S-EHR app
D2D Commands & Replies
HCP app
1. Request a Session Connection ID
S_ID
2. Receive request and respond with the Session Connection ID (including ACK)
ACK S_ID{string}
3. Request Health Organization Identity
HO_ID
4. Receive request and send Health Organization Identity (including ACK)
ACK HO_ID{object}
5. Receive Health Organization Identity and respond with a connection closure message or the requested Personal Identity (including ACK)
NO_CCM
ACK PER_ID{object}
6. Receive Personal Identity and respond with a connection closure message or with a request of a Temporary Consent (including ACK)
NO_CCM
ACK CONSENT_REQ{object}
7. Receive Temporary Consent request and respond with a connection closure message or the requested Healthcare Data (including ACK)
NO_CCM
ACK HEALTH_DATA{object}
8. Receive requested Healthcare Data and respond with the requested Evaluation Data (including ACK)
ACK EVAL_DATA{object}
9. Receive Evaluation Data and respond with a connection closure message (including ACK)
ACK_CCM
10. Receive connection closure message

Figure 11 - D2D exchanged messages

A description of the steps of Figure 11 follows, accompanied with the rules that each command and reply should follow.

1. **S-EHR app**: The S-EHR app will request to the HCP app a Session Connection number through sending a specific type of request, following the format of: [S_ID]

Session Connection ID request Rules

Name	Session Connection ID request
Description	The identifier of the current Bluetooth connection session between the involved applications
Accepted Format	This message should follow the format of **S_ID**

2. **HCP app**: The HCP app will send to the S-EHR app an acknowledgement message, followed by the current's Session Connection number, following the format of: [ACK S_ID{string}]

Session Connection ID response Rules

Name	Session Connection ID response
Description	The identifier of the current Bluetooth connection session between the involved applications
Accepted Format	This message should follow the format of **ACK S_ID** followed by a String indicating the Session Connection ID

3. **S-EHR app**: The S-EHR app will request to the HCP app the Health Organization details, following the format of: [HO_ID]

Health Organization ID request Rules

Name	Health Organization ID request
Description	The request of the Health Organization identity
Accepted Format	This message should follow the format of **HO_ID**

4. **HCP app**: The HCP app will send to the S-EHR app an acknowledgement message, followed by the object identifying the Health Organization identity, including information about the Healthcare Organization, following the format of: [ACK HO_ID{object}]

Health Organization ID response Rules

Name	Health Organization ID response
Description	The acknowledgment message and the message that corresponds to the Healthcare Organization identity
Accepted Format	This message should follow the format of **ACK HO_ID** followed by a HL7 FHIR Resource object (i.e. Practitioner Resource object) that specifies the identity of the Healthcare Organization. The included details of the object are specified in the Interoperability Profiles of D2.7 [D2.7].

5. **S-EHR app**: The S-EHR app will send to the HCP app an acknowledgement message, followed by the object identifying the Personal Identity of the S-EHR app owner, following the format of: [ACK PER_ID{object}]. In the case that the Health Organization identity will not be approved, the message that will be sent will be a one way message, informing about the connection closure, following the format of: [NO_CCM]

Personal ID response Rules

Name	Personal ID response
Description	The acknowledgment message and the message that corresponds to the Citizen's Personal identity.
Accepted Format	This message should follow the format of **ACK PER_ID** followed by a HL7 FHIR object (i.e. Patient Resource object). That specifies the identity of the owner of the S-EHR app. The included details of the object are specified in the Interoperability Profiles of D2.7 [D2.7]. In the case that a connection closure message should be sent, this should follow the format of **NO_CMM** indicating the connection closure.

6. **HCP app**: The HCP app will send to the S-EHR app an acknowledgement message, followed by the object of the Temporary Consent towards the S-EHR app owner, following the format of: [ACK CONSENT_REQ{object}]. In the case that the citizen's personal identity will not be approved, the message that will be sent will be a one way message, informing about the connection closure, following the format of: [NO_CCM]

Temporary Consent request Rules

Name	Temporary Consent request
Description	The acknowledgment message and the message that corresponds to the Consent Request
Accepted Format	This message should follow the format of **ACK CONSENT_RE**Q followed by a HL7 FHIR object (i.e. Consent Resource object) that specifies the requested consent. The included details of the object are specified in the Interoperability Profiles of D2.7 [D2.7]. In the case that a connection closure message should be sent, this should follow the format of **NO_CMM** indicating the connection closure.

7. **S-EHR app**: The S-EHR app will send to the HCP app an acknowledgement message, followed by the object identifying the Healthcare data to be provided by the S-EHR app owner, following the format of: [ACK HEALTH_DATA{object}]. In the case that the Consent Request will not be approved, the message that will be sent will be a one way message, informing about the connection closure, following the format of: [NO_CCM]

Healthcare Data response Rules

Name	Healthcare Data response
Description	The acknowledgment message and the message that corresponds to the citizen's requested Healthcare Data
Accepted Format	This message should follow the format of **ACK HEALTH_DATA** followed by a HL7 FHIR object (i.e. HealthRecord Resource object) that specifies the Citizen's requested Healthcare Data. The included details of the object are specified in the Interoperability Profiles of D2.7 [D2.7]. In the case that a connection closure message should be sent, this should follow the format of **NO_CMM** indicating the connection closure.

8. **HCP app**: The HCP app will send to the S-EHR app an acknowledgement message, followed by the object of the Evaluation Data towards the S-EHR app owner, following the format of: [ACK EVAL_DATA{object}]

Evaluation Data response Rules

Name	Evaluation Data response
Description	The acknowledgment message and the object that corresponds to the citizen's Evaluation Data
Accepted Format	This message should follow the format of **ACK EVAL_DATA** followed by a HL7 FHIR object (i.e. HealthRecord Resource object) that specifies the evaluation data. The included details of the object are specified in the Interoperability Profiles of D2.7 [D2.7].

9. **S-EHR app**: The S-EHR app will send to the HCP app a one way message, informing about the successful connection closure (upon receiving the evaluation data), following the format of: [ACK_CCM]

Connection Closure Message response Rules

Name	Connection Closure Message response
Description	The message that informs about the success of the reception of the evaluation data, and the connection closure between the involved applications
Accepted Format	This message should follow the format of ACK_CMM indicating the Bluetooth connection closure.

4. R2D PROTOCOLS: REMOTE HEALTH DATA EXCHANGE

As seen in the introduction chapter, the most characteristic aspect of InteropEHRate project is the fact that the EHR of a citizen resides on his / her mobile device. Citizen's (health) data transfer is realized by using the protocols described in this deliverable: the D2D protocol for short range transmission, and the R2D and R2D Cloud protocols for remote transmission over the internet. In the InteropEHRate standard architecture (i.e. the European architecture for health data exchange proposed by the project), data flows from an EHR system of Country A (the country of the citizen) to the Health Organization Information System of an HCP operating in Country B, with the mediation of the citizen using the supporting software applications.

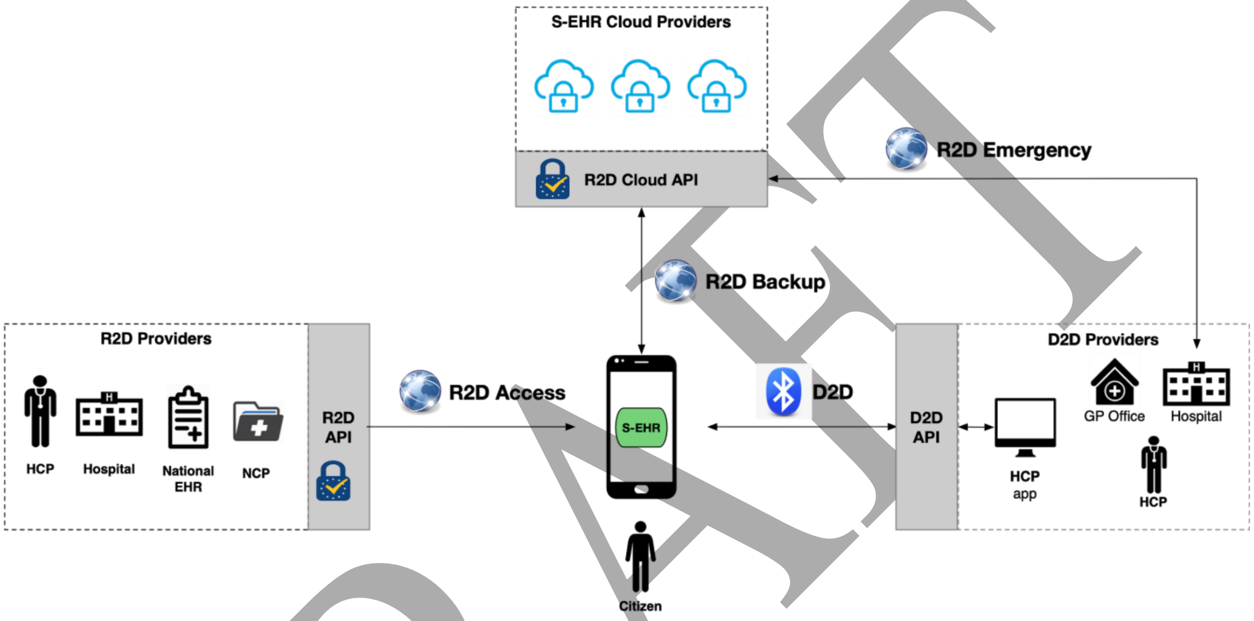


Figure 12 - R2D and R2D Cloud protocols scope

The R2D and R2D Cloud protocols are both internet based protocols but with different roles and purposes. The R2D protocol, defines the set of operations used for enabling (in a standard way) the download of health data between an application (S-EHR app) acting on behalf of a patient and a clinical record system. The clinical record system may be supporting a clinical care provider (e.g. a hospital, or a general practitioner), or a health data exchange, including a national health record system. The R2D allows a S-EHR to download health data of a citizen in a standard way from heterogeneous sources of data even pertaining to different countries. Differently from the R2D, the R2D Cloud protocol is mainly used to allow the citizen to have a private copy of their EHR in a remote repository (S-EHR Cloud), the purpose of the remote copy is both for backup reasons and to allow, in case of emergency, access to patient's data by an HCP. Requirements for R2D and R2D Cloud protocols are different, while for the first the main requirements is the ease of adoption for the latter the most relevant requirements is to ensure security of personal data stored remotely.

According to the objectives defined by the InteropEHRate consortium for the first and the second year of the project, the objectives for R2D and R2D Cloud protocols are: i) acquisition of medical data from a local or National EHR (to import them in the citizen's device), ii) transfer of medical data to the S-EHR Cloud. The introduction of this second objective represents the most important update from the previous version of this deliverable.

This section of the document is structured in four sub sections:

- R2D Related Works: a state of the art section regarding relevant projects in the field of Cross Border Health Data Exchange related to R2D goals.
- R2D Conceptual Description: a conceptual description of R2D operations and transactions, showing functional requirements of R2D.
- R2D Technical Specifications: a technical chapter providing specifications of the R2D protocol.
- R2D Cloud Conceptual Description: a conceptual description of R2D operations and transactions, showing functional requirements of R2D Cloud.

4.1. R2D Related Works

This section describes four relevant projects or initiatives that are related to one of the following thematic areas: i) FHIR, ii) interoperability in eHealth domain and iii) cross border health data exchange.

4.1.1. eHDSI

Importing health data in a standard way from EHRs of several European countries is a very challenging subject. At the moment each Member State has its own EHR, based on proprietary APIs, proprietary data models and proprietary data representation. These EHRs have been designed in order to satisfy only requirements coming from stakeholders of the Member State itself and not to be interoperable with each other. Many steps have been performed by the EU in order to foster interoperability between European eHealth systems, especially through the activities of the eHealth Network (established under Article 14 of Directive 2011/24/EU of the European Parliament and of the Council) and the CEF Programme.

The most important project regarding cross border health data exchange financed by the EU is the European eHealth Digital Services Infrastructure (eHDSI), whose objectives are the initial deployment and operation of services for cross-border health data exchange under the CEF. The architecture of eHDSI (Figure 13) is based on a closed and trusted federation of NCP one for each Member State named eHDSI Circle of Trust; an NCP is a software component representing a Member State's EHR. An NCP provides a reduced-but-common API designed for allowing EHRs of EU countries to InteropEHRate in order to exchange health data.

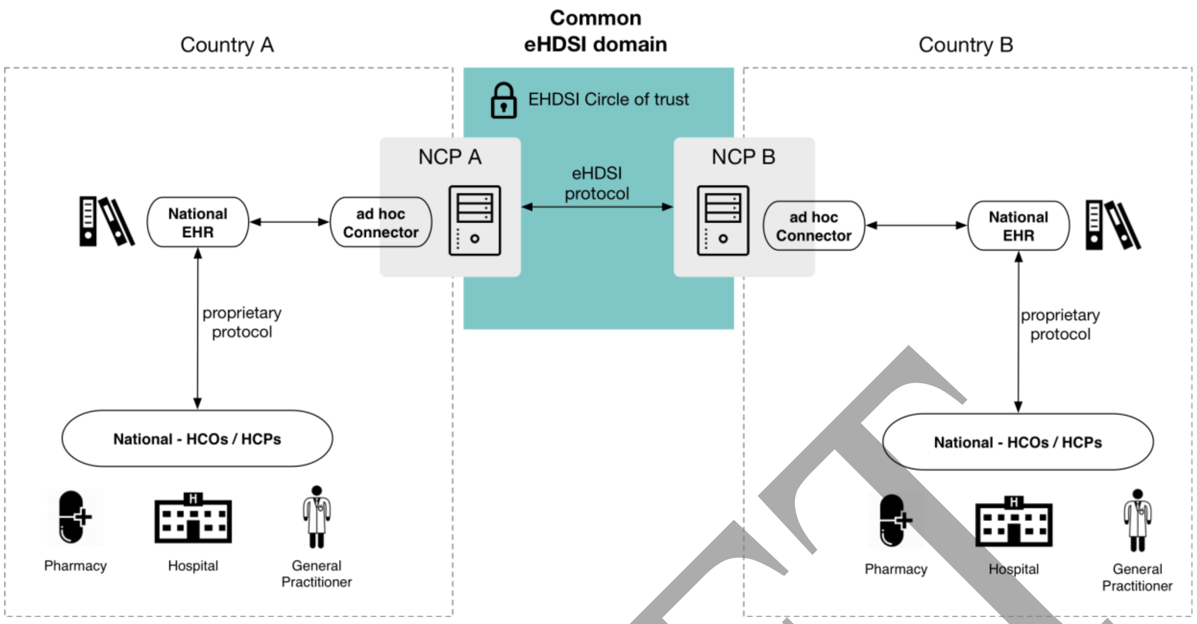


Figure 13 - eHDSI system architecture

The above figure shows the eHDSI System Architecture [EHDSI SYS SPECS], stressing the role of the NCPs federation as a common integration layer between the different EHRs of Member States. An NCP has the following responsibilities:

- forwarding requests coming from its country to other NCPs;
- handling incoming requests coming from other NCPs;
- converting data from eHDSI data representation to owner country data representation.

Differently from InteropEHRate, in eHDSI a citizen's EHR, despite the ownership of the citizen itself, resides in the IT National Infrastructure. In this context, an HCP of Country A, in order to request health data of a citizen of Country B (in that specific moment in Country A in front of the HCP), needs to access to National Infrastructure of his country (Country A) and use the provided cross border health data exchange functionalities to submit requests to Country B via NCP (requests are delegated to NCP of Country A, that forwards them to NCP of Country B). All the transactions in eHDSI follow this schema, with no exception. The HCP (requestor of health data of citizens) interacts only with his National Infrastructure, and in the background the two involved national systems exchange data using eHDSI as a common integration layer. An important aspect to underline is that in eHDSI transactions, the citizen does not have an active role. According to [EUCBEHR REC] (EU Commission Recommendation of the sixth of February 2019 on a European Electronic Health Record exchange format), the baseline for the exchange contains the following health data:

- Patient Summary;
- ePrescription / eDispensation;
- Laboratory results;
- Medical imaging and reports;
- Hospital discharge reports.

R2D and eHDSI share many objectives (mainly the exchange of health data in a standard way), however, they show also some relevant differences:

- eHDSI is a system available only to HCOs and HCP, it has not been designed considering the citizen as a primary actor. In eHDSI, a citizen cannot submit requests to his reference NCP (NCP of his country), because NCPs are configured to receive requests only from other NCPs (closed and trusted federation). Differently, in InteropEHRate the citizen is the actor at the centre of the platform, he periodically downloads his data from the National EHR to his smart mobile device, and he handles directly his data when in front of an HCP.
- eHDSI uses a standard representation of data (HL7/CDA), but it does not adopt any of the existing eHealth standard API, like for instance the API proposed by OpenEHR project [OPENEHR]. Differently, R2D will be based upon a reduced version of FHIR, it will support FHIR API language and FHIR data model and data representation. It means that every application already interacting with a FHIR compliant repository, may be easily converted to use R2D.

This last point is the most important for a possible future convergence between eHDSI and R2D. Also the European Commission outlined this issue in [EUCBEHR ANNEX] (ANNEX to the Commission Recommendation on a European Electronic Health Record exchange format): “The refinement of the exchange format should consider the possibility offered by resource driven information models (such as Health Level Seven Fast Healthcare Interoperability Resources (HL7 FHIR©)”.

The following figure (Figure 14) shows an hypothetical future scenario describing how eHDSI and InteropEHRate could coexist:

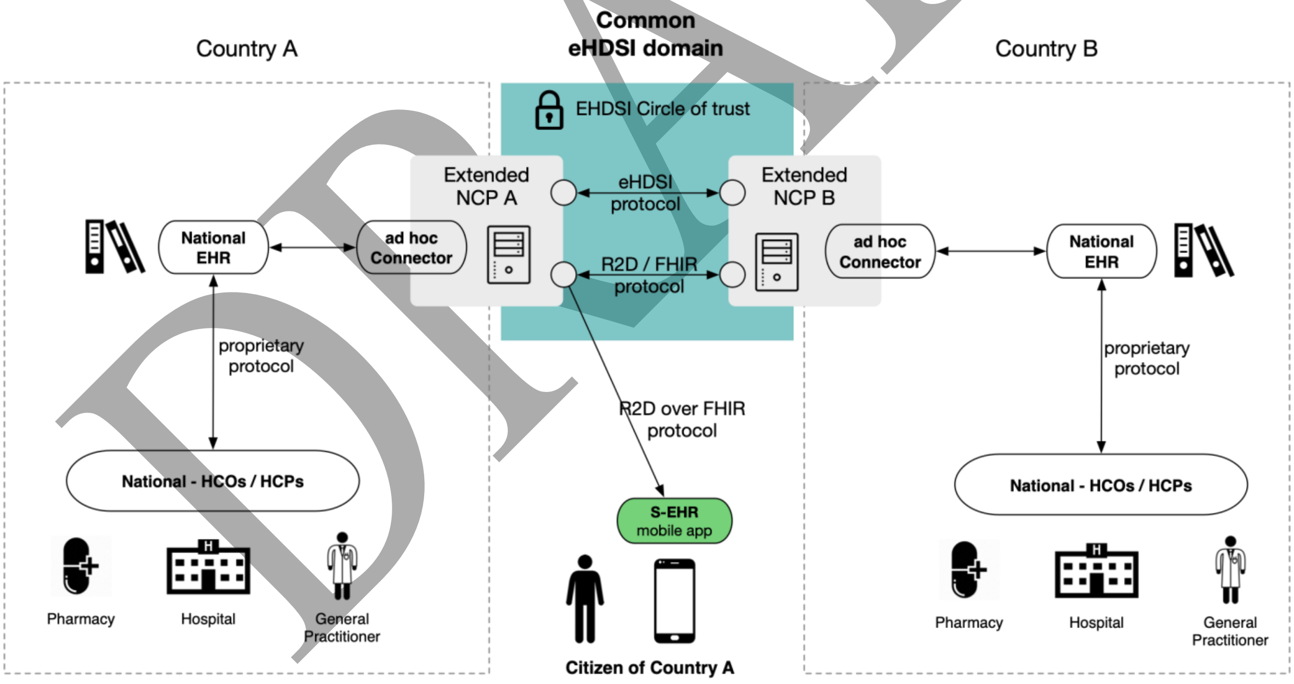


Figure 14 - Common eHDSI domain

The hypothesis behind this figure is the adoption of FHIR by eHDSI (that would have two interfaces), in this scenario, it would be very easy for applications developed using R2D to connect to an NCP in order to download citizen’s medical data. In this vision, the infrastructure of eHDSI (the only one that will allow apps to interact in a standard way with European’s EHRs) will be paired with a largely diffused and supported standard interoperability protocol and made available also to citizens: eHDSI infrastructure, FHIR interaction. The implementation of this architecture would represent a great step forward to cross border health data exchange and citizens empowerment. Results of the European International Patient Summary Project [IPSP] are a concrete step forward to make possible FHIR adoption in eHDSI.

4.1.2. International Patient Summary Project

Another important European project (2015-2018) that certifies the growing diffusion of FHIR is the International Patient Summary Project [IPSP]. This joint project, participated by HL7 and European Committee for Standardization (CEN), had as objective the world-wide specifications (in FHIR) of an International Patient Summary to become an European Standard, following guidelines adopted by the European eHealth Network (eHN). The project has followed two main development lines: one to define the specifications and the other one to obtain the status of standard by CEN. The project started by the initial proposition of Patient Summary dataset proposed by eHN in 2013, then produced the official IPS specifications adopted by eHN and published in 2019 (The IPS specifications are primarily designed to support cross-border emergency and unplanned care).

IPS project had several official contacts with eHDSI, first of all eHDSI was tasked with an experimental project across member states using the eHN PS guidelines as part of their work. Furthermore, a new European eHAction roadmap (2018-2020) seeks to leverage the work of the IPS Project going forward so as to support the eHDSI initiative.

4.1.3. Argonaut Project

"The Argonaut Project is a private sector initiative to advance industry adoption of modern, open interoperability standards. The purpose of the Argonaut Project is to rapidly develop a first-generation FHIR-based API and Core Data Services specification to enable expanded information sharing for electronic health records and other health information technology based on Internet standards and architectural patterns and styles" [ARGONAUT].

It is important to underline that the Argonaut Project is not an organization for the definition of new standards, objectives of the project are to accelerate and promote the adoption of FHIR and OAuth in eHealth care provisioning. The Argonaut Project consortium is: i) composed by leading technology vendors (Accenture, Apple) and private provider organizations (Mayo Clinic, Beth Israel Deaconess Medical Center), ii) financed by private sector, iii) collaborating with the most important FHIR official initiatives (SMART-on-FHIR and the Health Systems Platform Consortium) and organizations (FHIR Foundation).

One of the most important results of the Argonaut project is the adoption of FHIR by Apple in the Health, this app allows citizens to aggregate their health data, retrieved by multiple institutions, in their (Apple) mobile device. The interoperability standard adopted is FHIR, and this is an excerpt from the Apple web site:

"The connection between your electronic health record (EHR) and a user's Health app utilizes FHIR (Fast Healthcare Interoperability Resources) standard APIs as defined by the Argonaut Project. Supported data types are allergies, conditions, immunizations, lab results, medications, procedures, and vitals" [APPLE HEALTH].

4.1.4. IHE Project

IHE is a joint initiative, globally extended, developed with the aim of creating one methodology, shared and effective, to make systems and IT entities of the health sector interact with each other. The IHE experience saw the light in 1998 in the United States in response to growing problems of interoperability in the field of radiology. The Fatherhood of the organization is attributable to two user associations: the RSNA (Radiological Society of North America) and HIMSS (Healthcare Information and Management Systems Society). After a few years IHE also took hold in European countries and, at present, the structure of the initiative is divided into three wide Regions of interest (North America, Europe, Asia) in turn organized by nations. Although its committee is made up of users (and manufacturers) of these systems, the IHE organization was born as non-profit, the main objective of the initiative is, in fact, to promote the culture of integration, through an accurate definition of clinical needs and a combined use of the most important standards. The ultimate goal of this strategy is, in essence, to speed up and make health integration more efficient, and clinical practice in general.

IHE produces three main outputs: i) eHealth related use cases, ii) definition of the IHE Profiles iii) definition of the Technical Frameworks related to a profile. An IHE profile is a description of the context and of the architecture of the technical solution for implementing a use case, while the Technical Framework provides all the concrete interoperability specification: API exposed by each component involved and the type and structure of exchanged data.

IHE consortium, collaborating with HL7, is actually defining the IHE profile for accessing the International Patient Summary [IHE IPS] in order to execute the second step of the IPS standardization that regards how to access and retrieve in a standard way an instance of Patient Summary from an eHealth System. The IHE IPS Profile will expand the specification to cover four scenarios: unscheduled care, scheduled care, cross-border care and within-border care.

While the IHE IPS project is focused only on IPS access, it is worth mentioning a new IHE/HL7 project at a very initial stage (November 2019), named International Patient Access [IHE IPA] focused on the definition of an IHE profile for the standardized access to “patient records anywhere in the world”. The objective of the project is to identify a minimal but significant set of access methods and rules that are true everywhere in the world.

This International Patient Access specification describes how to access patient records anywhere in the world. It provides a very minimal set of access methods and rules about the medical contents that are true everywhere. The following list shows the set of FHIR resources identified by the IPA project as target resources:

- Basic patient details
- Problems / Conditions
- Encounters
- Current and past medications
- Immunization history
- Allergies and intolerances
- Diagnostic reports (e.g. labs, imaging)
- Vital signs and other clinical observations
- Patient forms / questionnaires

- Clinical notes & other patient documents
- Care plans and Care teams

R2D and IPA have so many things in common that it is difficult to think that there is no meeting point between the two in the future.

4.2. R2D Conceptual Description

Before going into technical details of R2D, the protocol is firstly described in a conceptual way, to simplify the understanding of how R2D transactions work and how they are structured (abstract R2D operations represent the functional requirements of R2D). Also, the non-functional requirement, expressed by the InteropEHRate consortium, to define R2D based on currently existing health standards makes it useful to have a conceptual description of the protocol that would be subsequently implemented with the concrete reference protocol. Moreover, the requirements expressed here allow the implementation of R2D (or only of specific functionalities of R2D) with more than one reference health standard. This particular aspect is described in detail in the ANNEX in a specific section describing how eHDSI may be mapped to some R2D operations.

4.2.1. Conceptual R2D API

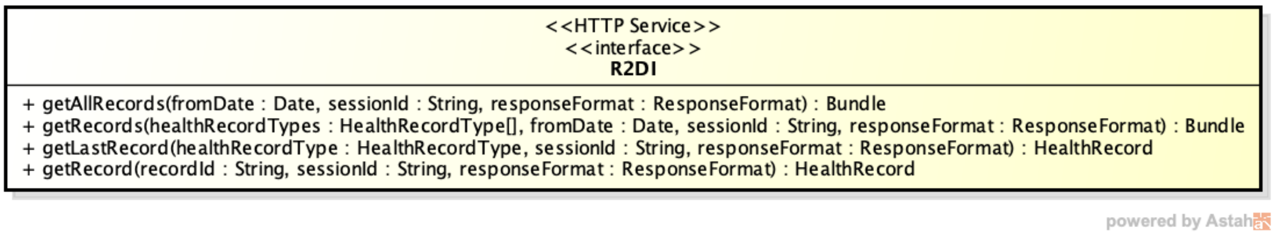
The Conceptual R2D defines the basic operations that a protocol for the exchange of medical data should support. In accordance with the InteropEHRate objectives of Year 1 and Year 2, the protocol defines operations for acquiring medical data from a remote EHR source.

Although R2D is focused on the exchange of medical data, it requires some security operations to authenticate a citizen (see report [D3.1] for further details). These security operations are the only operations that do not exchange medical data, and must be considered as preliminary operations invoked to enable a secure exchange of medical data. Authenticating to the NCP, is the only way to gain access to the core R2D functionalities for exchange of medical data. R2D has two main states:

- AUTHENTICATED: is the state that allows the use of the operations for the exchange of medical data.
- NOT_AUTHENTICATED: is the state that does not allow the use of the operations for the exchange of medical data, but allows only the use of the operations for the authentication of a citizen.

Once a citizen is authenticated and the protocol is in the AUTHENTICATED state, all the R2D operations for the exchange of medical data can be used, without any restrictions and without the need of following restrictions in the sequence of operations (indeed, a part of the security operations that changes the state of the communication, the rest of the R2D protocol is just a restful a service).

The following figure (Figure 15) shows an UML class diagrams representing the main interface of R2D named R2DI and implemented by the HTTP service providing access to the underlying EHR (the data referenced by this interface have been already described earlier in this document in the common data model section):



```
+ getAllRecords(fromDate : Date, sessionId : String, responseFormat : ResponseFormat) : Bundle
+ getRecords(healthRecordTypes : HealthRecordType[], fromDate : Date, sessionId : String, responseFormat : ResponseFormat) : Bundle
+ getLastRecord(healthRecordType : HealthRecordType, sessionId : String, responseFormat : ResponseFormat) : HealthRecord
+ getRecord(recordId : String, sessionId : String, responseFormat : ResponseFormat) : HealthRecord
```

Figure 15 - R2D interface operations

The main requirement that drove the design of the R2DI interface was the idea of allowing the download of a great amount of data invoking just one R2D operation. Most likely, R2D will be used for a massive initial download of citizen's data and for successive downloads of small portions of data needed to maintain the mobile device synchronized with the national EHR. Implementers of R2D MUST be aware that such operations will be performed by device with limited amount of memory, limited computation capability, limited storage space and that every operation may be interrupted by higher priority events happening to the mobile device. Support for pagination of large amounts of data MUST be taken into account and must be realized using lazy techniques without requiring the modification of R2DI interface with specific methods for pagination.

Always in order to reduce the number of invocations to download large amount of data, R2DI massive retrieval operations have not been partitioned by kind of data (i.e. getMedicalImages(), getLaboratoryResults(), etc. etc.), but they have been conceived as they would logically perform the SQL UNION operation, allowing to retrieve more than one kind of health data in just one single invocation. The next sections provide a complete description for each of the operations of interface R2DI.

Operation getAllRecords

Name	getAllRecords
Description	This is the main method of R2DI. It allows a client to request all kinds of health data (Patient Summary, Laboratory Reports, Medical Images, Prescriptions, Dispensation, Discharge Report) of a citizen starting from a certain date. Periodical execution of the method allows synchronization between mobile devices and National EHR.
Arguments	• Date fromDate: a date indicating the day after which the requested health data must have been produced. • String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method. • ResponseFormat responseFormat: one of the predefined ResponseFormat enumeration values identifying the output format of the requested health data.
Return Value	An instance of Bundle allowing access to retrieved data.

Exceptions	• Security exceptions related to the validation of the session. • Security exceptions related to the unauthorized access to health data. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation getRecords

Name	getRecords
Description	This is the main method of R2DI. It allows a client to request all kinds of health data (Patient Summary, Laboratory Reports, Medical Images, Prescriptions, Discharge Report) of a citizen starting from a certain date. Periodical execution of the method allows synchronization between mobile devices and National EHR. Differently from the getAllRecords() method, this method allows a client to request only one or more specific kinds of health data starting from a certain date. The requested kinds of health data to be retrieved are passed as arguments.
Arguments	• HealthRecordType[] healtRecordTypes: an array of predefined values containing the types of health data requested by the client. • Date fromDate: a date indicating the day after which the requested health data must have been produced. • String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method. • ResponseFormat responseFormat: one of the predefined ResponseFormat enumeration's value identifying the output format of the requested health data.
Return Value	An instance of Bundle allowing access to retrieved data.
Exceptions	• Security exceptions related to the validation of the session. • Security exceptions related to the unauthorized access to health data. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation getLastRecord

Name	getLastRecord
Description	This method allows a client to request the most recent instance of a certain type of health data (of the citizen). The kind of health data is passed as an argument to the method.
Arguments	● HealthRecordType healtRecordType: an instance of one of the predefined values of HealthRecordType indicating the type of health data requested by the client. ● String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method. ● ResponseFormat responseFormat: one of the predefined ResponseFormat enumeration values identifying the output format of the requested health data.
Return Value	An instance of HealthRecord containing the most recent instance of a specific health data type.
Exceptions	● Security exceptions related to the validation of the session. ● Security exceptions related to the unauthorized access to health data. ● Network exceptions related to failure during remote communication.
Preconditions	● The citizen has successfully executed the authentication to the InteropEHRate infrastructure. ● The session is still valid.

Operation getRecord

Name	getRecord
Description	This method allows a client to request a specific instance of health data identified by its unique id.
Arguments	● String recordId: a valid id of a health data. ● String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method. ● ResponseFormat responseFormat: one of the predefined ResponseFormat enumeration values identifying the output format of the requested health data.
Return Value	An instance of the requested Resource.

Exceptions	● Security exceptions related to the validation of the session. ● Security exceptions related to the unauthorized access to health data. ● Network exceptions related to failure during remote communication.
Preconditions	● The citizen has successfully executed the authentication to the InteropEHRate infrastructure. ● The session is still valid.

4.2.2. Involved Applications

This section provides a brief description of the applications involved in the use of R2D: the S-EHR app and the EHR Provider (Extended NCP, National or Local EHR).

S-EHR Application

The S-EHR app is also described in Section 3 of the current document. In a R2D transaction, it represents the client submitting requests.

EHR Provider

The EHR Provider is the entity that concretely implements the server side of R2D and replies to requests submitted by S-EHR applications. EHR providers may be:

- Local or National EHR providing R2D interface.
- An extended NCP, that is an eHDSI NCP extended to support the authentication of a Citizen (and not only of HCPs), and the submission of requests also by apps acting on behalf of the citizen itself (and not only by other NCPs).

4.2.3. R2D Sample Interaction

The following UML sequence diagram (Figure 16) shows how the R2D API, described in the previous section, may be used to retrieve health data of a citizen. The R2D is not a rigid protocol and the same results may be obtained using different sequences of methods invocation; the objective of this sequence diagram is to show one of the possible sequences of invocations in order to perform a search and browse all returned results.

The sequence diagram shows the usage of the following methods: getLastRecord(), getRecords(), getRecord(). In the diagram, the invocation of these methods are highlighted in red. The actors of this sequence diagram are the following:

- Citizen: a generic citizen interested in accessing his health data. The citizen is an active actor, he is the one that starts the import of data and that defines what kind of data must be imported.
- S-EHR App: the app used by the citizen to send requests to the NCP Node. The app represents a client of the R2D protocol.
- NCP Node: A National Contact Point as described in the eHDSI platform, but also implementing the R2D protocol (it represents the server side of the protocol).

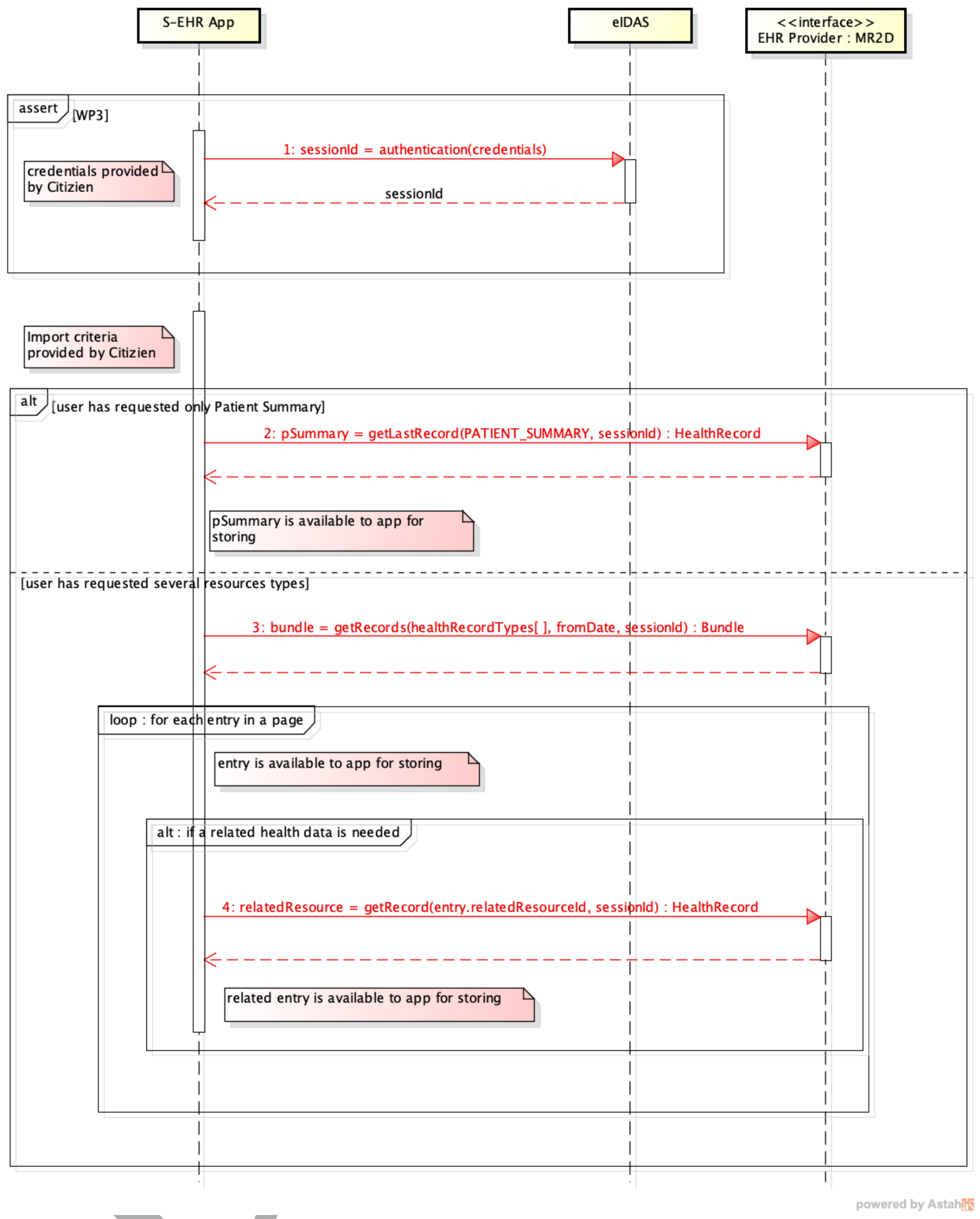

S-EHR App
eIDAS
<<interface>> EHR Provider : MR2D
assert [WP3]
credentials provided by Citizien
1: sessionId = authentication(credentials)
sessionId
Import criteria provided by Citizien
alt [user has requested only Patient Summary]
2: pSummary = getLastRecord(PATIENT_SUMMARY, sessionId) : HealthRecord
pSummary is available to app for storing
[user has requested several resources types]
3: bundle = getRecords(healthRecordTypes[ ], fromDate, sessionId) : Bundle
loop : for each entry in a page
entry is available to app for storing
alt : if a related health data is needed
4: relatedResource = getRecord(entry.relatedResourceId, sessionId) : HealthRecord
related entry is available to app for storing
powered by Astah

Figure 16 - R2D protocol interactions

Following a detailed description of the sequence diagram:

- **Step 1 - authentication**: this is a preliminary step that moves the R2D state from NOT_AUTHENTICATED to AUTHENTICATED and allows the app citizen to gain access to all the services of the R2D protocol. A detailed description of this specific interaction is provided in the deliverables about security of work package 3.
- **Step 2 - import**: this is the step started by the citizen that triggers the import of health data from the remote repository. This step is composed by the following sub steps and its execution foresees the provisioning of some parameters by the citizen, in order to drive the data download. The parameters are the following: i) healthRecordTypes[]: an array of predefined values containing the

types of health data requested by the client; ii) fromDate: a date indicating the day after which the requested health data must have been produced.

- **Step 2.1 - getLastResource**: optional step executed only if the citizen has requested only to download his / her patient summary. In this case the client invokes the protocol method named getLastResource(), providing as input parameter the enum value PATIENT_SUMMARY. The method returns to the client the most recent instance of HealthRecord of type HealthRecordType.PATIENT_SUMMARY. The patient summary is now available to the client for storing.
- **Step 2.2 - getRecords**: optional step executed only if the citizen has requested more than one type of HealthRecord. In this case the client invokes the protocol method named getRecords(), passing as input parameters the array of types provided by the citizen and the reference date. This method returns an instance of Bundle containing only the first of the overall pages that compose the entire result.
- **Step 2.3 - getRecord**: at this point, the client has obtained the first page of the overall results and starts looping the items of the current page (entries of the bundle) processing each one of them. If during this processing, the client needs to download a health data related to the one under processing, the client will invoke the protocol method named getRecord(), passing the id of the related resource as input. After this invocation, the client is able to process both the primary health data and its related resource.

4.3. R2D Technical Specifications

So far, the R2D has been described only from a conceptual point of view, while this section provides concrete technical specifications. The R2D is an internet based protocol, abstract operations described in the previous section are converted into software services exposed on the HTTP as RESTful services. A strong requirement from the InteropEHRate consortium, is to define R2D using already existing eHealth standards thus avoiding the definition of a completely new protocol: the reference standard chosen is FHIR [HL7 FHIR]. FHIR is a well accepted standard supported by HL7, its importance and adoption is increasing constantly (as described in the state of the art section).

The objective of this section is to define the technical specifications of the API exposed by the R2D to implement the operations listed in the R2D conceptual protocol, mapping them to FHIR RESTful API [FHIR API SPEC] services, defining constraint and usage of search parameters. R2D does not provide any specifications about the concrete data model but will reference data structures defined in deliverable [D2.7]. R2D specifications and "FHIR profile for EHR interoperability" provide all the technical information needed to integrate software applications with the InteropEHRate platform, they define respectively: i) how two systems interact to exchange health data, ii) what kind of health can be exchanged between the two.

At the moment due to some mismatch between the conceptual R2D and the FHIR API, the Conceptual R2D cannot be translated into FHIR, but most of all of its elementary operations can be. Translations of these elementary operations represent the core part of the following specifications, the following table contains the set of elementary operations that compose the R2D:

Operation	Internal Code
Search of structured Patient Summary	R2D_SRC_STR_PAT_SUM
Search of unstructured Patient Summary	R2D_SRC_UNSTR_PAT_SUM
Retrieval of most recent structured Patient Summary	R2D_LST_STR_PAT_SUM
Retrieval of most recent unstructured Patient Summary	R2D_LST_UNSTR_PAT_SUM
Search of structured Laboratory Result	R2D_SRC_STR_LAB_RES
Search of unstructured Laboratory Result	R2D_SRC_UNSTR_LAB_RES
Retrieval of most recent structured Laboratory Result	R2D_LST_STR_LAB_RES
Retrieval of most recent unstructured Laboratory Result	R2D_LST_UNSTR_LAB_RES
Search of structured Medical Image	R2D_SRC_STR_MED_IMG
Search of unstructured Medical Image	R2D_SRC_UNSTR_MED_IMG
Retrieval of most recent structured Medical Image	R2D_LST_STR_MED_IMG
Retrieval of most recent unstructured Medical Image	R2D_LST_UNSTR_MED_IMG
Search of structured Prescription	R2D_SRC_STR_PRE
Search of unstructured Prescription	R2D_SRC_UNSTR_PRE
Retrieval of most recent structured Prescription	R2D_LST_STR_PRE
Retrieval of most recent unstructured Prescription	R2D_LST_UNSTR_PRE
Direct access by ID to a specific resource	R2D_GET_RES

For each operation, listed in previous table, are specified the following features:

- the URL to be invoked;
- the type of FHIR resource that is the subject of the request;
- the specific version and FHIR profile of the resource that is the subject of the request (returned data);
- The allowed parameters that may be added to the URL, and when needed, the constraints defined over parameter values;
- The set of possible HTTP return codes.

R2D conceptual operations manage the following types of health data: Patient Summary, Laboratory Report, Medical Image, Prescription. In order to translate conceptual operations to specific FHIR requests, it is fundamental to define the correspondence between such conceptual data types and concrete FHIR resources. Such mapping (focus of deliverable [D2.7]) is briefly reported in the following table to facilitate the understanding of R2D specifications (a complete reading of deliverable D2.7 [D2.7] is essential for a complete comprehension of R2D and D2D specifications):

Conceptual Type	FHIR Resource	FHIR Version	FHIR Profile
Patient Summary	Bundle	DSTU3	International Patient Summary profile
Laboratory Result	DiagnosticReport and Observation	R4	InteropEHRateLaboratoryRequest profile
Medical Image	DiagnosticReport and ImagingResource	R4	InteropEHRateMedicalImages profile
Prescription	MedicationOrder	R4	InteropEHRatePrescription profile
Discharge Report	Not yet mapped	NA	NA

R2D over FHIR has been specified applying restrictions and constraints to the standard HL7 FHIR RESTFul API [FHIR API SPEC], R2D is a read-only protocol that operates on a specific subset of the FHIR resource, all operations requested are constrained to the authenticated citizen (they do not require filtering criteria for patient) and are used mainly for periodic import of citizen's health data from an EHR to the mobile device. Such kind of operations do not require all the expression capabilities provided by the [FHIR API SPEC], so to facilitate the adoption of R2D by EHR providers, R2D has a reduced number of filtering criteria while querying for resources.

The operations defined in conceptual R2D, are based on the idea of requesting different health data types of the citizen with just one operation of the protocol, grouping results. Such kinds of operations do not fit well with the way RESTful FHIR API has been structured because, in accordance with guidelines and best practices of the RESTful architectures, they are resource-based, so they work at best only at resource level. Despite this, the designers of the FHIR RESTful API defined also a search operation that works over multiple resources (https://www.hl7.org/fhir/http.html#search) acting similarly to the SQL UNION operator, this operation has the following signature:

```
http://<baseurl>/?_type=[Res1,Res2,ResN]&[params]{&_format=[mime-type]}
```

Obviously, this operation has a strong restriction regarding the allowed set of parameters that may be provided by the client, because, differently from the SQL UNION, the WHERE condition is expressed only once, it is in common with all listed resources and can include only parameters (search criteria) that are in common with all listed resources (https://www.hl7.org/fhir/searchparameter-registry.html#common).

For what concern, our specific case, the limitation of the `_type` operator does not allow to execute the following operations as they have been designed in conceptual R2D:

1. Patient Summary retrieval: common parameters are totally missing search criteria about Bundle;

2. Laboratory Results and Medical Images retrieval: common parameters are missing search criteria over the "category" attribute of DiagnosticReport.

However, FHIR provides several extensions mechanisms and the update of the `_type` operator will be investigated in the next years of the project by the InteropEHRate consortium, collaborating with the FHIR community to see if it would be possible to extend the standard. At the moment, the Conceptual R2D cannot be translated into FHIR as it is, but all of its elementary operations can be; the translation to FHIR of these elementary operations represents the core part of the following specifications. In case it would not be possible to solve `_type` operator issue, the grouping functionality will be delegated to the client libraries defined in deliverable [D4.4].

The following table shows what R2D operations are supported by R2D over FHIR implementation:

Operation	Internal Code	Supported
Search of structured Patient Summary	R2D_SRC_STR_PAT_SUM	YES
Search of unstructured Patient Summary	R2D_SRC_UNSTR_PAT_SUM	YES
Retrieval of most recent structured Patient Summary	R2D_LST_STR_PAT_SUM	YES
Retrieval of most recent unstructured Patient Summary	R2D_LST_UNSTR_PAT_SUM	YES
Search of structured Laboratory Result	R2D_SRC_STR_LAB_RES	YES
Search of unstructured Laboratory Result	R2D_SRC_UNSTR_LAB_RES	YES
Retrieval of most recent structured Laboratory Result	R2D_LST_STR_LAB_RES	YES
Retrieval of most recent unstructured Laboratory Result	R2D_LST_UNSTR_LAB_RES	YES
Search of structured Medical Image	R2D_SRC_STR_MED_IMG	YES
Search of unstructured Medical Image	R2D_SRC_UNSTR_MED_IMG	YES
Retrieval of most recent structured Medical Image	R2D_LST_STR_MED_IMG	YES
Retrieval of most recent unstructured Medical Image	R2D_LST_UNSTR_MED_IMG	YES
Search of structured Prescription	R2D_SRC_STR_PRE	YES
Search of unstructured Prescription	R2D_SRC_UNSTR_PRE	YES
Retrieval of most recent structured Prescription	R2D_LST_STR_PRE	YES

Retrieval of most recent unstructured Prescription	R2D_LST_UNSTR_PRE	YES
Direct access by ID to a specific resource	R2D_GET_RES	YES

4.3.1. Search Narrowing

All the R2D operations imply an (eIDAS) authenticated citizen that must match to an existing patient stored in the same server where the request has been submitted and that is the concrete subject of such operations. Incoming requests submitted by citizens that do not have a matching patient, MUST be rejected with HTTP error code: 404 - Not Found.

R2D specifications do not allow specifying any subject in queries. Incoming requests providing search criteria over a subject / patient are not compliant to R2D specifications and MUST be rejected with HTTP error code: 403 - Forbidden.

R2D implementers MUST apply search narrowing to incoming requests, to narrow the scope of searches in order to automatically restrict the searches to the only authorized compartment, which is the one of the authenticated citizen. The following request:

```
http://<baseurl>/DiagnosticReport?code=XYZ
```

must be handled in the compartment of the authenticated citizen as if it would be:

```
http://<baseurl>/DiagnosticReport?subject=<auth citizen>code=XYZ
```

Technical details about identity and session management are specified in the deliverable [D3.3].

4.3.2. Allowed HTTP methods

R2D is a read only protocol, thus the only allowed HTTP method for R2D is the GET method. All other FHIR requests executed over one of the following HTTP methods POST, PUT, DELETE, MUST BE rejected with HTTP error code: 405 - Method Not allowed.

4.3.3. R2D FHIR mapping

Operation R2D_SRC_STR_PAT_SUM (Search of structured Patient Summary)
This operation allows the retrieval of the structured version of the patient summary of the authenticated citizen.
The operations working with structured version of patient summary, follow rules defined by International Patient Summary [FHIR IPS SPECS], these specifications assert that a valid Patient Summary must be contained in a FHIR resource named Bundle with type “document”, the first entry of the Bundle must be an instance of Composition with a specific code (LOINC 60591-5). The Bundle must contain all the instances of other resources referenced by the Composition, so that patient summary is consistent.

The following tables show the specifications of a valid R2D-FHIR request for searching patient summary:

Property	Value
FHIR Resource	Bundle
HTTP Method	GET
Header Params	● Accept: application/fhir+xml,application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/Bundle
Client Search Params	● type ● composition.type ● _sort
Return Value	● Instance of Bundle of type searchset containing instances of Bundle of type document represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: DSTU3 ● Profile: http://hl7.org/fhir/uv/ips/StructureDefinition/composition-uv-ips
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access requested resources due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point.

	• **406 Not Acceptable**: client requested a not supported content-type format. • **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
composition.type	token	YES	Value MUST be = http://loinc.org|60591-5
type	token	YES	Value MUST be = document
_sort	text	YES	Value MUST be = -timestamp

Examples:

```
<base>/Bundle?type=document&composition.type=http://loinc.org|60591-5&_sort=-timestamp
```

Operation R2D_LST_STR_PAT_SUM (Retrieval of most recent structured Patient Summary)

This operation retrieves the most recent structured version of the patient summary of the authenticated citizen. The following tables show the specifications of a valid R2D-FHIR request:

Property	Value
FHIR Resource	Bundle
HTTP Method	GET
Header Params	● Accept: application/fhir+xml,application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/Bundle
Client Search Params	● type ● composition.type ● _sort ● _count
Return Value	● Instance of Bundle of type searchset containing one instance of Bundle of type document represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: DSTU3 ● Profile: http://hl7.org/fhir/uv/ips/StructureDefinition/composition-uv-ips
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access the requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	Type	Mandatory	Constraint
composition.type	token	YES	Value MUST be = http://loinc.org|60591-5
type	token	YES	Value MUST be = document
_count	number	YES	Value MUST be = 1
_sort	string	YES	Value MUST be = -timestamp

Examples:

```
<base>/Bundle?type=document&composition.type=http://loinc.org|60591-5&_count=1&_sort=-
timestamp
```

Operation R2D_SRC_STR_LAB_RES (Search of structured Laboratory Results)
This operation executes a search over the set of structured Laboratory Results of the authenticated citizen.

According to InteropEHRate FHIR profile, Laboratory Results have been mapped to the FHIR resource named DiagnosticReport. The values of the laboratory result are mapped to resource Observation and are accessible from Diagnostic Report's attribute named "result".

The following tables show the specifications of a valid request for searching Laboratory Results:

Property	Value
FHIR Resource	Diagnostic Report
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/DiagnosticReport
Client Search Params	● date ● category ● _sort
Return Value	● Instance of Bundle of type searchset containing instances of DiagnosticReport represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access the requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
date	date	NO	Value is assigned by client, MUST respect FHIR date data type specifications
category	token	YES	Value MUST be = http://terminology.hl7.org/CodeSystem/v2-0074|LAB
_sort	string	YES	Value MUST be = -date

Examples:

```
<base>/DiagnosticReport?category=http://terminology.hl7.org/CodeSystem/v2-
0074|LAB&_sort=-date
```

```
<base>/DiagnosticReport?category=http://terminology.hl7.org/CodeSystem/v2-
0074|LAB&date=ge2020-10-20&_sort=-date
```

Operation R2D_LST_STR_LAB_RES (Search of most recent Laboratory Results)
This operation retrieves the most recent structured version of Laboratory Results of the authenticated citizen.

According to InteropEHRate FHIR profile, Laboratory Results have been mapped to the FHIR resource named DiagnosticReport. The values of the laboratory result are mapped to resource Observation and are accessible from Diagnostic Report's attribute named "result".

The following tables show the specifications of a valid request for search patient summary:

Property	Value
FHIR Resource	Diagnostic Report
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/DiagnosticReport
Client Search Params	● category ● _count ● _sort
Return Value	● Instance of Bundle of type searchset containing one instance of DiagnosticReport represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access the requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
category	token	YES	Value MUST be = http://terminology.hl7.org/CodeSystem/v2-0074|LAB
_count	number	YES	Value MUST be = 1
_sort	string	YES	Value MUST be = -date

Examples:

```
<base>/DiagnosticReport?category=http://terminology.hl7.org/CodeSystem/v2-
0074|LAB&_count=1_sort=-date
```

Operation R2D_SRC_STR_MED_IMG (Search of structured Medical Images)

This operation executes a search over the set of structured Medical Images of the authenticated citizen.

According to InteropEHRate FHIR profile, Medical Images have been mapped to a FHIR resource named DiagnosticReport, where an instance of DiagnosticReport acts as a container for the set of images that are part of the study. Data about images are mapped to the ImagingStudy FHIR resource and are accessible from DiagnosticReport's attribute named "imagingStudy".

The following tables show the specifications of a valid request for searching Medical Images:

Property	Value
FHIR Resource	Diagnostic Report
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/DiagnosticReport
Client Search Params	● date ● category ● _sort
Return Value	● Instance of Bundle of type searchset containing instances of DiagnosticReport represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access the requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
date	date	NO	Value is assigned by client, MUST respect FHIR date data type specifications
category	token	YES	Value MUST be = http://terminology.hl7.org/CodeSystem/v2-0074|RAD,http://terminology.hl7.org/CodeSystem/v2-0074|RX,http://terminology.hl7.org/CodeSystem/v2-0074|NMR
_sort	text	YES	Value MUST be = -date

Examples:

```
<base>/DiagnosticReport?category=http://terminology.hl7.org/CodeSystem/v2-
0074|RAD,http://terminology.hl7.org/CodeSystem/v2-
0074|RX,http://terminology.hl7.org/CodeSystem/v2-0074|NMR&_sort=-date
```

```
<base>/DiagnosticReport?category=http://terminology.hl7.org/CodeSystem/v2-
0074|RAD,http://terminology.hl7.org/CodeSystem/v2-
0074|RX,http://terminology.hl7.org/CodeSystem/v2-0074|NMR&date=ge2020-10-20&_sort=-date
```

Operation R2D_LST_STR_MED_IMG (Retrieval of most recent structured Medical Images)
This operation retrieves the most recent structured Medical Images of the authenticated citizen.

According to InteropEHRate FHIR profile, Medical Images have been mapped to the FHIR resource named DiagnosticReport, where an instance of DiagnosticReport acts as a container for the set of images that are part of the study. Data about images are mapped to the ImagingStudy FHIR resource and are accessible from DiagnosticReport's attribute named "imagingStudy".

The following tables show the specifications of a valid request for searching Medical Images:

Property	Value
FHIR Resource	Diagnostic Report
HTTP Method	GET
Header Params	• Accept: application/fhir+xml, application/fhir+json • Authorization: JSON Web Token for session management • every header parameter defined by FHIR specifications
URL	http://[base url]/DiagnosticReport
Client Search Params	• category • _sort • _count
Return Value	• Instance of Bundle of type searchset containing one instance of DiagnosticReport represented with Content-Type corresponding to requested Accept parameter, default value is JSON. • Version: R4 • Profile: InteropEHRate profile
HTTP Return Codes	• **200 Successful**: request was successfully processed. • **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. • **401 Not Authorized**: authorization is required for the interaction that was attempted. • **403 Forbidden**: client is not allowed to access the requested resource due to security policy. • **404 Not Found**: resource type not supported, or not a valid FHIR end-point. • **406 Not Acceptable**: client requested a not supported content-type format. • **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
category	token	YES	Value MUST be = http://terminology.hl7.org/CodeSystem/v2-0074|RAD,http://terminology.hl7.org/CodeSystem/v2-0074|RX,http://terminology.hl7.org/CodeSystem/v2-0074|NMR
_count	number	YES	Value MUST be = 1
_sort	text	YES	Value MUST be = -date

Examples:

```
<base>/DiagnosticReport?category=http://terminology.hl7.org/CodeSystem/v2-
0074|RAD,http://terminology.hl7.org/CodeSystem/v2-
0074|RX,http://terminology.hl7.org/CodeSystem/v2-0074|NMR&_count=1&_sort=-date
```

Operation R2D_SRC_STR_PRE (Search of structured Prescriptions)

This operation executes a search over the set of structured Prescriptions of the authenticated citizen. According to InteropEHRate FHIR profile, Prescriptions have been mapped to the FHIR resource named MedicationRequest.

The following tables show the specifications of a valid request for searching Prescriptions:

Property	Value
FHIR Resource	MedicationRequest
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/MedicationRequest
Client Search Params	● authoredon ● category ● _sort
Return Value	● Instance of Bundle of type searchset containing instances of MedicationRequest represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access the requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
authoredon	date	NO	Value is assigned by client, MUST respect FHIR date data type specifications
category	token	YES	Value MUST be = http://hl7.org/fhir/ValueSet/medicationrequest-category|outpatient, http://hl7.org/fhir/ValueSet/medicationrequest-category|community
_sort	text	YES	Value MUST be = -authoredon

Examples:

```
<base>/MedicationRequest?category=http://hl7.org/fhir/ValueSet/medicationrequest-
category|outpatient,http://hl7.org/fhir/ValueSet/medicationrequest-
category|community&_sort=-authoredon
```

```
<base>/MedicationRequest?category=http://hl7.org/fhir/ValueSet/medicationrequest-
category|outpatient,http://hl7.org/fhir/ValueSet/medicationrequest-
category|community&authoredon=ge2020-10-20&_sort=-authoredon
```

Operation R2D_LST_STR_PRE (Retrieval of most recent structured Prescriptions)

This operation retrieves the most recent structured Prescriptions of the authenticated citizen. According to the FHIR InteropEHRate profile, Prescriptions have been mapped to the FHIR resource named MedicationRequest.

The following tables show the specifications of a valid request for searching Prescriptions:

Property	Value
FHIR Resource	MedicationRequest
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/MedicationRequest
Client Search Params	● category ● _count ● _sort
Return Value	● Instance of Bundle of type searchset containing one instance of MedicationRequest represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access the requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
category	token	YES	Value MUST be = http://hl7.org/fhir/ValueSet/medicationrequest-category|outpatient, http://hl7.org/fhir/ValueSet/medicationrequest-category|community
_count	number	YES	Value MUST be = 1
_sort	text	YES	Value MUST be = -authoredon

Examples:

```
<base>/MedicationRequest?category=http://hl7.org/fhir/ValueSet/medicationrequest-
category|outpatient,http://hl7.org/fhir/ValueSet/medicationrequest-
category|community&_count=1&_sort=-authoredon
```

Operation R2D_GET_RES (Direct access by ID to a specific resource)

This operation in FHIR is naturally executed using the URL provided with every instance of Resource. The following tables show the specifications of a valid request for searching Prescriptions:

Property	Value
FHIR Resource	One of: DiagnosticReport, Bundle, Composition, Observation, MedicationRequest, AllergyIntollerance, Condition, Medication, MedicationStatement
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	corresponds to provide resource id (universal FHIR identifier are expressed with URL)
Client Search Params	No parameters allowed
Return Value	● Instance of the requested resource represented with Content-Type corresponding to requested Accept parameter, default value is JSON.
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access the requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Operation R2D_SRC_UNSTR_PAT_SUM ()

This operation searches unstructured Patient Summary of the authenticated citizen. According to the FHIR InteropEHRate profile, unstructured Patient Summary has been mapped to FHIR resource named DocumentReference.

The following tables show the specifications of a valid request for searching unstructured Patient Summary:

Property	Value
FHIR Resource	DocumentReference
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/DocumentReference
Client Search Params	● type ● _sort
Return Value	● Instance of Bundle of type searchset containing instances of DocumentReference represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	Type	Mandatory	Constraint
type	token	YES	Value MUST be = http://loinc.org|60591-5
_sort	text	YES	Value MUST be = -date

Examples:

```
<base>/DocumentReference?type=http://loinc.org|60591-5&_sort=-date
```

Operation R2D_LST_UNSTR_PAT_SUM ()

This operation retrieves the most recent unstructured Patient Summary of the authenticated citizen. According to the FHIR InteropEHRate profile, unstructured Patient Summary has been mapped to FHIR resource named DocumentReference.

The following tables show the specifications of a valid request for searching unstructured Patient Summary:

Property	Value
FHIR Resource	DocumentReference
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/DocumentReference
Client Search Params	● type ● _sort
Return Value	● Instance of Bundle of type searchset containing one instance of DocumentReference represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
type	token	YES	Value MUST be = http://loinc.org|60591-5
_count	number	YES	Value MUST be = 1
_sort	text	YES	Value MUST be = -date

Examples:

```
<base>/DocumentReference?type=http://loinc.org|60591-5&_count=1&_sort=-date
```

Operation R2D_SRC_UNSTR_LAB_RES ()

This operation searches unstructured Laboratory Results of the authenticated citizen. According to the FHIR InteropEHRate profile, unstructured Laboratory Results have been mapped to the FHIR resource named DocumentReference.

The following tables show the specifications of a valid request for searching unstructured Laboratory Results:

Property	Value
FHIR Resource	DocumentReference
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/DocumentReference
Client Search Params	● category ● _sort
Return Value	● Instance of Bundle of type searchset containing instances of DocumentReference represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	**Type**	**Mandatory**	**Constraint**
category	token	YES	Value MUST be = http://terminology.hl7.org/CodeSystem/v2-0074|LAB
_sort	text	YES	Value MUST be = -date

Examples:

```
<base>/DocumentReference?category=http://terminology.hl7.org/CodeSystem/v2-
0074|LAB&_sort=-date
```

Operation R2D_LST_UNSTR_LAB_RES ()

This operation retrieves the most recent unstructured Laboratory Results of the authenticated citizen. According to the FHIR InteropEHRate profile, unstructured Laboratory Results have been mapped to the FHIR resource named DocumentReference.

The following tables show the specifications of a valid request for searching unstructured Laboratory Results:

Property	Value
FHIR Resource	DocumentReference
HTTP Method	GET
Header Params	● Accept: application/fhir+xml, application/fhir+json ● Authorization: JSON Web Token for session management ● every header parameter defined by FHIR specifications
URL	http://[base url]/DocumentReference
Client Search Params	● category ● _count ● _sort
Return Value	● Instance of Bundle of type searchset containing one instance of DocumentReference represented with Content-Type corresponding to requested Accept parameter, default value is JSON. ● Version: R4 ● Profile: InteropEHRate profile
HTTP Return Codes	● **200 Successful**: request was successfully processed. ● **400 Bad Request**: search could not be processed or failed basic FHIR validation rules. ● **401 Not Authorized**: authorization is required for the interaction that was attempted. ● **403 Forbidden**: client is not allowed to access requested resource due to security policy. ● **404 Not Found**: resource type not supported, or not a valid FHIR end-point. ● **406 Not Acceptable**: client requested a not supported content-type format. ● **500 Internal Server Error**: server encountered an unexpected internal error, the request could not be processed.

Allowed search parameters:

Client Search Parameters			
Name	Type	Mandatory	Constraint
category	token	YES	Value MUST be = http://terminology.hl7.org/CodeSystem/v2-0074|LAB
_count	number	YES	Value MUST be = 1
_sort	text	YES	Value MUST be = -date

Examples:

```
<base>/DocumentReference?category=http://terminology.hl7.org/CodeSystem/v2-
0074|LAB&_count=1&_sort=-date
```

Operation R2D_SRC_UNSTR_MED_IMG ()

This operation will be supported by R2D over FHIR, but definition of FHIR Profile for unstructured Medical Images is scheduled after the release of this version (2) of [D4.2]. Specifications for this operation will be added to version 3 of this deliverable.

Operation R2D_LST_UNSTR_MED_IMG ()

This operation will be supported by R2D over FHIR, but definition of FHIR Profile for unstructured Medical Images is scheduled after the release of this version (2) of [D4.2]. Specifications for this operation will be added to version 3 of this deliverable.

Operation R2D_SRC_UNSTR_PRE ()

This operation will be supported by R2D over FHIR, but definition of FHIR Profile for unstructured Prescriptions is scheduled after the release of this version (2) of [D4.2]. Specifications for this operation will be added to version 3 of this deliverable.

Operation R2D_LST_UNSTR_PRE ()

This operation will be supported by R2D over FHIR, but definition of FHIR Profile for unstructured Prescriptions is scheduled after the release of this version (2) of [D4.2]. Specifications for this operation will be added to version 3 of this deliverable.

4.4. R2D Cloud Conceptual Description

The InteropEHRate platform is composed also by a storage sub system, available to citizens (read and write mode) for backup purposes and to HCPs (read only) for emergency purposes.

The cloud storage is first of all partitioned per citizen, each citizen has its own private storage space. Each citizen partition is furtherly organized into three buckets:

- Emergency: contains mainly the Patient Summary but also all other relevant data needed by HCPs in case of emergency.
- EHR: contains Laboratory Reports and Prescriptions of the citizen.
- Images: contains medical images and imaging studies.

This structure cannot be modified by the citizen and is managed by the administrator of the platform. All data is saved to the cloud as a binary encrypted collection, and can be decrypted only by the citizen itself, or in case of emergency by HCPs. Periodic backups executed by the citizen are saved to Cloud transferring only the subset of data modified or created after the previous backup. Every transfer operation creates a commit on the Cloud, the sequence of commits allows cloud users to rebuild the correct sequence of transfer of data (Figure 17).

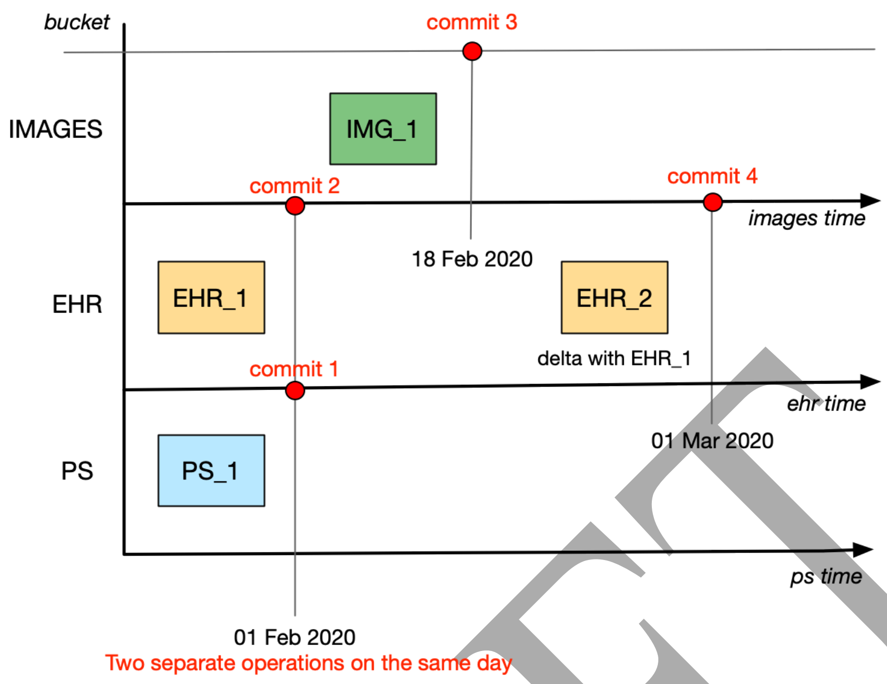


Figure 17 - Sequence of commits

The download operations will always return blob files sorted by commit so that it will be possible to restore the same situation as the reference commit. Download operations may work per bucket (allow to rebuild a bucket) or cross bucket (allows to rebuild entire storage). Data contained in a blob file may override data restored by a previously downloaded blob file; it is important to remember that health data on the device cannot be deleted, but only updated or created.

4.4.1. R2D Cloud Data Model

The following class diagram (Figure 18) shows the data model of R2Cloud that is an extension of the common R2D and D2D data model shown at the beginning of this document: Conceptual D2D and R2D datamodel. The most important class of the diagram is the Bucket that is a container of BlobInfo (metadata about a crypted bundle). The BlobInfo class contains details for requesting to R2D Cloud the download of their content as an encrypted bundle of health data.

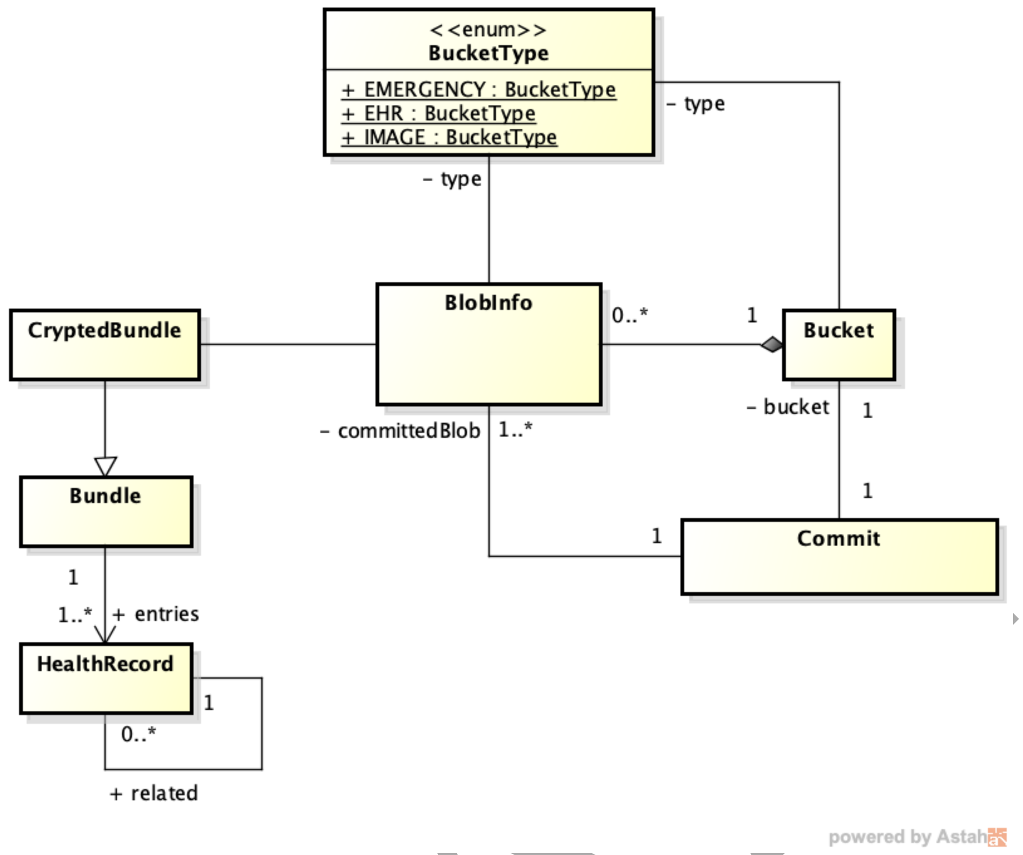


Figure 18 - R2Cloud data model

The following table describes every class of the data model (except for those that have been described in the common data model section):

Name	Description
CryptedBundle	CryptedBundle acts as a normal Bundle (container of a set of HealthRecord) with the difference that the content is represented as an encrypted set of bytes. Relationships: ● Bundle: extended class
BlobInfo	BlobInfo contains some general meta data (type, size, number of items, creation date, etc. etc.) about a blob file. Relationships: ● CryptedBundle: this relationship connects an instance of BlobInfo with the related content that it describes.
Bucket	A Bucket is a container of BlobInfo. Relationships: ● BlobInfo: relationship with the contained instances of Blobinfo. ● BucketType: relationship used to represent the type of an instance of

	Bucket. ● Commit: relationship with the set of instances of commit made to an instance of Bucket.
BucketType	An enumeration that defines the type of allowed buckets.
Commit	A Commit represents a successful event of storage of a blob file inside an instance of Bucket. Relationships: ● Bucket: relationship with the instance of Bucket that was the subject of the commit.

4.4.2. R2D Cloud API

APIs of R2D Cloud are partitioned into two interfaces (Figure 19), one defining read operations (to be used both by citizens and HCPs), one defining write operations (to be used only by citizens). The following class diagrams show the conceptual API of R2Cloud (such interfaces will be furtherly transformed into concrete technical specifications):

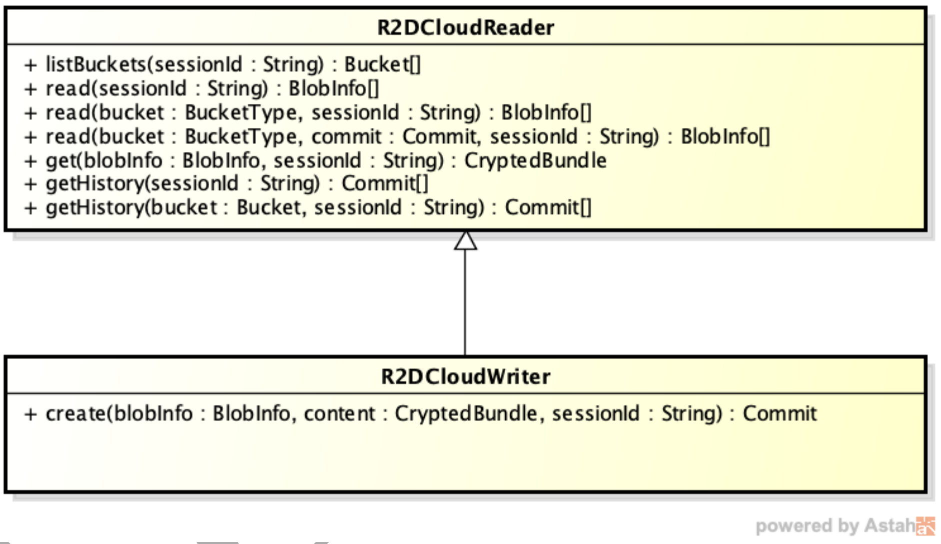

R2DCloudReader
+ listBuckets(sessionId : String) : Bucket[]
+ read(sessionId : String) : BlobInfo[]
+ read(bucket : BucketType, sessionId : String) : BlobInfo[]
+ read(bucket : BucketType, commit : Commit, sessionId : String) : BlobInfo[]
+ get(blobInfo : BlobInfo, sessionId : String) : CryptedBundle
+ getHistory(sessionId : String) : Commit[]
+ getHistory(bucket : Bucket, sessionId : String) : Commit[]
R2DCloudWriter
+ create(blobInfo : BlobInfo, content : CryptedBundle, sessionId : String) : Commit
powered by Astah

Figure 19 - R2Cloud Conceptual API

Operation R2DCloudReader.listBuckets

Name	listBuckets
Description	This operation allows retrieval of the set of instances of Bucket that compose the citizen storage space. Each instance of Bucket contains operation and information for retrieving their content.
Arguments	● String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.

Return Value	list of Bucket instances.
Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation R2DCloudReader.read

Name	read
Description	This operation allows the reading of all blob files (of a citizen) stored to the cloud. This operation retrieves all the instances of BlobInfo of all buckets sorted by creation date.
Arguments	• String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.
Return Value	list of BlobInfo instances.
Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation R2DCloudReader.read

Name	read
Description	This operation allows the reading of all blob files (of a citizen) stored in a specific bucket. This operation retrieves all the instances of BlobInfo of the requested bucket sorted by creation date.
Arguments	• BucketType bucket: an instance of BucketType, representing the bucket subject of the operation. • String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.
Return Value	list of BlobInfo instances.

Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation R2DCloudReader.read

Name	read
Description	This operation allows the reading of the blob files (of a citizen) stored in a bucket after a specific commit. All the instances of returned BlobInfo are sorted by creation date.
Arguments	• BucketType bucket: an instance of BucketType, representing the bucket subject of the operation. • Commit commit: an instance of Commit (retrieved previously from the cloud), it represents the commit after which the client is requesting data. • String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.
Return Value	list of BlobInfo instances.
Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation R2DCloudReader.get

Name	get
Description	This operation allows the retrieval of the blob file (content) associated with an instance of BlobInfo.
Arguments	• BlobInfo blobInfo: an instance of blobInfo providing information about the blob file that must be downloaded. • String sessionId: a valid session token representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.
Return Value	Instance of CryptedBundle

Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation R2DCloudReader.getHistory

Name	getHistory
Description	This operation allows the retrieval of the complete set of commits made by the citizen to his / her cloud storage space.
Arguments	• String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.
Return Value	List of instances of Commit sorted by creation date.
Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

Operation R2DCloudReader.getHistory

Name	getHistory
Description	This operation allows the retrieval of the complete set of commits made by the citizen to one specific bucket of his / her cloud storage space.
Arguments	• Bucket bucket: a valid instance of Bucket, representing the bucket subject of the operation. • String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.
Return Value	List of instances of Commit sorted by creation date.
Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure.

	• The session is still valid.

Operation R2DCloudWriter.create

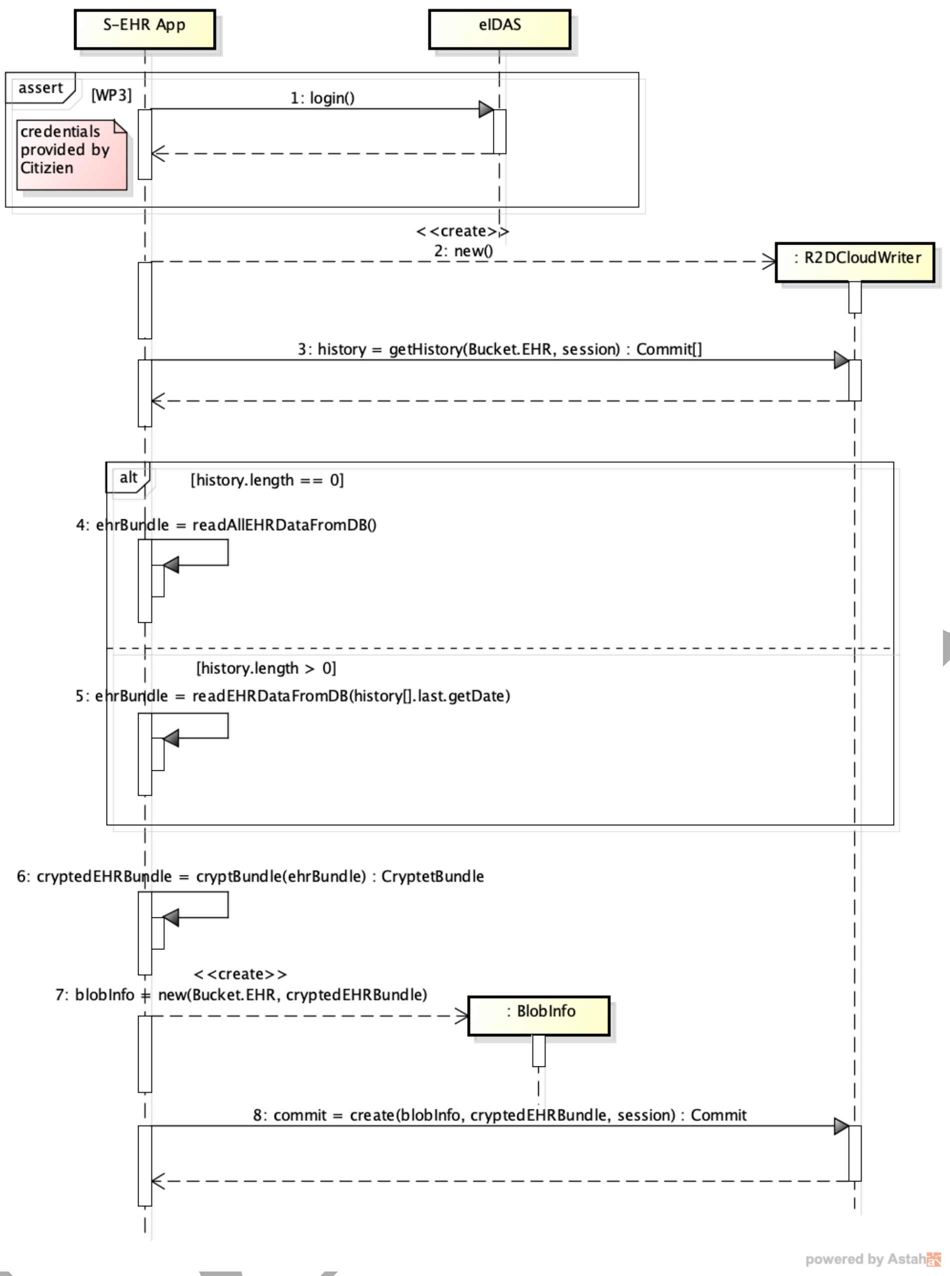
Name	create
Description	This operation allows a client to create a new blob file on the cloud.
Arguments	• BlobInfo blobInfo: an instance of blobInfo providing information about the blob file that must be created. • CryptedBundle content: an instance of CryptedBundle representing the content that must be stored to the cloud. • String sessionId: a valid session token, representing the user who successfully executed the login. This sessionId is obtained directly from the platform after successful execution of the login method.
Return Value	
Exceptions	• Security exceptions related to the validation of the session. • Network exceptions related to failure during remote communication.
Preconditions	• The citizen has successfully executed the authentication to the InteropEHRate infrastructure. • The session is still valid.

4.4.3. R2D Cloud Sample Interactions

This section describes two sample interactions that show how the API and the data of R2D, described in previous sections, are used to store data on the cloud and to retrieve data from the cloud.

Cloud Data Storage

The following figure (Figure 20) shows the UML sequence diagram for the storage of a new blob file on the EHR bucket of the cloud:



S-EHR App
eIDAS
assert
[WP3]
credentials provided by Citizien
1: login()
<<create>>
2: new()
: R2DCloudWriter
3: history = getHistory(Bucket.EHR, session) : Commit[]
alt
[history.length == 0]
4: ehrBundle = readAllEHRDataFromDB()
[history.length > 0]
5: ehrBundle = readEHRDataFromDB(history[].last.getDate)
6: cryptedEHRBundle = cryptBundle(ehrBundle) : CryptetBundle
<<create>>
7: blobInfo = new(Bucket.EHR, cryptedEHRBundle)
: BlobInfo
8: commit = create(blobInfo, cryptedEHRBundle, session) : Commit
powered by Astah

Figure 20 - Sequence of storing blob files

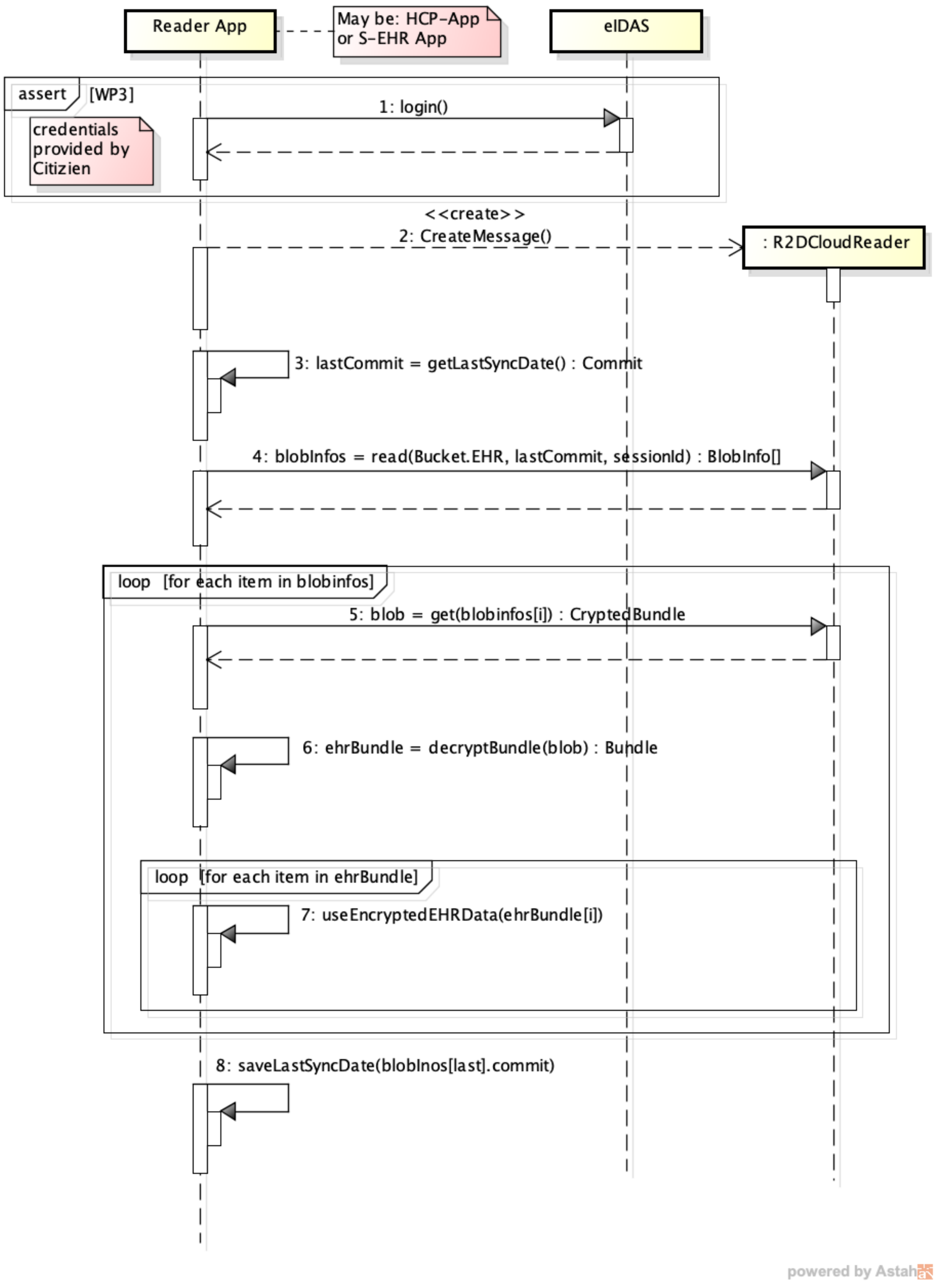
- Step 1: the citizen authenticates on an eIDAS compliant server. All the operations of R2D Cloud require an eIDAS authenticated citizen.
- Step 2: S-EHR app creates an instance of R2DCloudWriter.
- Step 3: S-EHR app asks to cloud the history of commits of bucket EHR. This operation is useful because in this way the S-EHR app gets information about the status of the EHR bucket in order to determine the status of previous backups.
- Step 4: if the S-EHR app does not retrieve any status from the EHR bucket, it means that this bucket is empty and no backup has ever been saved inside it. So, the S-EHR app retrieves all EHR data (laboratory results and prescriptions) from its internal database and creates an instance of Bundle (ehrBundle).
- Step 5: if the S-EHR app retrieves a status from the bucket, it means that the bucket is not empty and some backups have already been made. So the S-EHR app retrieves from its internal DB all EHR

data created or updated after the date of the last status and creates an instance of Bundle (ehrBundle).

- Step 6: The bundle created in step 4 or 5 is encrypted.
- Step 7: The S-EHR app creates a new instance of BlobInfo, passing as argument the encrypted bundle.
- Step 8: the S-EHR app invokes the create() method providing as arguments the blobInfo and the encrypted bundle. This method returns the instance of the related Commit created for this creation operation. It is up to the S-EHR to choose if this instance of Commit should be saved locally or if (as shown in this diagram) asks the status to the server prior to the beginning of a create operation.

Cloud Data Copy

The following figure (Figure 21) shows the UML sequence diagram for reading and copying (in a local database) all the blob files from the EHR bucket of the cloud.



Reader App
May be: HCP-App or S-EHR App
eIDAS
assert [WP3]
credentials provided by Citizien
1: login()
<<create>>
2: CreateMessage()
: R2DCloudReader
3: lastCommit = getLastSyncDate() : Commit
4: blobInfos = read(Bucket.EHR, lastCommit, sessionId) : BlobInfo[]
loop [for each item in blobinfos]
5: blob = get(blobinfos[i]) : CryptedBundle
6: ehrBundle = decryptBundle(blob) : Bundle
loop [for each item in ehrBundle]
7: useEncryptedEHRData(ehrBundle[i])
8: saveLastSyncDate(blobInos[last].commit)
powered by Astah

Figure 21 - Sequence of copying data

- Step 1: the citizen authenticates on an eIDAS compliant server. All the operations of R2D Cloud require an eIDAS authenticated citizen.
- Step 2: S-EHR app creates an instance of R2DCloudReader.
- Step 3: the S-EHR app retrieves from its local database the last synchronization timestamp (stored as an instance of Commit). If no commit is stored (first synchronization) then the value null will be passed to the read method (next step).
- Step 4: the S-EHR app invokes the read method to ask the cloud the list of all BlobInfo instances from the EHR Bucket after a certain Commit (if it is null, all BlobInfo will be returned).
- Step 5-6-7: the S-EHR loops over the set of returned BlobInfo instances executing the following operation for each item:
  - Step 5: invokes the get method to retrieve the associated blob file.
  - Step 6: invokes the decryption method to decrypt blob file
  - Step 7: starts another loop iterating over the set of HealthRecords contained in the decrypted bundle and storing them in a local database.
- Step 8: the S-EHR app stores the Commit associated to the last instance of BlobInfo, to determine last synchronization timestamp.

5. CONCLUSIONS AND NEXT STEPS

The objective of this report was to deliver the second version of the design and the specification of both the D2D and the R2D protocols. With the implementation of D2D it will become feasible to achieve the exchange of healthcare related data between a healthcare practitioner and a mobile application of a citizen, without the usage of internet connection, whereas with the implementation of R2D it will become feasible to exchange healthcare related data between any EHR or S-EHR Cloud and a mobile application of a citizen, with the usage of internet. It should be mentioned that one final updated version of this report is planned to be released in March 2021, including the final relevant updates, of both the D2D and the R2D protocols, that will have taken place until then.

In more detail, regarding the specification of the D2D protocol it should be mentioned that in the 1st version of the D2D protocol specification [D4.1], the specification was performed based only on the Android Bluetooth API since the D2D protocol was in a very immature phase, and we would like to perform additional research on the different platforms for identifying the Bluetooth profiles that could satisfy all the requirements as specified in D2.2 [D2.2]. However, in the current version of the D2D protocol specification [D4.2], the D2D protocol specification is independent from any API in order for any developer that would like to use this protocol to be able to use the listed operations or develop her own operations, in order to implement the corresponding data exchange between a S-EHR and an HCP application.

Moreover, regarding the specification of the R2D protocol, in the 1st version of the R2D protocol specification [D4.1] the focus was on the Patient Summary transfer, while the second version has enlarged the set of supported health data and has introduced requirements for R2DCloud. Next version will define specifications for all health data defined in [D2.7] and will define complete technical specifications of R2DCloud.

REFERENCES

- **[D2.2]** InteropEHRate Consortium, *D2.2-User Requirements for cross-border HR integration - V2*, March 2020. https://www.interopehrate.eu/resources/#dels
- **[D2.5]** InteropEHRate Consortium, *D2.5 - InteropEHRate Architecture - V2*, March 2020. https://www.interopehrate.eu/resources/#dels
- **[D2.7]** InteropEHRate Consortium, *D2.7-FHIR profile for EHR interoperability - V1*, September 2019. https://www.interopehrate.eu/resources/#dels
- **[D3.1]** InteropEHRate Consortium, *D3.1 - Specification of S-EHR mobile privacy and security conformance levels - V1*, March 2020. https://www.interopehrate.eu/resources/#dels
- **[D3.3]** InteropEHRate Consortium, *D3.3-Specification of remote and D2D IDM mechanisms for HRs Interoperability - V1*, June 2019. https://www.interopehrate.eu/resources/#dels
- **[D4.1]** InteropEHRate Consortium, D4.1-Specification of remote and D2D protocol and APIs for HR exchange - V1, June 2019. https://www.interopehrate.eu/resources/#dels
- **[D4.2]** InteropEHRate Consortium, D4.2-Specification of remote and D2D protocol and APIs for HR exchange - V2, March 2020. https://www.interopehrate.eu/resources/#dels
- **[D4.8]** InteropEHRate Consortium, *D4.8-Specification of protocol and APIs for research health data sharing - V1*, March 2020. https://www.interopehrate.eu/resources/#dels
- **[D4.4]** InteropEHRate Consortium, *D4.4-Design of libraries for remote and D2D HR exchange V1*, June 2019. https://www.interopehrate.eu/resources/#dels
- **[ANT+]** ANT+, Website: https://www.thisisant.com/consumer/ant-101/what-is-ant
- **[BLUETOOTH]** Bluetooth Core Specification, Website: https://www.bluetooth.com/specifications/bluetooth-core-specification/
- **[ENOCEAN]** EnOcean Self-powered IoT, Website: https://www.enocean.com/en/
- **[NFC]** NFC forum, Website: https://nfc-forum.org/
- **[RFID]** How RFID Works, Website: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/rfid.htm
- **[ZIGBEE]** Zigbee alliance, Website: https://www.zigbee.org/
- **[WIFIDIRECT]** Wi-Fi Direct, Website: https://www.wi-fi.org/discover-wi-fi/wi-fi-direct
- **[ZWAVE]** Z-wave, Website: https://www.z-wave.com/
- **[SDDB]** Li, Sing, and Jonathan Knudsen. *Beginning J2ME: from novice to professional*. Apress, 2006.
- **[BCS]** Bluetooth Core Specification, https://www.bluetooth.com/specifications/bluetooth-core-specification/

- **[SPP]** Serial Port Profile, https://learn.sparkfun.com/tutorials/bluetooth-basics/bluetooth-profiles
- **[RFCOMM]** RFCOMM protocol, https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/rfcomm.html
- **[BASEBAND]** Bluetooth Baseband, https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/baseband.html
- **[LMP]** Link Manager Protocol, https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/lmp.html
- **[L2CAP]** Logical Link Control and Adaptation Protocol, https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/l2cap.html
- **[OSI]** The protocol stack, http://www.informit.com/articles/article.aspx?p=27591&seqNum=5
- **[GSM]** GSM ts07.10, https://www.gefos-leica.cz/data/.../877301_leica_ts03_07_10_eql_v1-0-0_en.pdf
- **[SDP]** Service Discovery Protocol Layer, https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/sdp.html
- **[PAN]** Personal Area Networking Profile, http://grouper.ieee.org/groups/802/15/Bluetooth/PAN-Profile.pdf
- **[APPLE BT]** Bluetooth profiles that iOS and iPadOS support, https://support.apple.com/en-us/HT204387
- **[ME]** Management Entity, http://grouper.ieee.org/groups/802/15/Bluetooth/PAN-Profile.pdf
- **[EUCBEHR REC]** COMMISSION RECOMMENDATION of 6.2.2019 on a European Electronic Health Record exchange format.
- **[EUCBEHR ANNEX]** ANNEX to the Commission Recommendation of 6.2.2019 on a European Electronic Health Record exchange format.
- **[EHDSI SYS SPECS]** EHDSI System Architecture Specification v2.1.0, 01 June 2017.
- **[NCPeH ARCH SPECS]** NCPeH System Architecture Specification v2.1.0, 01 June 2017.
- **[EPSOS ARCH]** D3.2.2 System Technical Specification v 1.4, 28 April 2010.
- **[EPSOS SPECS]** D3.4.2 epSOS Common Components v 1.0, 16 July 2010.
- **[HL7 FHIR]** HL7 FHIR, https://www.hl7.org/fhir
- **[FHIR API SPEC]** FHIR RESTful API R4 (version 4.0.0), Website: https://www.hl7.org/fhir/http.html
- **[FHIR IPS SPECS]** FHIR International Patient Summary, https://build.fhir.org/ig/HL7/fhir-ips/index.html
- **[OPENEHR]** OpenEHR, https://www.openehr.org/
- **[ARGONAUT]** HL7 Argonaut, https://argonautwiki.hl7.org/
- **[APPLE HEALTH]** Apple Healthcare,https://www.apple.com/healthcare/health-records/
- **[IHE IPS]** International Patient Summary IPS project, http://www.ehealth-standards.eu/en/projects/international-patient-summary-ips-project/
- **[IHE IPA]** HL7 International Patient Access - v 0.1.0 draft, https://build.fhir.org/ig/grahamegrieve/ipa-candidate/

ANNEX

Users' questionnaire

Communication Requirements

No.	Requirement	Answer to Requirement
CR1	What is the maximum delay in communication that is acceptable to you?	A few seconds
CR2	Would you prefer the data to be exchanged in a distance of a few centimeters between the 2 devices, or in a distance of no more than 10 meters?	There is not any preference

Table 4 - Communication Requirements

Data Requirements

No.	Requirement	Answer to Requirement
DR1	Do you foresee to exchange only textual data? Will you also exchange images and videos?	Both, textual data and images
DR2	For how long would you like the Physician to keep the accessed data after the Citizen leaves?	It depends on the situation
DR3	Would you like the Citizen to update her EHR with the newly derived data (i.e. the data that will emerge from her visit to the Physician)?	Yes

Table 5 - Data Requirements

Security Requirements

No.	Requirement	Answer to Requirement
SR1	Should the data exchange part be reliable (e.g. requirement of confirmation messages)?	Yes
SR2	How important is the authentication of the parties?	Very important
SR3	Is the transferred data confidential?	Yes

Table 6 - Security Requirements

User Interaction Requirements

No.	Requirement	Answer to Requirement
UI1	How many types of parties will be involved? Only Physicians and Citizens?	HCPs and citizens
UI2	Who will initiate the process? The Physician or the Citizen?	The Citizens
UI3	How would the Physician explain to the Citizen the data she wants to access? How would the Citizen understand the Physician's demands?	By providing an initial list of attribute names

Table 7- User Interaction Requirements

R2D over eHDSI

This section defines an architectural hypothesis based on the current eHDSI API specifications [NCPeH ARCH SPECS] and epSOS technical specifications [EPSOS ARCH][EPSOS SPECS]. The current API of an NCP is based on the IHE profile named XCA (Cross Community Access) extended for specific epSOS purposes (knowledge of IHE XCA profile is considered very important in order to completely understand eHDSI technical details). Deliverable [EPSOS SPECS] contains all technical details for server side implementation including XML examples of request and response messages.

The InteropEHRate project is not able to define technical specifications for the eHDSI project, but R2D and eHDSI protocols share so many objectives (concerning the exchange of health data) that technological impediments should be solved to facilitate cross border health data exchange for citizens and application developers. Current eHDSI specifications [EHDSI SYS SPECS] assert that the users of the system are only authenticated HCPs (having executed the authentication to their National Infrastructure), thus the exchange of health data occurs only between authorized HCOs pertaining to different member states, a citizen is not allowed to access eHDSI (neither the NCP of his/her country). In the hypothesis reported here, we describe the NCP as an Extended NCP, considering it as an evolution of the current NCP but also available to authenticated citizens belonging to the same country of the NCP (citizens of Country A are allowed only to access Extended NCP A). The motivations behind the choice of having also an implementation of R2D working on eHDSI have been already described before in this deliverable, but they can be summarized as: provide a unique interface to access health data in all Europe, facilitating life to citizens and applications developers.

The following figure shows a Citizen of Country A that download his health data using the S-EHR app and downloading data from the NCP of his country.

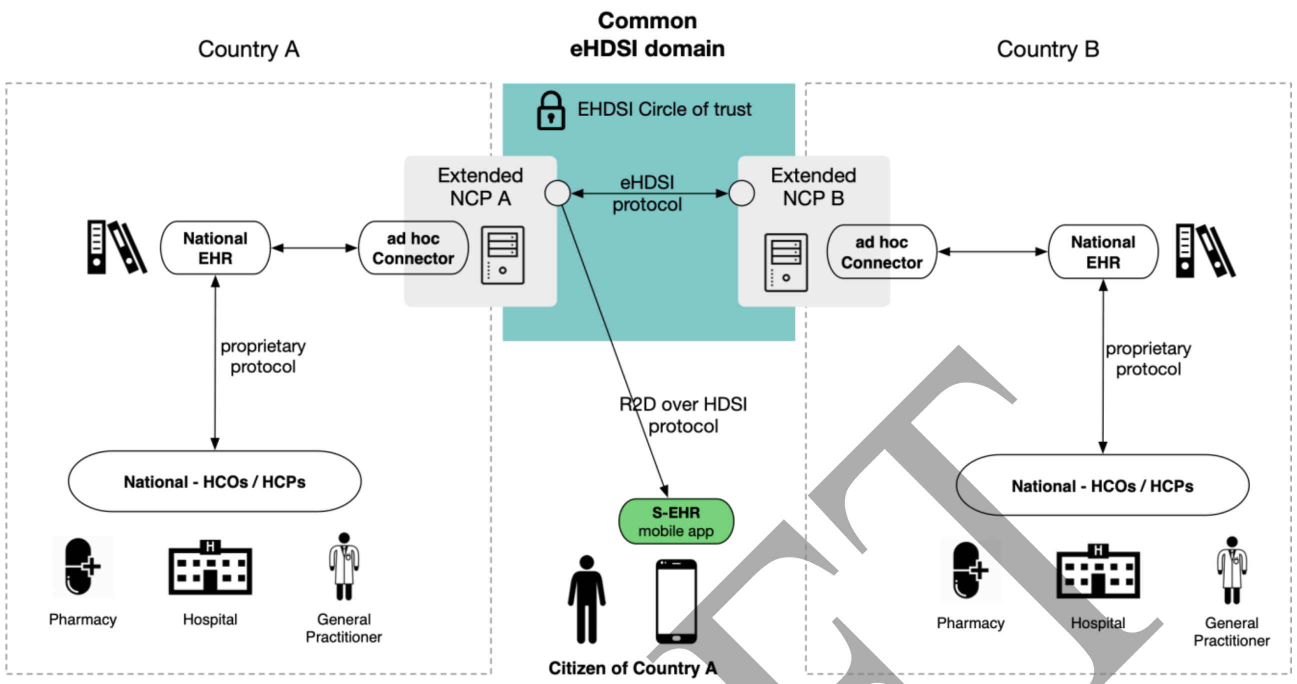


Figure 22 - Common eHDSI domain

Furthermore, current eHDSI technical specifications do not cover the whole kind of health data defined in [EUCBEHR_REC] by the EU Commission, they cover only the request of the Patient Summary and of the ePrescription / eDispensation. The following table contains shows what R2D operations may be supported by R2D over FHIR implementation:

Operation	Internal Code	Supported
Search of structured Patient Summary	R2D_SRC_STR_PAT_SUM	YES
Search of unstructured Patient Summary	R2D_SRC_UNSTR_PAT_SUM	YES
Retrieval of most recent structured Patient Summary	R2D_LST_STR_PAT_SUM	NO
Retrieval of most recent unstructured Patient Summary	R2D_LST_UNSTR_PAT_SUM	NO
Search of structured Laboratory Result	R2D_SRC_STR_LAB_RES	NO
Search of unstructured Laboratory Result	R2D_SRC_UNSTR_LAB_RES	NO
Retrieval of most recent structured Laboratory Result	R2D_LST_STR_LAB_RES	NO
Retrieval of most recent unstructured Laboratory Result	R2D_LST_UNSTR_LAB_RES	NO
Search of structured Medical Image	R2D_SRC_STR_MED_IMG	NO
Search of unstructured Medical Image	R2D_SRC_UNSTR_MED_IMG	NO
Retrieval of most recent structured Medical Image	R2D_LST_STR_MED_IMG	NO

Retrieval of most recent unstructured Medical Image	R2D_LST_UNSTR_MED_IMG	NO
Search of structured Prescription	R2D_SRC_STR_PRE	YES
Search of unstructured Prescription	R2D_SRC_UNSTR_PRE	YES
Retrieval of most recent structured Prescription	R2D_LST_STR_PRE	YES
Retrieval of most recent unstructured Prescription	R2D_LST_UNSTR_PRE	YES
Direct access by ID to a specific resource	R2D_GET_RES	NO

eHDSI is defined with SOAP Web Services, specifications of R2D over eHDSI defines how a specific operation is mapped to specific eHDSI SOAP service, in particular for each operation will be specified:

- the eHDSI interface(s) involved;
- the methods of the interfaces to be invoked;
- The arguments provided to methods and, if needed, the constraints defined over arguments values.

Operation R2D_SRC_STR_PAT_SUM and R2D_SRC_UNSTR_PAT_SUM (Search of structured and unstructured Patient Summary)

This operation, as it has been designed in eHDSI, allows to retrieve both structured (ePSOS pivot) or unstructured (source coded) versions of PatientSummary. The following table shows all specifications of such operation.

Property	Value
eHDSI Service Interface	PatientService
Service Interface method	list()
Header Params	● Session token obtained by authentication method ● epSOS HCP Identity Assertion
Body Arguments	● Instance of ListPatientRequest compliant to specifications reported in table below.
Header Parameters	● [PT] X.509 NCP-B service certificate ● [ST] eHealth DSI HP Identity Assertion ● [ST] eHealth DSI Treatment Relationship ● Confirmation Assertion [O]
Return Value	● Instance of ListPatientResponse containing results
Error	● 4701 - No consent ● 4702 - Weak Authentication ● 4703 - Insufficient rights ● 1102 - No Data ● 4201 - Unsupported Feature ● 4202 - Unknown Signifier ● 4203 - Transcoding Error ● 4204 - Unknown Filter

Allowed value for the argument instance of ListPatientRequest:

Method Arguments		
Name	**Mandatory**	**Constraint**
Patient Identifier	YES	Value MUST be equals to the identifier of the patient who performed the authentication.

epSOS CDA template qualifier	NO	Code defining the output format of requested health data. It may be an epSOS CDA common template or document coded according to the source of health data. The value of this argument is related to the value of the argument responseFormat and must be mapped as defined in the following list: ● STRUCTURED_CONVERTED: "urn:epSOS:ps:ps:2010" ● STRUCTURED_UNCONVERTED: "urn:epSOS:ps:ps:2010" ● UNSTRUCTURED: "urn:ihe:iti:xds-sd:pdf:2008" ● ALL: null or parameter not provided by client

Operation R2D_SRC_STR_PRE and R2D_SRC_UNSTR_PRE (Search of structured Prescriptions)

This operation, as it has been designed in eHDSI, allows to retrieve both structured (ePSOS pivot) or unstructured (source coded) versions of Prescriptions. The following table shows all specifications of such operation.

Property	Value
eHDSI Service Interface	OrderService
Service Interface method	list()
Header Params	● Session token obtained by authentication method ● epSOS HCP Identity Assertion
Body Arguments	● Instance of ListOrderRequest compatible to specifications reported in table below.
Header Parameters	● [PT] X.509 NCP-B service certificate ● [ST] eHealth DSI HP Identity Assertion ● [ST] eHealth DSI Treatment Relationship ● Confirmation Assertion [O]
Return Value	● Instance of ListOrderResponse containing list of citizen prescriptions in the requested format. ● [PT] X.509 NCP-A service certificate
Error	● 4701 - No consent ● 4702 - Weak Authentication ● 4703 - Insufficient rights ● 1102 - No Data ● 4201 - Unsupported Feature

	• 4202 - Unknown Signifier • 4203 - Transcoding Error • 4204 - Unknown Filter

Allowed value for the argument instance of ListPatientRequest:

Method Arguments		
Name	**Mandatory**	**Constraint**
Patient Identifier	YES	Value MUST be equals to the identifier of the patient who performed the authentication.
epSOS CDA template qualifier	NO	Code defining the output format of requested health data. It may be an epSOS CDA common template or document coded according to the source of health data. The value of this argument is related to the value of the argument responseFormat and must be mapped as defined in the following list: • STRUCTURED_CONVERTED: "urn:epSOS:ps:ps:2010" • STRUCTURED_UNCONVERTED: "urn:epSOS:ps:ps:2010" • UNSTRUCTURED: "urn:ihe:iti:xds-sd:pdf:2008" • ALL: null or parameter not provided by client