D4.3

Specification of remote and D2D protocol and APIs for HR exchange - V3

ABSTRACT

This report describes the third 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 the internet. On the other side, the R2D protocols - consisting of the R2D Access, R2D Emergency, and R2D Backup protocols, define 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 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	27th July 2021
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, Flora Malamateniou, Aikaterini Aggeli	UPRC
	Alessio Graziani, Francesco Torelli	ENG
	Chrysostomos Symvoulidis	BYTE
Reviewers	Simone Bocca	UNITN
	Konstantinos Koutsoukos	BYTE

LOG TABLE

Version	Date	Change	Author	Partner
0.1	2021-05-05	First draft of ToC	Thanos Kiourtis	UPRC
0.2	2021-05-12	Provided the content as in D4.2	Thanos Kiourtis, Argyro Mavrogiorgou, Alessio Graziani	UPRC, ENG
0.3	2021-05-25	Split the R2D protocol into three new section	Alessio Graziani, Chrysostomos Symvoulidis	ENG, BYTE
0.4	2021-06-03	Provided related work for all the involved protocols	Thanos Kiourtis, Argyro Mavorgiorgou, Alessio Graziani, Chrysostomos Symvoulidis	UPRC, ENG, BYTE
0.5	2021-06-19	Specified new messages to be exchanged for the D2D protocol specification	Thanos Kiourtis, Aikaterini Aggeli, Argyro Mavrogiorgou	UPRC
0.6	2021-06-23	Defined new operations and updated old operations for the R2D protocols	Alessio Graziani, Chrysostomos Symvoulidis	ENG, BYTE
0.7	2021-06-29	Finalized operations and UML diagrams for all the protocols.	Thanos Kiourtis, Argyro Mavorgiorgou, Alessio Graziani, Chrysostomos	UPRC, ENG, BYTE

			Symvoulidis	
0.8	2021-07-07	Added the support of DICOM	Thanos Kiourtis, Argyro Mavorgiorgou, Alessio Graziani, Chrysostomos Symvoulidis	UPRC, ENG, BYTE
0.9	2021-07-10	Finalized and homogenized content, and sent for internal review	Thanos Kiourtis, Argyro Mavorgiorgou, Flora Malamateniou, Alessio Graziani, Chrysostomos Symvoulidis	UPRC, ENG, BYTE
1.0	2021-07-25	Provided further updates with respect to the exchanged messages for the D2D protocol, and sent for Quality Check	Thanos Kiourtis, Argyro Mavorgiorgou, Alessio Graziani	UPRC, ENG
1.1	2021-07-26	Quality Check	Argyro Mavorgiorgou	UPRC
VFinal	2021-07-27	Final revision and submission	Francesco Torelli Laura Pucci	ENG

ACRONYMS

Acronym	Term 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
HCO	Healthcare Organization
CEF	Connecting Europe Facility
eHDSI	eHealth Digital Service Infrastructure
JSON	JavaScript Object Notation
JWT	JSON Web Token
API	Application Programming Interface
FHIR	Fast Healthcare Interoperability Resources
UML	Unified Modelling Language
SQL	Structured Query Language
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

1 INTRODUCTION 1
1.1 Scope of the document 1
1.2 Intended audience 2
1.3 Structure of the document 2
1.4 Updates with respect to previous version (if any) 3
2 D2D AND R2D PROTOCOLS VISION AND COMMON PARTS 6
2.1 Conceptual D2D and R2D Protocol Data Model 7
2.2 Involved Applications 11
2.2.1 S-EHR App 11
2.2.2 HCP App 12
2.2.3 R2D Access server 12
2.2.4 S-EHR Cloud 12
3 D2D PROTOCOL: SHORT RANGE HEALTH DATA EXCHANGE 13
3.1 D2D Protocol Scope 13
3.2 Related Work 14
3.2.1 Short Range Wireless Communication 14
3.2.1.1 ANT+ 15
3.2.1.2 Bluetooth & Bluetooth Low Energy 15
3.2.1.3 EnOcean 18
3.2.1.4 Near Field Communication (NFC) 18
3.2.1.5 RFID 19
3.2.1.6 ZigBee 19
3.2.1.7 Wi-Fi Direct 20
3.2.1.8 Z-Wave 21
3.2.2 Related Work similar to the D2D approach 21
3.2.3 D2D over Bluetooth 25
3.2.3.1 Bluetooth Profiles 29
3.2.3.2 Android-side D2D over Bluetooth 34
3.2.3.3 iOS-side D2D over Bluetooth 34
3.3 D2D Protocol Overview 35
3.4 D2D Protocol Technical Specification 38
3.4.1 General Structure of Request Messages 40

3.4.2 General Structure of Response Messages 42
3.4.3 Specification of D2D Requests and Responses 43
3.4.3.1 Search all the resource types 43
3.4.3.2 Search the Resources that belong to specific categories 45
3.4.3.3 Search the Resources that belong to a single category 47
3.4.3.4 Search the Resources that belong to a single category and subcategory 48
3.4.3.5 Search the most recent Resources that belong to a single category 50
3.4.3.6 Search the most recent Resources that belong to a single category and subcategory 52
3.4.3.7 Read specific Resources of a specific Id 54
3.4.3.8 Write specific Resources 55
3.4.4 Management of Large Image Files 56
4 R2D ACCESS PROTOCOL: REMOTE HEALTH DATA EXCHANGE 58
4.1 R2D Access Related Works 59
4.1.1 eHDSI 59
4.1.2 International Patient Summary Project 60
4.1.3 Argonaut Project 61
4.1.4 IHE Project 61
4.2 R2D Access Overview 63
4.2.1 Health data completeness and asynchronous interactions 64
4.2.2 Citizen identification 67
4.3 R2D Access Specifications 72
4.3.1 URI schemas 74
4.3.2 Supported HTTP methods 75
4.3.3 Supported interactions 75
4.3.4 Supported common search parameter and search options 77
4.3.5 Search Narrowing 77
4.3.6 Interface R2DAccess 78
4.3.6.1 Search Patient 78
4.3.6.2 Search DocumentReference 79
4.3.6.3 Search DocumentManifest 82
4.3.6.4 Search Condition 85
4.3.6.5 Search AllergyIntolerance 88
4.3.6.6 Search Observation 91
4.3.6.7 Search Immunization 94

4.3.6.8 Search MedicationRequest 96
4.3.6.9 Search Procedure 99
4.3.6.10 Search Encounter 101
4.3.6.11 Search DiagnosticReport 104
4.3.6.12 Search Composition 106
4.3.6.13 Get everything of a Patient 109
4.3.6.14 Get everything of an Encounter 111
4.3.6.15 Get everything of a Composition 113
4.3.6.16 Get everything of a DiagnosticReport 114
4.3.7 Interface R2DAccessIdentification 115
4.3.8 Interface R2DAccessDICOM 116
4.3.8.1 Get a Study 117
4.3.8.2 Get a Series 118
4.3.8.3 Get an Instance 119
4.3.9 Support for transactions 120
4.3.10 Capability Statement 120
5 R2D BACKUP PROTOCOL: BACKUP HEALTH DATA ON THE S-EHR CLOUD 122
5.1 R2D Backup Protocol scope 122
5.2 Related Work 123
5.3 R2D Backup Overview 124
5.3.1 Registration and upload of health data 125
5.3.2 Download of health data 128
5.4 R2D Backup Specifications 129
5.4.1 Register to a S-EHR Cloud 129
5.4.2 Login to the S-EHR Cloud 130
5.4.3 List of available buckets 131
5.4.4 List of encrypted health records stored in a bucket 132
5.4.5 Retrieval of metadata information of an object 133
5.4.6 Download of an object from the S-EHR Cloud 134
5.4.7 Upload of a health record to the S-EHR Cloud 135
5.4.8 Download of consent for S-EHR Cloud provider 136
5.4.9 Upload of signed consent for S-EHR Cloud provider 137
5.4.10 Download of consent for trusted HCO 138
5.4.11 Upload of signed consent for trusted HCO 139

5.4.12 Download of auditing information 139
5.4.13 Withdrawal of signed consent for trusted HCO 141
5.4.14 Removal S-EHR Cloud account 142
6 R2D EMERGENCY PROTOCOL: EMERGENCY ACCESS OF HEALTH DATA FROM THE S-EHR CLOUD 143
6.1 R2D Emergency Protocol scope 143
6.2 Related Work 143
6.3 R2D Emergency Overview 146
6.3.1 R2D Emergency download of encrypted health data from a S-EHR Cloud 147
6.3.2 R2D Emergency upload of encrypted health data to a S-EHR Cloud 149
6.4 R2D Emergency Specifications 150
6.4.1 Request to access health data 151
6.4.2 List of available buckets 152
6.4.3 List of health records in a bucket 153
6.4.4 Metadata retrieval 154
6.4.5 Health record download 155
6.4.6 Health record upload 156
6.4.7 DICOM Support 157
7 CONCLUSIONS AND NEXT STEPS 159

LIST OF FIGURES

Figure 1 - Usage of the D2D and the R2D protocols
Figure 2 - Conceptual data model
Figure 3 - Conceptual data model types
Figure 4 - D2D protocol scope
Figure 5 - Bluetooth pairing process
Figure 6 - Bluetooth profiles
Figure 7 - Bluetooth serial port profile (SPP)
Figure 8 - Bluetooth personal area network (PAN) profile
Figure 9 - Bluetooth generic object exchange profile (GOEP)
Figure 10 - Bluetooth profiles comparison
Figure 11 - D2D protocol interfaces
Figure 12 - D2D protocol interactions
Figure 13 - D2D protocol security interactions
Figure 14 - D2D protocol interface operations
Figure 15 - R2D Access overview
Figure 16 - R2D Access example
Figure 17 - eHDSI system architecture
Figure 18 - Asynchronous request processing
Figure 19 - Citizen identification process
Figure 20 - R2DAccess Service interfaces
Figure 21 - The R2D Access interface
Figure 22 - R2DAccessIdentification interface
Figure 23 - DICOM support
Figure 24 - R2DAccessDICOM interface
Figure 25 - R2D Backup protocol scope
Figure 26 - R2D Backup protocol interface
Figure 27 - R2D Backup protocol interface operations
Figure 28 - R2D Backup protocol interactions with respect to the upload of encrypted health data to the S-EHR Cloud
Figure 29 - R2D Backup protocol interactions with respect to the download of encrypted health data from the S-EHR Cloud
Figure 30 - R2D Emergency protocol scope
Figure 31 - R2D Emergency protocol interface

Figure 32 - R2D Emergency protocol interface operations
Figure 33 - R2D Emergency protocol interactions with respect to the download of encrypted health data from the S-EHR Cloud to the HCP app during an emergency
Figure 34 - R2D Emergency protocol interactions with respect to the upload of encrypted health data related to the emergency to the S-EHR Cloud from the HCP app during or after an emergency
Figure 35 - Common eHDSI domain

LIST OF TABLES

Table 1 - Description of the data model classes
Table 2 - Mapping of Conceptual data model types to FHIR Resources
Table 3 - Comparison between short-range wireless communication technologies
Table 4 - Size of files that can be exchanged through the D2D protocol
Table 5 - Communication Requirements
Table 6 - Data Requirements
Table 7 - Security Requirements
Table 8 - User Interaction Requirements

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 his/her mobile device. The Citizen, using the dedicated mobile app (S-EHR app) is able to coordinate the exchange of health data between his/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 data exchange protocols defined by the InteropEHRate platform: the D2D protocol for short range distance transmission, and the R2D protocols (i.e. R2D Access, R2D Emergency, R2D Backup) 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 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 protocol 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 the 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 final version of the D2D protocol is still specified based on the Bluetooth Classic, and more specifically the Bluetooth SPP and PAN profiles. The reason for this choice is explained in Section3, 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 Access 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 the EHR of hospitals). Moreover, it is used for initial import of health data to the mobile device and for the subsequent periodic synchronization operations.

Similar to the R2D Access protocol, the R2D Backup protocol is a protocol defined by the InteropEHRate project to work over the internet. The R2D Backup protocol defines the set of operations used by the citizens in order to safely back up and store their health data on the S-EHR Cloud. In addition, it allows the citizens to gain access to the data that is stored in the S-EHR Cloud from another S-EHR mobile application.

The R2D Emergency protocol on the other side is also used for the communication with the S-EHR Cloud but from the HCP's perspective. In more detail, the R2D Emergency protocol, which similar to the R2D

Backup and R2D Access protocols makes use of the internet, allows authorized HCPs to communicate with S-EHR Cloud providers during emergency situations and gain access to the health data of the citizen in need. Not only that, with the use of the R2D Emergency protocol, an authorized HCP may upload health data related to the specific emergency on the S-EHR Cloud.

To this end, it should be also mentioned that the current document received as input the reference architecture as elaborated in deliverable D2.6 - InteropEHRate Architecture - V2 [D2.6], and the user requirements that were analysed and presented in deliverable D2.3 - User Requirements for cross-border HR integration - V2 [D2.3].

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 for 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 extend and update 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, while it also includes a short introduction to all the involved applications.
- 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 Access protocol (R2D Access), 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 includes a description of all the involved operations.
- Section 5 describes the remote health data exchange Backup protocol (R2D Backup), explaining in deep detail the purpose of the existence of the protocol, stating the research that was undertaken for concluding into the design of the protocol, and the overall description of it. It also includes a description of all the involved operations.
- Section 6 describes the remote health data exchange Emergency protocol (R2D Emergency), explaining in deep detail the purpose of the existence of the protocol, stating the research that was undertaken for concluding into the design of the protocol, and the overall description of it. It also includes a description of all the involved operations.

- Section 7 outlines the conclusions of the current document, including the updates and the future development plans for the D2D and R2D protocols.

1.4 Updates with respect to previous version (if any)

With regards to the previous version of the deliverable (D4.2 Specification of remote and D2D protocol and APIs for HR exchange V2 [D4.2]), the following changes have been performed for each separate section of the current document.

Section 1 - Introduction:

- The introduction has been updated including a new description for the D2D Protocol, the R2D Access Protocol, the R2D Backup Protocol, and the R2D Emergency Protocol, since D4.3 has been structured in a different way than the previous versions of the deliverable.
- The Structure of the document has been updated as well, based on the new content.

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

- The section provides an overall view of D2D and all R2D protocols. This section has been enhanced to better show the relationships between all the protocols, their similarities and their differences. To ease the understanding of all the operations of the protocols, this section also contains a first initial mapping between classes defined in the common conceptual data model and FHIR resources.
- This section has been also redesigned to contain a short description of the applications which are involved in the context of the D2D and R2D protocols, so that these descriptions are not duplicated within this document.

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

- Additional related work has been provided with similar research works which aim on exchanging health data both in short-range and in long-range distances in order to better identify the importance and the benefits of the D2D protocol
- A comparative study has been provided that was undertaken for the bluetooth profiles that can be supported by the D2D protocol, for the main market operating systems. This has been provided in order to not only conclude that Bluetooth is the best solution for the D2D protocol, but also to identify the most efficient Bluetooth profile to be used, for accelerating and improving the overall performance of the protocol.
- The D2D protocol operations, as well as the interfaces providing these operations, have been redesigned since the D2D protocol has been redesigned in order for the HCP app to be able to send requests in the form of Bluetooth messages, and the S-EHR app to reply with specific responses. All these messages are clearly stated in the technical specification of the D2D protocol, categorized into four (4) main categories that specify the overall protocol. The reason for this update is mainly to make the D2D protocol more generic and independent from the scenario in which it is involved, and to better align it with the R2D Access protocol so as to be more easily understandable by the stakeholders.
- It has been added a specification on how the DICOM images (i.e., large image files) can be supported in the case of the D2D protocol in order to specify the way that additional types of data can be exchanged over Bluetooth

Section 4 - R2D Access Protocol: Remote Health Data Exchange:

- The section has been split from the ones describing the other two protocols of the R2D family (i.e. R2D Backup and R2D Emergency).
- Also, the section has been restructured to provide a first technical overview before introducing technical specification of the R2D Access protocol. This technical overview introduces two new topics that represents the most important changes between V2 and V3:
    - introduction of the asynchronous pattern for processing R2D Access requests.
    - Addition of new operations to support the matching of the Citizen's eIDAS identity with the patient identity of the HCO.
- Addition of specifications regarding the search narrowing applied to resources status, to retrieve only committed health data from an R2D Access server.
- Addition of new instance level operations to simplify retrieval of large amount of data in a single operation:
    - Patient$everything;
    - Encounter$everything;
    - DiagnosticReport$everything;
    - Composition$document.
- Addition of the support for the following resources:
    - Condition
    - AllergyIntolerance
    - Immunization
    - Procedure
    - Composition
    - DocumentReference
    - DocumentManifest
- The support for parameters _include, _revinclude has been removed and it has been replaced by the aforementioned operations (operations are simpler to use than _include and _revinclude parameters).
- Although the International Patient Summary defines the FHIR structure of a Patient Summary document, it does not constrain how to store it, therefore different repositories could use two different approaches. So retrieval of the Patient Summary has been changed to supported searches on the Composition resource (in case it is stored as a Composition according to IPS) or in the DocumentReference resource (in case it is stored as a Bundle or as a PDF document).
- It has added the support for the interface R2DAccessDICOM, to exploit the standard WADO-RS to download DICOM images not embedded within the FHIR resources, but just referred to by them.

Section 5 - R2D Backup Protocol: Backup Health Data on the S-EHR Cloud

- This is a totally new section which derived after splitting the R2D Backup protocol from the R2D generic section. The rationale behind the creation of a section dedicated to the R2D Backup protocol is to separate the R2D Backup protocol from the R2D Access and provide its specification in a way that highlights the differences and purpose between the R2D Backup and R2D Access protocols.
- It has been provided a clear definition of the R2D Backup protocol
- The following subsections have been added:

- R2D Backup Protocol's scope
- Related Work
- R2D Backup Protocol's technical description including a detailed overview of its usage and interactions
- R2D Backup Protocol's technical specification, containing the basic operations in the form of exposed interfaces

Section 6 - R2D Emergency Protocol: Emergency access of Health Data from the S-EHR Cloud

- This is a totally new section which derived after splitting the R2D Emergency protocol from the R2D generic section. The rationale behind the creation of a section dedicated to the R2D Emergency protocol is to separate the R2D Emergency protocol from the R2D Access and provide its specification in a way that highlights the differences and purpose between the R2D Emergency and R2D Access protocols.
- It has been provided a clear definition of the R2D Emergency protocol
- The following subsections have been added:
  - R2D Emergency Protocol's scope
  - Related Work
  - R2D Emergency Protocol's technical description including a detailed overview of its usage and interactions
  - R2D Emergency Protocol's technical specification, containing the basic operations in the form of exposed interfaces
- Similarly to R2DAccess, now also R2DEmergency exploits the new interface R2DAccessDICOM.

Section 7 - Conclusion and Next Steps:

- The conclusions and next steps have been updated based on the current specification and our future plans

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) [D2.6], 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 InteropEHRate protocols: R2D and D2D, where R2D is a family of protocols Internet-based (R2D Access, R2D Backup, R2D Emergency) while D2D is a unique protocol Bluetooth-based.

The operations depicted in Figure 1, do not happen simultaneously. Firstly, the citizen is importing (using R2D Access or D2D) his/her health records stored within the EHR of a healthcare organization of his/her Country (e.g. a hospital) to his/her smart mobile device. Only after this operation has been executed, the citizen is able to transfer his/her health records (using D2D) to an HCP in Country B for a medical visit abroad. If, during this medical visit, new health data are produced they can be immediately transferred to the citizen's smartphone using D2D, otherwise the citizen will import them later using R2D Access.

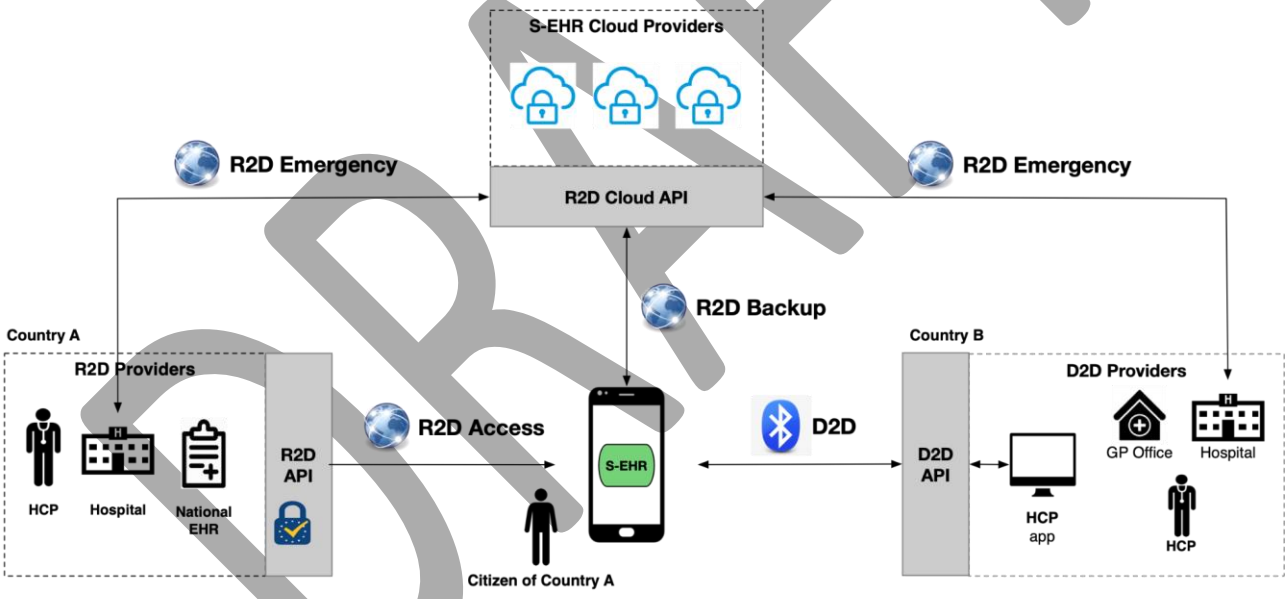

Figure 1 - Usage of the D2D and the R2D protocols

It is very likely that after having imported health data from several healthcare organizations, the citizen has created a unique aggregated health data repository on his/her mobile phone. And this is where the R2D Backup enters the scene, supporting the citizen in periodically saving to the S-EHR Cloud the local health data repository so that the citizen may restore it in case the mobile phone gets damaged, lost or stolen. Finally, data stored in the S-EHR Cloud, may be accessed using the R2D Emergency protocol by healthcare organizations during an emergency situation (abroad) if the citizen is unconscious.

Except for the emergency scenario just mentioned, all the incoming and outgoing operations are coordinated by the citizen using the S-EHR app: the citizen owns the full control of his/her health data. Data importing through the R2D Access 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 his/her demographic details, and the citizen has given the needed consent to the healthcare organization of the HCP.

D2D and all the R2D protocols deal with the same types of health data, all the 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 description of the common data model, to facilitate the understanding of the operations provided by the protocols.

2.1 Conceptual D2D and R2D Protocol Data Model

The following UML class diagram (Figure 2) depicts this data model, showing the main classes and their relationships (each class is fully described in the specific table):

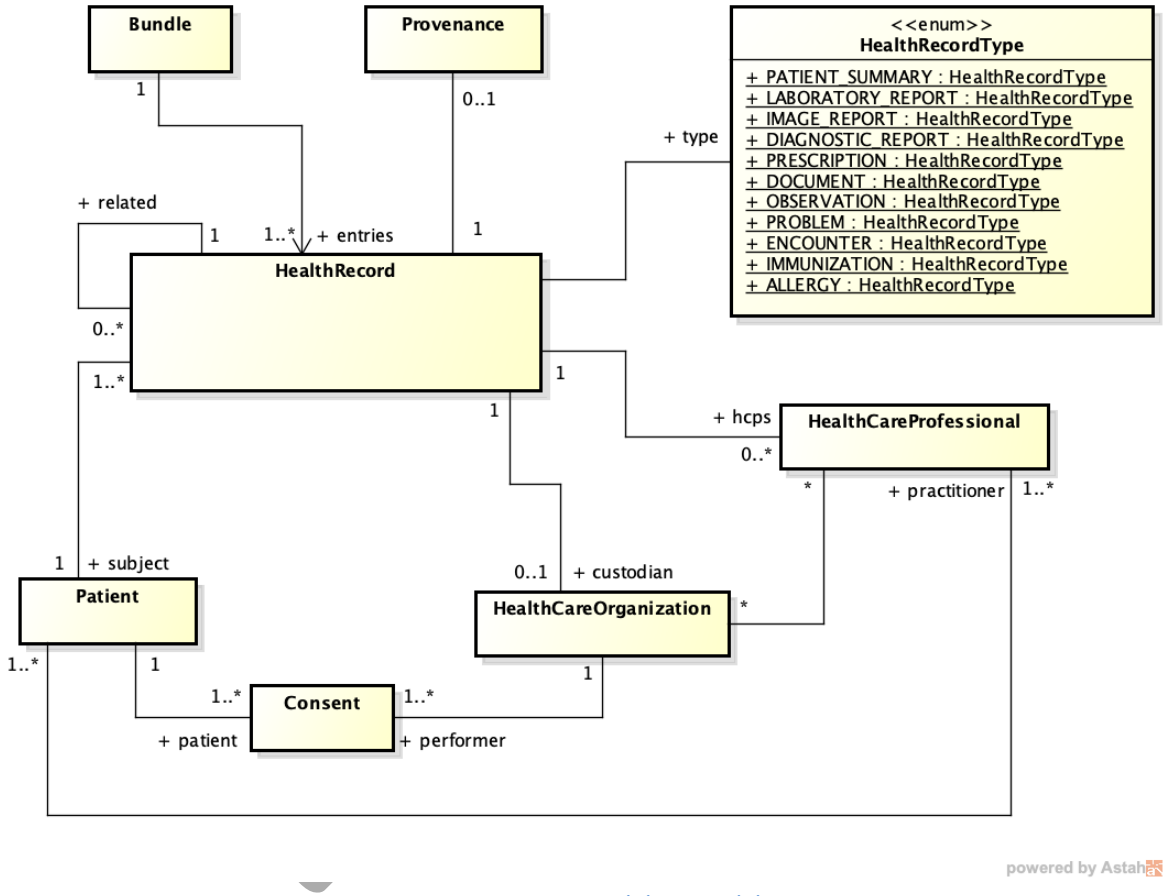

Figure 2 - Conceptual data model

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 (simple or complex, structured and unstructured) of the patient (Patient Summary, laboratory reports, 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. ● *Provenance*: optional relationship relating an instance of HealthRecord to an instance of Provenance. This relationship is used when provenance information is added to an instance of HealthRecord.
HealthRecordType	An enumeration representing the set of defined types of health data. Its values are: ● PATIENT_SUMMARY ● LABORATORY_REPORT ● IMAGE_REPORT ● DIAGNOSTIC_REPORT ● PRESCRIPTION ● DOCUMENT ● OBSERVATION ● PROBLEM ● ENCOUNTER ● IMMUNIZATION ● ALLERGY 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 he/ 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 a 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 the HealthRecord class. Relationships: • HealthRecords: this mandatory relationship, links a Bundle to the set of contained HealthRecord.
Provenance	Provenance represents information about the producer of an instance of health data.

Table 1 - Description of the data model classes

Providing a concrete representation of this abstract data model is not an objective of this deliverable, (it is an objective of deliverable [D2.9]), but to better understand the operations exposed by protocols it is important to anticipate that the whole conceptual data model has been implemented using FHIR data model release 4 [FHIR R4].

Health data types defined in the conceptual data model (Figure 3) (values of the enumeration HealthRecordType) have been mapped to FHIR R4 resources as specified in the following table (Table 2):

Conceptual data model type	Mapped FHIR Resource
DIAGNOSTIC_REPORT	DiagnosticReport
PRESCRIPTION	MedicationRequest
DOCUMENT	• DocumentReference • DocumentManifest
OBSERVATION	Observation
PROBLEM	Condition
ENCOUNTER	Encounter
IMMUNIZATION	Immunization
ALLERGY	AllergyIntolerance
PATIENT_SUMMARY	Composition (according to International Patient Summary specifications [FHIR IPS SPECS])
LABORATORY_REPORT	• DocumentReference • DiagnosticReport
IMAGE_REPORT	• DocumentReference • DiagnosticReport

Table 2 - Mapping of Conceptual data model types to FHIR Resources

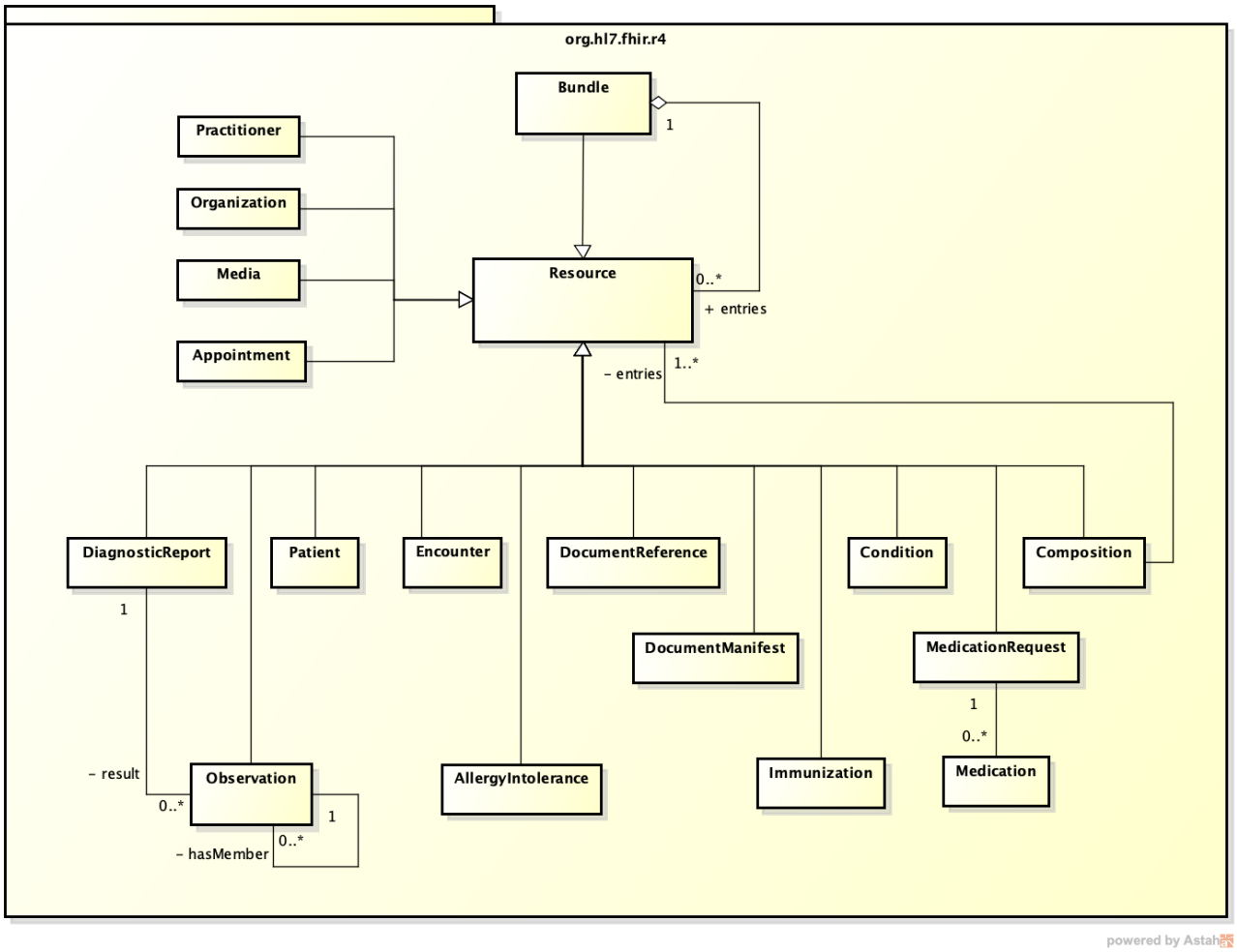

Figure 3 - Conceptual data model types

2.2 Involved Applications

This section provides a brief description of the applications that are involved in the use of D2D and R2D Protocols. In more detail, the applications are involved as follows:

- D2D Protocol: The D2D protocol involves the S-EHR app and the HCP app
- R2D Protocols:
    - R2D Access Protocol: The R2D Access protocol involves the S-EHR app and the EHR Provider (Extended NCP, National or Local EHR through the R2D Access server)
    - R2D Backup Protocol: The R2D Backup protocol involves the S-EHR app and the S-EHR Cloud
    - R2D Emergency Protocol: The R2D Emergency protocol involves the HCP app and the S-EHR Cloud

2.2.1 S-EHR App

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 [D3.4] and that supports the InteropEHRate protocols defined in the current deliverable and in [D4.8]. Different vendors may develop different S-EHR 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 [D2.6].

2.2.2 HCP App

An HCP app is a software application designed to provide medical staff with the ability to access and operate upon citizens' 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 identification data and request healthcare data 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 [D2.6].

2.2.3 R2D Access server

The R2D Access server is a server application deployed by HCOs that want to provide an access point to citizens that want to retrieve their health data in a machine readable format. R2D Access server must be compliant to the specifications defined in this deliverable, providing the interfaces defined in [D2.6].

2.2.4 S-EHR Cloud

A S-EHR Cloud is a service used by citizens for the safe storage of their health data. The citizens can utilize this storage service through their S-EHR Application in order to backup their health data. In addition, the S-EHR Cloud service can be used by authorized HCPs in order to access the health data of a citizen in need during an emergency, and to upload information related to that emergency (i.e. Patient Discharge Report) once the emergency is over. More details on what is supported by a S-EHR Cloud can be found in [D6.7].

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 an HCP app in short-range distance (i.e. in the context of a distance of up to 100m long) using Bluetooth, 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 specify the D2D protocol.

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 4).

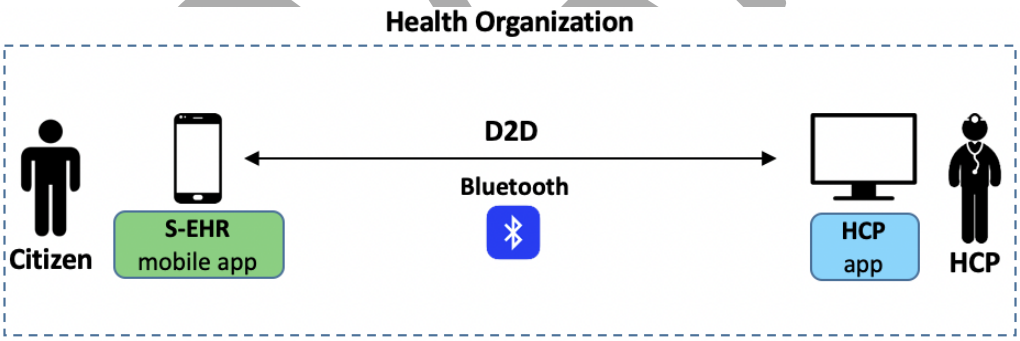

Figure 4 - D2D protocol scope

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 to be offered by a healthcare organization to request health data contained in the S-EHR app, as well as the Bluetooth services to be offered by the S-EHR app for receiving requests from the HCP app of the

Healthcare Organization Information System and provide specific responses. The D2D protocol will also exploit the D2D Security protocol 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 communication 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.

3.2.1.1 *ANT+*

ANT and ANT+ [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.

3.2.1.2 *Bluetooth & Bluetooth Low Energy*

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 a 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 colocated 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.

3.2.1.3 *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 [ENOCEAN] 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).

3.2.1.4 *Near Field Communication (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.

3.2.1.5 *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.

3.2.1.6 *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.

3.2.1.7 *Wi-Fi Direct*

Wi-Fi Direct [WIFIDIRECT], 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

3.2.1.8 *Z-Wave*

Z-Wave [ZWAVE] 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 Related Work similar to the D2D approach

Current research promises to solve the challenge of maintaining and facilitating the exchange, sharing and analysis of healthcare information. Nevertheless, when this information must be shared between healthcare ecosystem entities, this happens by specifically designed medical applications, protocols or even by the patients themselves, who often bring their files from appointment to appointment. In the current deliverable, the D2D Protocol is being specified, in the form of a secure communication protocol for exchanging messages and healthcare-related data between two nearby smart devices, adopting short range communication technologies, and in particular Bluetooth v4.0. Shortly, the D2D protocol is being specified

as an open specification protocol that can give the ability to EU citizens to exchange health information stored in their smart devices directly with Healthcare Institutions. This health information is structured in a common format with respect to the widely adopted HL7 FHIR standard, whereas the D2D protocol can check the conformance of the exchanged information based on specifically parameterized HL7 FHIR interoperability profiles. Furthermore, the D2D protocol provides a fully encrypted communication channel with easily usable symmetric key authentication, supporting data encryption and decryption mechanisms, without allowing any third-party to access this communication. Hence, through adopting the D2D protocol, the idea of securely exchanging HL7 FHIR structured health information even in cases where internet connection is not available, can become a reality.

Currently, the European Commission is presenting the need for a secure system that will enable citizens to access their electronic health records in all Member States of the European Union by 2021 [Marco][Europa][eHealth]. The aim is to facilitate European citizens' access to their health data wherever they are in the EU, to have the ability to easily exchange it, to improve medical monitoring and to make emergency care more efficient. In France, the Shared Medical Records (SMR – Dossier Medical Partagé in French) [SMR] was officially launched in November 2018. This can be best described as an electronic health record that stores, centralizes, and secures all patient health information (reimbursements, pathologies, treatments, consultation and hospitalization reports, examination results, etc.). Furthermore, the northern European countries (Finland, Sweden, Denmark, and Estonia) are pioneers in the digitization of their administrations [Europa]. Online services, connected identity cards, nearly all citizens now use the Internet for their administrative procedures. The same applies to health services. In Denmark, the Medcom [MedCom] initiative was launched in 1994 as a national health data network, with an Internet portal that collects and distributes medical data between health professionals and citizens. Sweden adopted a national e-health strategy in 2006. The Swedish Medical Record (NPÖ) [NPO] is accessible to authorised personnel, and any existing computer system can be connected to it. In southern Europe, in Spain, the Andalusian region has launched the Diraya program [Diraya], which aims at ensuring the continuity of care by harmonizing information so that it is accessible to everyone, patients and doctors alike. A single medical file has been set up and the information is centralized on specialized websites. It is undeniable that a European system of health data exchange would have many advantages since it would make it easier for European citizens to move from one country to another without having to worry about their current medical prescriptions or treatments. In addition, in the context of emergency care abroad, the patient file would be accessible more quickly. Hence, citizens would be treated more quickly and securely, and with the appropriate care.

Nevertheless, there exist several competitor technologies and research projects, with a similar purpose as the D2D protocol, aiming towards the Health Information Exchange (HIE) initiative. Among the research projects - which are not however using short-range wireless communication technologies to exchange health information, there exist several solutions. The Direct Secure Messaging [Direct] is a digital messaging tool which is used in healthcare for such communication as referrals, admissions, discharges, and transfers, pharmacy prescriptions, automated push-event notifications, and even messaging directly with patients. Direct can be used with several different interfaces such as an email client, healthcare IT portals, mobile devices, or an automated date delivery feed. Depending on the intended workflow, Direct can integrate with healthcare IT systems in multiple ways. To participate, both sender and receiver need a specific Direct email address. Carequality [Carequality] provides a national-level, consensus-built, common interoperability framework to enable data exchange between and among health data sharing networks. CommonWell Health Alliance members and service providers have created and deployed a vendor-neutral

platform for health data exchange. Major EHR vendors such as Cerner and Greenway Health are CommonWell members but also can exchange data through Carequality. In addition to hospitals and physicians, networks have been made for radiologists for images, as well as ambulatory care centers, home health, laboratories, and post-acute care providers. Capabilities include sharing Consolidated-Clinical Document Architecture (CCDA) documents. Those who belong to a particular network can query records from various providers on the other networks as part of the Carequality framework– regardless of geography. With cloud fax [CloudFax], digital documents and reports can be efficiently routed into folders and organized in a larger document solution or in your own user interface without the need to touch physical paper. Cloud fax can plan an integral role in achieving functional interoperability, both among highly connected healthcare providers and those that may lag on the technology front. KONFIDO [KONFIDO] is a research project aiming to develop a holistic paradigm for secure cross-border health data exchange. It builds its solution upon existing/evolving European frameworks, such as OpenNCP [OpenNCP], the open-source NCP reference software implementation part of project epSOS [epSOS], and eIDAS [eIDAS]. Another research is [Gavrilov] where the authors propose a model for an Electronic Health Record (EHR) for Health Data Exchange based on a SaaS (Software as a service) service model developed on top of cloud computing technology. In the proposed model EHRs are stored in the cloud and can be accessed through a web portal or background web services through SOA. The system is using Health Level Seven International (HL7) standard and can store information on: healthcare processes, resources, users, and authorization, also capable of recording of all data on the patient history, diagnostic test data, therapy, decisions request for further investigations, and treatment. Furthermore, in [Masud] the authors implemented an anonymous, query-based, on the fly secure data exchange protocol between two clouds for collaborative cloud-based healthcare environments. It used pairing-based cryptography without the use of a public key. The healthcare service providers are helped by the cloud based big data frameworks. Each cloud source acts autonomously and creates an on-the-fly data exchange session to exchange data with another cloud. The frameworks contain a platform for exchanging medical data in multiple clouds. The healthcare providers use the cloud based big data frameworks to store, manage, share, and analyze health data. For the exchange of the data each cloud determines the acquaintance in a pair wise fashion with the other clouds. Firstly, a secret session key is randomly generated by each cloud for each data sharing session. After that, a verification is conducted to prove that the protocol prevents masquerade, replay and man in the middle attacks. As soon as this process is terminated, the exchange of health data may begin.

As for the research projects aiming in short-range distance communication protocols for exchanging healthcare data, in [Basjaruddin] the authors developed a medical record system based on Near Field Communication (NFC). NFC is a short-range wireless technology consisting of a set of communication protocols. It is used for contactless small amounts of data exchange. With NFC technology, mobile phones can store data securely as well as send data to other mobile phones which are equipped with NFC without the use of the internet. The authors considered the average level of Internet service availability in hospitals, which is not high, so they wanted to exchange healthcare data offline in a secure way. The authors developed an electronic medical record (EMR) application where it is used by doctors and patients. The patients use the application only for reading the content while the doctors can view and change the EMR content, even add a new patient EMR. Firstly, both must register on the application. The first time that a doctor examines a new patient, the doctor has to tap between his/her phone and the patient's phone to get the patient's EMR. After that, the doctor can send the latest EMR to the patient application through NFC and the EMR is stored in the patient's phone. The patient can visit other doctors and send his/her EMR to the doctor's phone using NFC in the same way. Another technology that can be used towards the Health

Information Exchange initiative is Wi-Fi Direct, also known as Wi-Fi P2P. The authors in [Khan] performed research in Wi-Fi Direct in which WPA2, devices such as smartphones, laptops etc. can establish a direct Wi-Fi connection without using a wireless access point. Traffic overload can be caused in the communication through the wireless access point, so Wi-Fi Direct can be the solution to this problem. In the same context, the authors in [Jin] also did research in Wi-Fi Direct and they proposed a real-time data transmission system by using Wi-Fi Direct for the transfer of healthcare data. Generally, Wi-Fi Direct networks include the Wi-Fi Direct owner and the Wi-Fi Direct Clients. The device that implements the wireless access point is called Wi-Fi Direct owner and all the other devices are called Wi-Fi Direct Clients and act like clients. A transmitter requests to establish a Wi-Fi Direct connection to a receiver. If the receiver confirms the request, a secure communication is established, and the data exchange may initiate. The authors evaluated the performance of Wi-Fi Direct using a test bed and compared it to Wi-Fi. The results of the research showed that the healthcare data are transported with greater reliability through Wi-Fi Direct than through Wi-Fi, so they recommended the Wi-Fi Direct for health data transfer.

With regards to the research conducted using a mix of distance communication protocols for exchanging data of any type (i.e., not only healthcare data), the authors in [Mockel] present a Bluetooth scatternet protocol (SNP) that provides the user with a serial link to all connected members in a transparent wireless Bluetooth (BT) network. The authors show how their software layer simplifies a variety of tasks like the synchronization of central pattern generator controllers for actuators, collecting sensory data and building modular robot structures. The whole BT software stack including their newly proposed scatternet layer is implemented on a single Bluetooth and memory chip. In [Qiu] the authors propose a secure data storage and sharing method consisting of a selective encryption algorithm combined with fragmentation and dispersion to protect the data safety and privacy even when both transmission media (e.g., cloud servers) and keys are compromised. This method is based on a user-centric design that protects the data on a trusted device such as the end-users' smartphone and lets the end-user control the access for data sharing. Furthermore, similarly to the previous research, the authors in [Chong] describe an `image beacon' system that is capable of broadcasting colour images over a very long period (years, as opposed to days or weeks) using a set of cheap, low-power, memory-constrained Bluetooth Low Energy (BLE) beacon devices. They have designed an image processing pipeline that considers the background and foreground information of an image and then applies an adaptive encoding method which prioritizes more important regions of an image during encoding, to achieve the best quality image under a very strict size limit. To this end, the authors in [Chung] proposed a data sharing method among multi smart devices at close range using inaudible frequencies and Wi-Fi. Most of the near data sharing methods are unable to be operated using different operating systems. This research tried to correct this problem by using the inner speaker and the microphone of smart devices. The transmitter generates inner signals consisting of inaudible sound. The receivers receive the shared data from the transmitter via Wi-Fi. The proposed method was compared with the Bump application and indeed the proposed method is more useful than Bump. More specifically, with the use of inaudible frequencies the transmitter generates an inner signal from an inner speaker and sends the data and its current GPS information to the sharing server. The receivers, which are located near the transmitter, receive the sound-data through their microphone. In the case that they understand that the sound they received is the data-sharing signal, they also send their current GPS information to the server. Finally, they download the shared data from the server.

However, among the previous implementations, some of them can lead to a time-consuming process which can result in inaccurate information, with increased inefficiencies, leading to care of low value. Among the problems that the Health Information Exchange (HIE) initiative is facing are (i) that third party health data

cannot be accessed without internet connection - which cannot be always available, (ii) that there exist crucial delays in accessing current citizens' data, while (iii) a major obstacle is that direct exchange of health information can only happen among Healthcare Institutions, without the active participation or involvement of the current data owners (i.e., citizens). Moreover, there exist several smart devices' applications that are offering the ability to exchange health information between citizens and healthcare practitioners, which are however using vendor-specific and non-interoperable protocols, without respecting any common data representation or terminology structures. Finally, the usage of multiple credentials to authenticate citizens and Healthcare Institutions - even in the same countries, could also be a major barrier towards this objective. Among the above difficulties, the D2D protocol is being specified to solve each one of these issues.

3.2.3 D2D over Bluetooth

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 D2.3 deliverable [D2.3], 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	Android: Applicable iOS: Needs certification under MFI program	Android: Applicable iOS: GATT-based API	Android: Applicable iOS: Applicable (data transfer by NFC packets to

	PC			be understood)
Ease of Use	High	High	High	High
Power Consumption	High	Medium	Low	Low

Table 3 - 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 3, as well as the needs and the criteria that were set, the two most likely candidate 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 4, 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 4 - Size of files that can 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.

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 5).

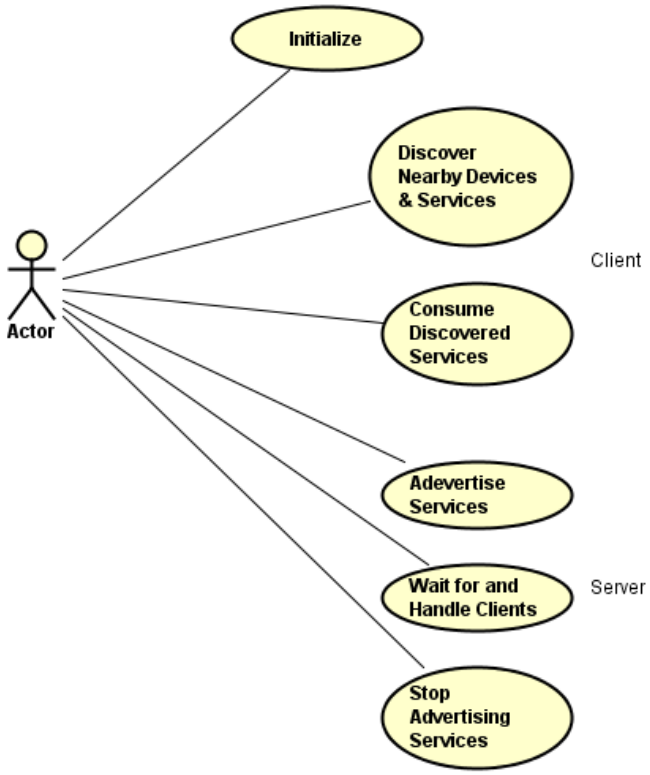

Figure 5 - Bluetooth pairing process

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.

3.2.3.1 *Bluetooth Profiles*

In the case of Bluetooth, a device should understand specific Bluetooth profiles, in the form of definitions of specific applications which have a predefined nature that Bluetooth devices use to communicate with each other. In these profiles it can be found the settings to change and to manage the overall communication from the start. Through using Bluetooth profile, it saves time for transmitting the parameters anew before the bi-directional link becomes effective. As a result, for two devices to communicate to complete a specific task, both devices must use a common profile. As depicted in Figure 6, there exist several Bluetooth Profiles, according to the task that must be performed and the domain in which they should be applied. These domains are categorized into five (5) main categories, with regards to the Medical domain (i.e., Medical Group) that deals with medical devices supporting Bluetooth connection, Serial Port Profile domain (i.e., SPP Group) that consists of devices that emulate serial cable data transfer, Telephony domain (i.e., Telephony Group) that deals with smart Bluetooth connected devices that support mobile phones and communication devices, Personal Access Network domain (i.e., PAN Group) consisting of devices that have the ability to create local network groups to communicate over Bluetooth, and Audio/ Video domain (i.e., A/V Group) that supports Bluetooth devices that deal with audio and video files, which in most cases are of large data size and complexity. Some of these profiles are depicted below: Advanced Audio Distribution Profile (A2DP), Audio/Video Remote Control Profile (AVRCP), Common ISDN Access Profile (CIP), General Audio/Video Distribution Profile (GAVDP), Generic Object Exchange Profile (GOEP), Hard Copy Cable Replacement Profile (HCRP), Human Interface Device Profile (HID), Personal Area Networking Profile (PAN), Phone Book Access Profile (PBAP), Serial Port Profile (SPP), Service Discovery Profile (SDAP), SIM Access Profile (SAP, SIM), Video Distribution Profile (VDP), Wireless Application Protocol Bearer (WAPB), Basic Imaging Profile (BIP), Basic Printing Profile (BPP), Cordless Telephony Profile (CTP), Device ID Profile (DID), Dial-up Networking Profile (DUN), Fax Profile (FAX), File Transfer Profile (FTP), Generic Access Profile (GAP), Hands-Free Profile (HFP), Headset Profile (HSP), Intercom Profile (ICP), Object Push Profile (OPP), or Synchronization Profile (SYNCH). Among this extensive list of Bluetooth profiles, since the D2D protocol is being used among devices running the most commonly installed Operating Systems (OS) (Android, Windows and iOS/iPadOS Operating Systems), the three best candidates are the Serial Port Profile (SPP), Personal Area Networking Profile (PAN), and the Generic Object Exchange Profile (GOEP) which are designed to be supported by the latter vendors.

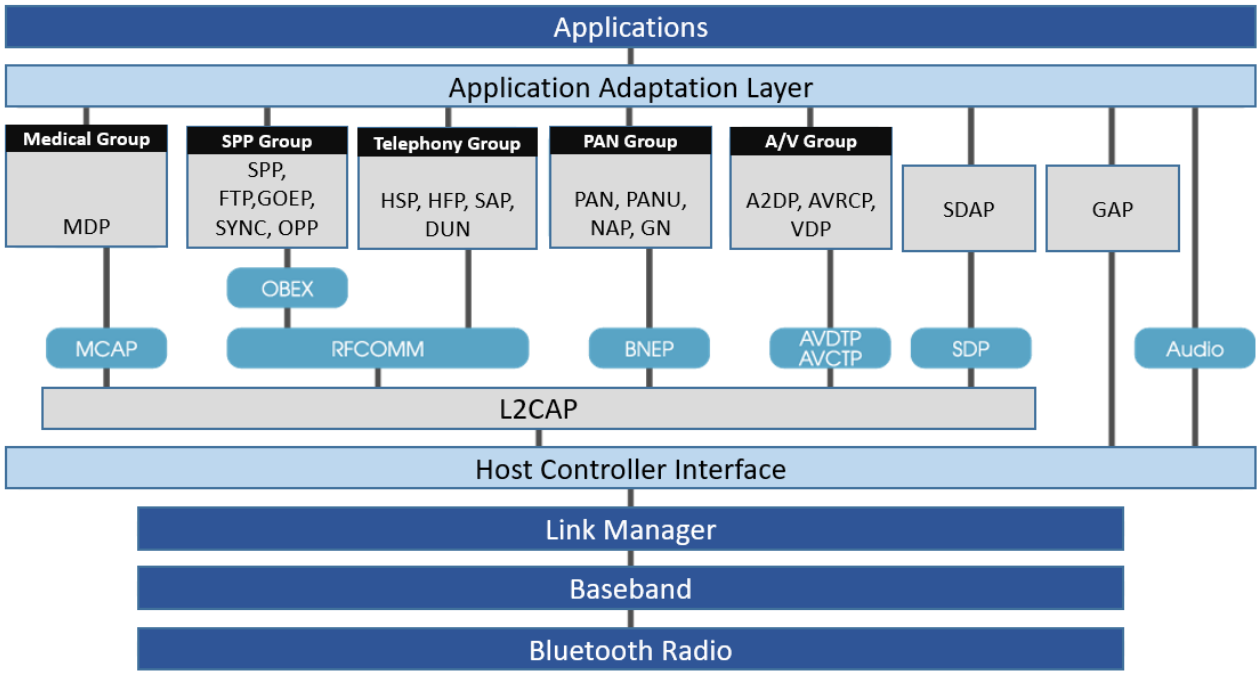

Figure 6 - Bluetooth profiles

3.2.3.1.1 Bluetooth Serial Port Profile

The Bluetooth SPP profile [SPP] defines the requirements for Bluetooth devices that are necessary for creating emulated serial cable connections using the RFCOMM protocol between the involved devices. The requirements can be described as services which are provided to the involved applications, as well as through the definition of the features and processes that are needed for communication among the interacting devices.

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

- Initiator: it initiates a connection to another device.
- Acceptor: it waits for another device to initiate the connection.

Figure 7 shows the protocols and entities used in the Bluetooth SPP profile. Shortly, the Baseband, LMP and L2CAP are the OSI layer of the Bluetooth protocols. RFCOMM is the Bluetooth adaptation of GSM TS 07.10, providing the needed serial port emulation, while SDP is the Bluetooth Service Discovery Protocol. 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.

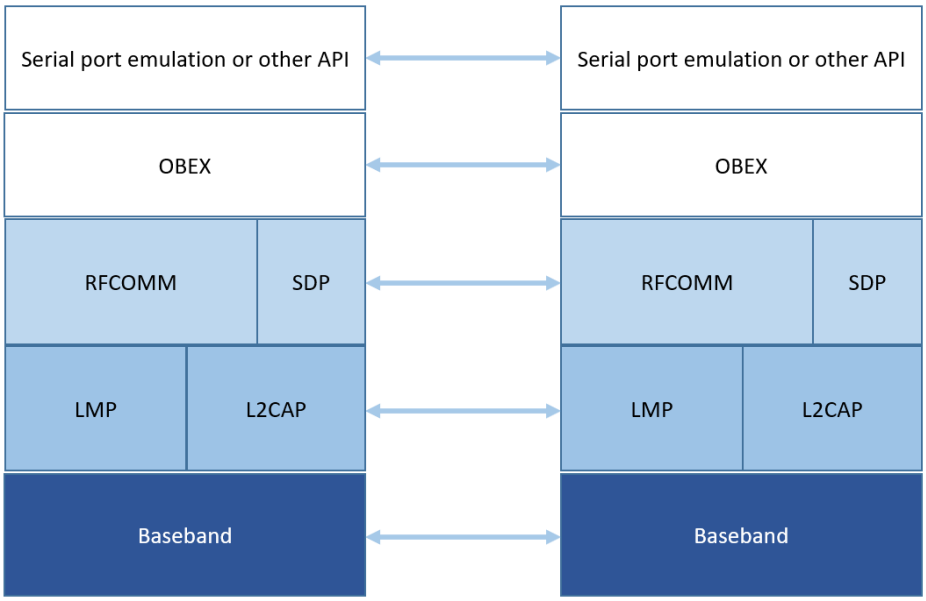

Figure 7 - Bluetooth serial port profile (SPP)

3.2.3.1.2 Bluetooth Personal Area Network (PAN) Profile

The Bluetooth PAN profile [PAN] describes the way that two or more Bluetooth devices can create an ad-hoc network and how it can be re-used for accessing 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 case has unique network architecture and requirements, creating multiple combinations of a PAN profile.

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 for forwarding Ethernet packets to each of the connected Bluetooth devices, speaking about the PAN users, as needed.
- PAN User (PANU) is a Bluetooth device that can use either the NAP or the GN service. PANU supports the client role for both the NAP and GN roles.

Figure 8 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. As in the previous case, SDP is the Bluetooth Service Discovery Protocol, while the Management Entity (ME) is the entity that coordinates the procedures for the initialization, parameterization, and connection management.

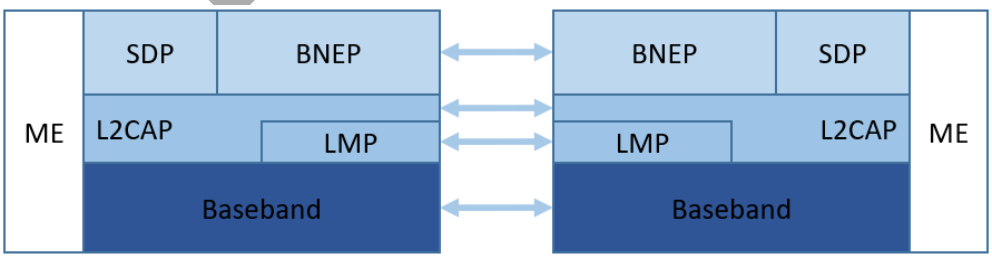

Figure 8 - Bluetooth personal area network (PAN) profile

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.

3.2.3.1.3 Generic Object Exchange Profile (GOEP)

The Bluetooth GOEP profile [GOEP] describes the protocols and procedures that the involved applications must use, providing the usage models for facilitating the overall object exchange capabilities. Such models can be a Synchronization, File Transfer, or Object Push model. In the Bluetooth GOEP profile, the following roles are defined:

- Server: it provides an object exchange server to and from which data objects have the ability to be pushed and pulled, respectively.
- Client: it can push or/and pull data object(s) to and from the Server

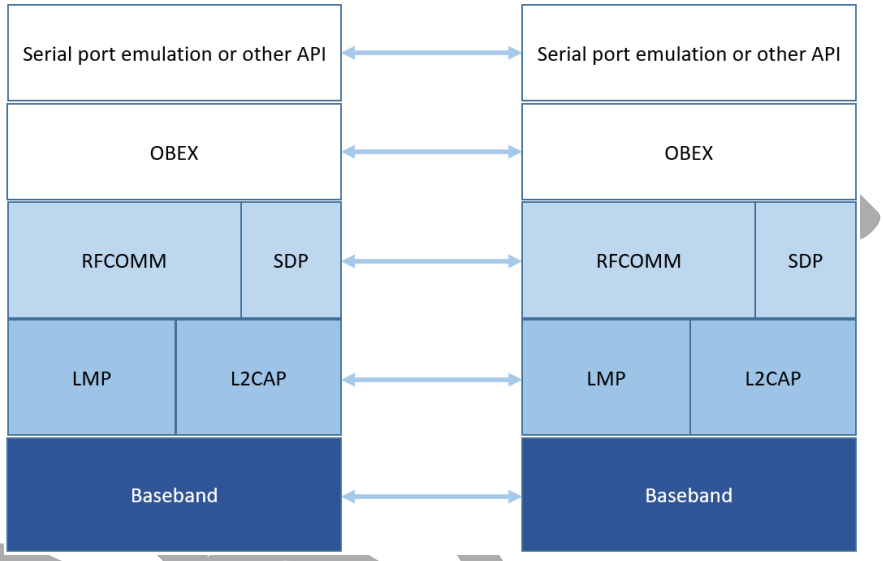

Figure 9 - Bluetooth generic object exchange profile (GOEP)

Figure 9 shows the protocols and entities used in the Bluetooth GOEP profile. The Baseband, LMP and L2CAP are the OSI layer 1 and 2 Bluetooth protocols. RFCOMM is the Bluetooth adaptation of GSM TS 07.10. SDP is the Bluetooth Service Discovery Protocol, while OBEX is the Bluetooth adaptation of IrOBEX. The Application Client layer gives the ability to send and retrieve data objects from the side of the Server using the OBEX operations. The application Server is offering storage capabilities for the data to and from which the data object can be sent or retrieved.

3.2.3.1.4 Bluetooth Profiles Comparison & Evaluation

To compare the Bluetooth SPP profile, the Bluetooth PAN profile, and the Bluetooth GOEP profile, the following evaluation metrics were considered for the three profiles, for their further comparison with regards to the purposes of the D2D protocol.

- **Transmission time**: This metric refers to the transmission time (transfer rate), in which a data exchange process takes place with each different Bluetooth profile.
- **Power Consumption**: This metric refers to the power that the devices consume when using the Bluetooth profiles.

- **Probability of packet flush**: A specific flush timer is initiated when a packet enters the transmit buffer of the controller. In the case of timeout, the packet is flushed. Each data packet must have a specific number of slots in order to have a successful data exchange. Generally, the Bluetooth baseband uses an Automatic Repeat Request (ARQ) in order to send again baseband packets that contain errors. ARQ retransmissions can happen for a specific number of times, denoted by the number of transmissions, before a timeout occurs.
- **Probability of L2CAP retransmission**: A L2CAP retransmission happens in the case that there is a timeout in the retransmission timer. Such a thing can happen if the original packet or its acknowledgment is flushed.
- **Probability of packet loss**: Packets which are not retransmitted by L2CAP are lost in the case of flushing. This can happen to all streaming packets and packets that use the Bluetooth SPP and Bluetooth GOEP profiles. Hence, for these profiles the loss probabilities are equal to the flushing probabilities, while for the Bluetooth PAN profile, the loss probabilities are the L2CAP retransmission failure probabilities.

Based on the aforementioned metrics, all the three different Bluetooth profiles were evaluated under the same conditions, while their evaluation results are being depicted in Figure 10.

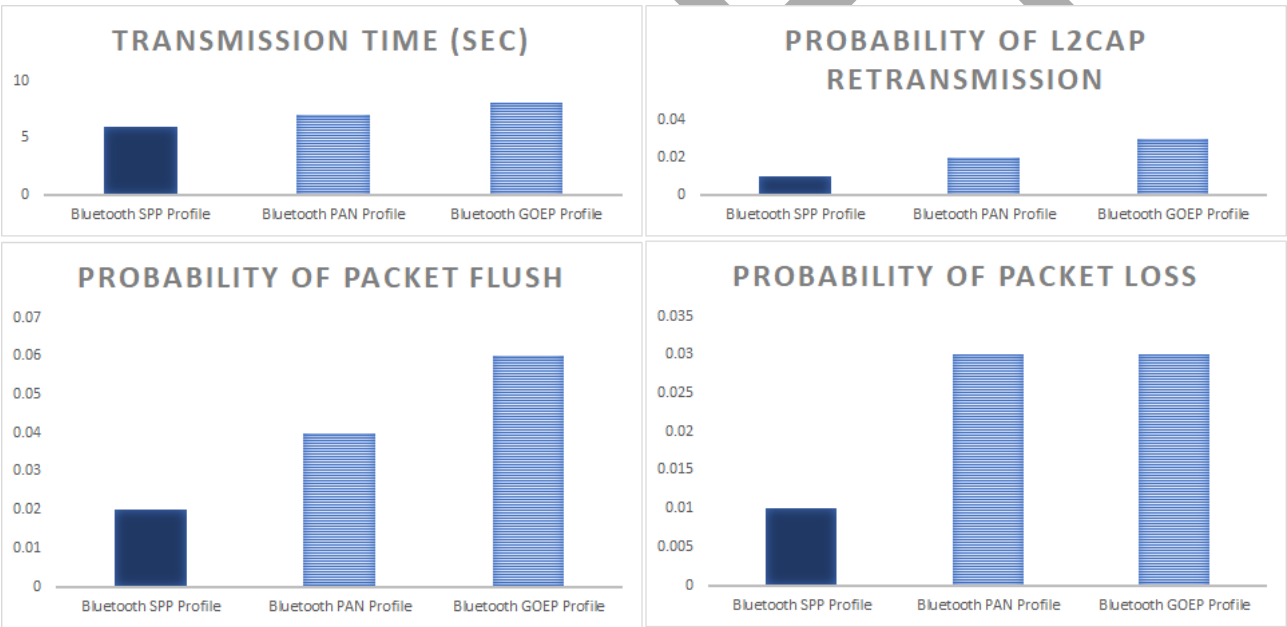

Figure 10 - Bluetooth profiles comparison

Among the different evaluation metrics, the Bluetooth SPP profile can be considered as the best candidate for the purposes of the D2D protocol. It performs much better, achieving higher transmission time with decreased probabilities of packet loss, flush and L2CAP retransmission. In more detail it can be seen that with regards to the Transmission time, the Bluetooth SPP profile performs better since its average mean was almost 6 sec., in comparison with the Bluetooth PAN profile which was 7 sec. and the Bluetooth GOEP profile which was 8 sec. Such a difference is caused due to the fact that the Bluetooth SPP profile follows a different protocol stack than the other two, minimizing the overall obligatory pairing and data exchange interactions. As for the Power consumption, in all cases there was not any significant difference, since for the three Bluetooth profiles there was not calculated any major battery consumption of the interacting

devices. With regards to the Probability of packet flush, the Bluetooth GOEP profile dealt with most packet flushes, since it is built for exchanging binary objects having a nature similar to client-server applications, and as a result the exchanging of roles between the involved parties was leading to more packet flushes, increased L2CAP retransmission and as a result, an increased probability of packet loss. On the other hand, as for the Probability of packet flush the Bluetooth SPP profile was slightly better than the Bluetooth PAN profile, since the nature of the D2D protocol is more identical to the environment created and supported by the Bluetooth SPP profile. Consequently, by the time that the D2D protocol follows the paradigm of emulating a serial cable data transfer instead of creating a local area network, this resulted in less L2CAP retransmission and packet losses for the Bluetooth SPP profile, resulting in being the most appropriate candidate for the overall D2D protocol communication. Nevertheless, it should be considered that the overall implementation was performed mainly for devices supporting Android operating systems. However, it should be also mentioned that the overall purpose of the D2D protocol is to be supported by the top market OS, including the support of the operating systems of Windows and Apple devices. Despite the fact that Windows devices do not have any specific restrictions and requirements, it has been studied that Apple generally 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. Among the supported Bluetooth profiles, the Bluetooth PAN profile is considered as the most appropriate for the current D2D protocol specification for Apple devices, since the Bluetooth SPP profile cannot be supported by the latter due to Apple's development restrictions. Consequently, the overall study results that in the case that the D2D protocol would not be considered to be vendor specific but would only be considered to be efficient and effective, then the Bluetooth SPP profile should be the only choice among the different Bluetooth profiles. However, taking into consideration the desired nature of the D2D protocol to be fully supported by the main market OS, it should be specified for also supporting the Bluetooth PAN profile for the cases of Apple devices.

3.2.3.2 *Android-side D2D over Bluetooth*

Regarding the devices that run Android OS, since the D2D protocol will be based on the Bluetooth v4.0, based on the research that was performed to the different Bluetooth profiles that are supported by the Bluetooth Core Specification version 4.0 [BSC], 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.

3.2.3.3 *iOS-side D2D over Bluetooth*

Regarding the devices that run iOS (i.e. Apple's Operating System), based on the research that was performed to the different Bluetooth profiles that are supported by the Bluetooth Core Specification version 4.0 [BCS], 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. As already discussed, 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.

3.3 D2D Protocol Overview

The D2D protocol defines the Bluetooth operations represented by the interface that will be offered by the mobile application to the healthcare organization's application speaking of the S-EHR app and the HCP app accordingly. The citizens and the HCPs (i.e. actors) will be the only involved ones in the overall interaction, for exchanging identification and healthcare related data, as well as 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 are designed and realized. The first interface (D2D) will be mainly responsible for offering the bluetooth operations to the HCP app in order to invoke them and consequently creating specifically structured messages. These messages will have the form of a request, which will be provided from the HCP app over the Bluetooth connection and as soon as these will be received from the S-EHR app, they will be identified and specific responses will be created. The second interface (D2DClientSecurity) will be responsible for offering the bluetooth operations to the S-EHR app for performing successful and secure Bluetooth connection with the HCP app.

As discussed in Section 3.2-Related Work, 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 11 displays the overall connection between a citizen and an HCP, including the aforementioned interfaces, as well as the involved applications.

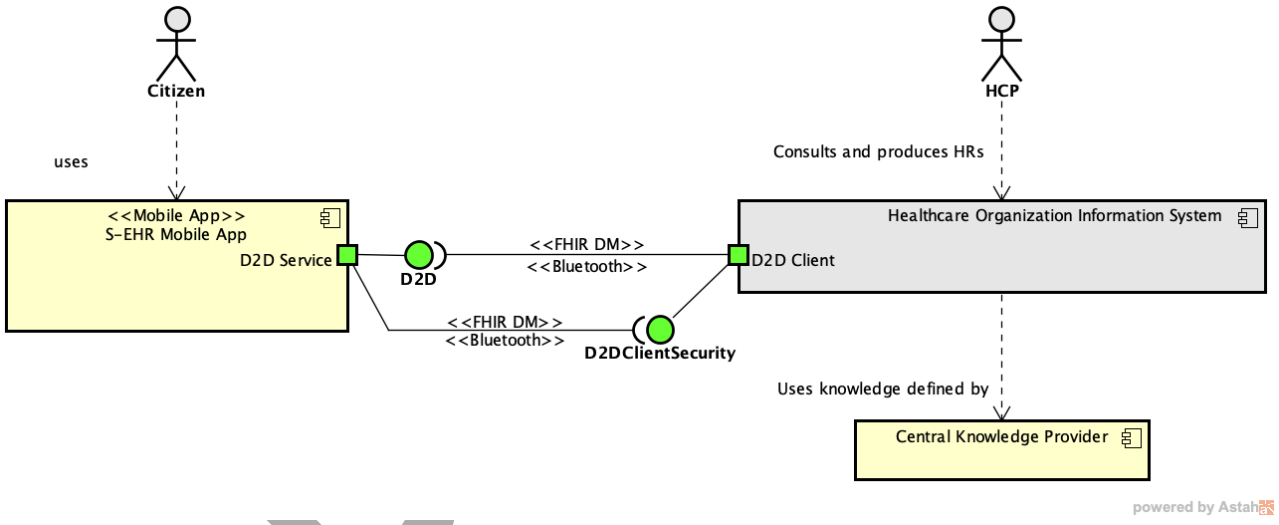

Figure 11 - D2D protocol interfaces

The interactions between the interfaces S-EHR app and the HCP app are explained and visualized through the Sequence Diagram of Figure 12 in high-level detail, in order to provide the sequence that should be followed according to the D2D protocol specification. It should be also noted that the step dealing with the health data request from the side of the HCP is being also described in high-level details (i.e., being classified in an operation called getResources), since the purpose of this diagram is to illustrate the described interactions and the specification of the D2D protocol. For this assumption, more detail is being provided in the explanation of Step 8 of the current Figure. It should be noted that this is a conceptual sequence diagram where the requested operations are embedded in the messages that are being exchanged through the D2D protocol, they are then interpreted by the receiver and are finally executed.

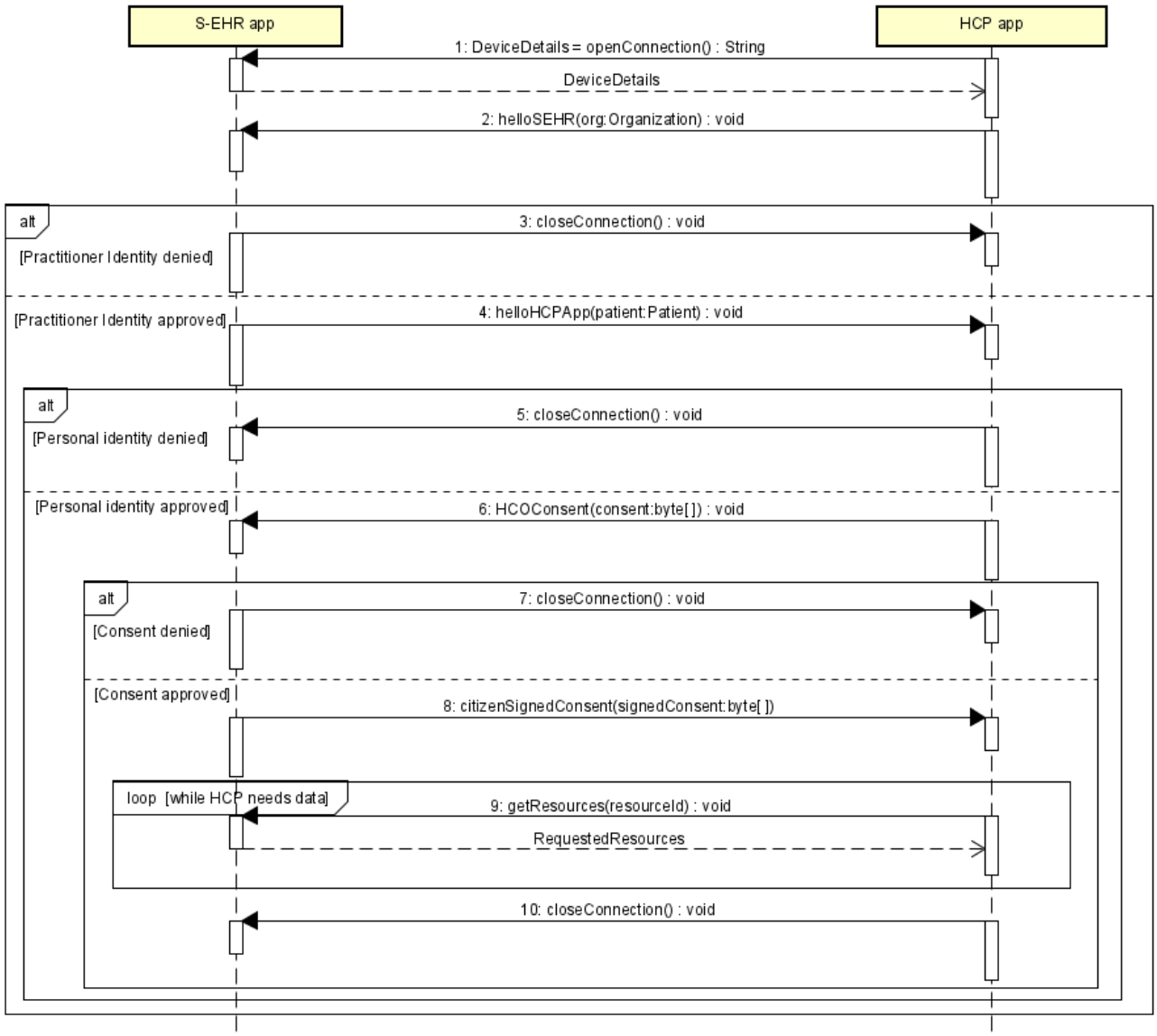

Figure 12 - D2D protocol interactions

The steps of Figure 12 are classified into four (4) main categories for the easier understanding of the D2D protocol specification. These categories are the (i) Establishment of Bluetooth Connection, (ii) Security Handshake, (iii) Exchange Request of Healthcare Data, and (iv) Bluetooth Connection Closure.

(i) Establishment of Bluetooth Connection:

- Step 1 - openConnection: 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 3.2.3, where the involved protocols and interactions are thoroughly described.

(ii) Security Handshake:

- Step 2 - helloSEHR: The next step is for the S-EHR app to invoke this operation for getting the Practitioner's details (i.e. Healthcare Organization), for identifying whether the identity is valid or not.
- Step 3-4: 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 (closeConnection) 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 (helloHCPApp) in the form of an object (Patient).

  It must be mentioned that after this step, the creation and exchange of specific public keys and symmetric keys takes place in order to secure and create a trusted communication. More detail regarding this process is provided in Section 3.4 and in the context of deliverables [D3.4][D3.6][D3.8].

- Step 5-6: 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 (closeConnection) 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 (HCOConsent) in the form of an object (Consent), requesting access to data stored in the S-EHR app.
- Step 7-8: In the case that the temporary consent has been provided, 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 requesting 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 (closeConnection) in the form of a String, indicating that the temporary consent was not approved, hence the connection will be closed.
    - The digitally signed consent of the S-EHR app owner (citizenSignedConsent).

(iii) Exchange Requests of Healthcare Data:

- Step 9 - getResources: In the case that the secure and trusted Bluetooth connection has been established, the HCP app invokes this method for requesting the healthcare data (HL7 FHIR Resources) from the side of the S-EHR app in the form of a request, in order to be securely transmitted over the Bluetooth channel, and to be provided as a response back to the HCP app. This step can be replaced by the following operations, indicating the need of the HCP to read or search on top of specific healthcare data:
    - getResources

- getResourcesByCategories
- getResourcesByCategory
- getMosRecentResourcesByCategory
- getResourcesByIds

Moreover, in order for the HCP app to provide healthcare data (HL7 FHIR Resources) towards the side of the S-EHR app, this can happen along with the previously mentioned operations, following the operation, indicating the need of the HCP to write specific healthcare data:

- setHealthData

All these operations are clearly identified in Section 3.4.3, where their overall meaning, parameters and return values are thoroughly explained. It should be clarified that there exist the three (3) following categories of requests:

- read operation: it is an operation where the HCP app sends the identifier (id) of a specific resource, and the S-EHR app replies with the requested healthcare data
- search operation: it is a complex operation where there are different kinds of requests that can be sent from the HCP to the S-EHR app for searching a specific resource, where the S-EHR app executes different actions depending on the content of the search request (e.g., sending of a specific resource for a specific date)
- write request: it is a request that initiates from the HCP app and is used to provide healthcare data to be stored to the S-EHR app (e.g., the sending of evaluation data from the HCP app to the S-EHR app)

(iv) Bluetooth Connection Closure:

- Step 10 - getConnectionClosureMessage: The last step 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. It should be mentioned that this method can also be invoked from the S-EHR app and has exactly the same functionality. Nevertheless, the D2D protocol specifies that this method should be invoked from the HCP app.

3.4 D2D Protocol Technical Specification

The Conceptual D2D defines the basic operations that the protocol could contain, for the exchange of specific data in the form of specifically designed messages between the HCP and the citizen through the HCP app and the S-EHR app accordingly. Moreover, it must be clarified that regarding the steps of (i) Establishment of Bluetooth Connection, (ii) Security Handshake, and (iii) Bluetooth Connection Closure, the exchanged messages for achieving a secure Bluetooth connection are identified within [D3.4][D3.6][D3.8] in the context of the Security Protocol. Nevertheless, Figure 13 is depicting a high-level sequence diagram of the process defined in Figure 12, including additional details in the context of the security-related tasks, for achieving a secure Bluetooth connection prior to exchanging any health-related data. Shortly, the overall process that is being followed is that the HCP app is providing a public key from its side (step 1) while the S-EHR app is also answering through a public key from the other side (step 2), in the case that citizen identity has been approved by the HCP. Consequently, the exchange of public keys takes place between the involved applications, and finally the exchange of consent happens from the HCP app to the S-EHR app to access the S-EHR app's data (step 3). In the case that this is approved, the consent is digitally signed from the S-EHR app and provided to the HCP app (step 4). As soon as all these steps have been finalized, then the Exchange of Requests of Healthcare Data takes place, where different messages are structured from the

HCP app as requests of different HL7 FHIR resources, until the closure of the Bluetooth connection. These requests are sent through the secure Bluetooth connection to the S-EHR app, where these messages are identified, and specific responses are being created from the S-EHR app to be provided as a reply to the HCP app. All the different types of messages in terms of requests and responses are provided in Section 3.4.3.

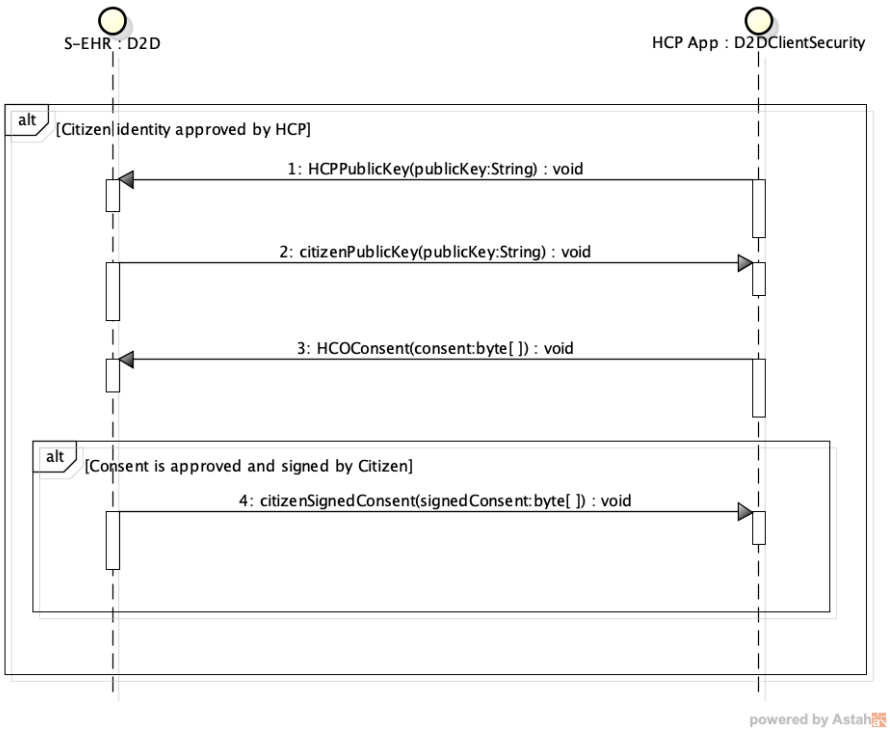

Figure 13 - D2D protocol security interactions

The following figure (Figure 14) shows a UML class diagram representing the involved interfaces of the D2D protocol. D2D is the name of the interface that is offered to the HCP 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. The D2DServerSecurity and the D2DClientSecurity interfaces contain the operations for letting the HCP app and the S-EHR app establish a secure Bluetooth Connection following the process of Figure 13. More detail can be found in the context of [D3.4][D3.6][D3.8].

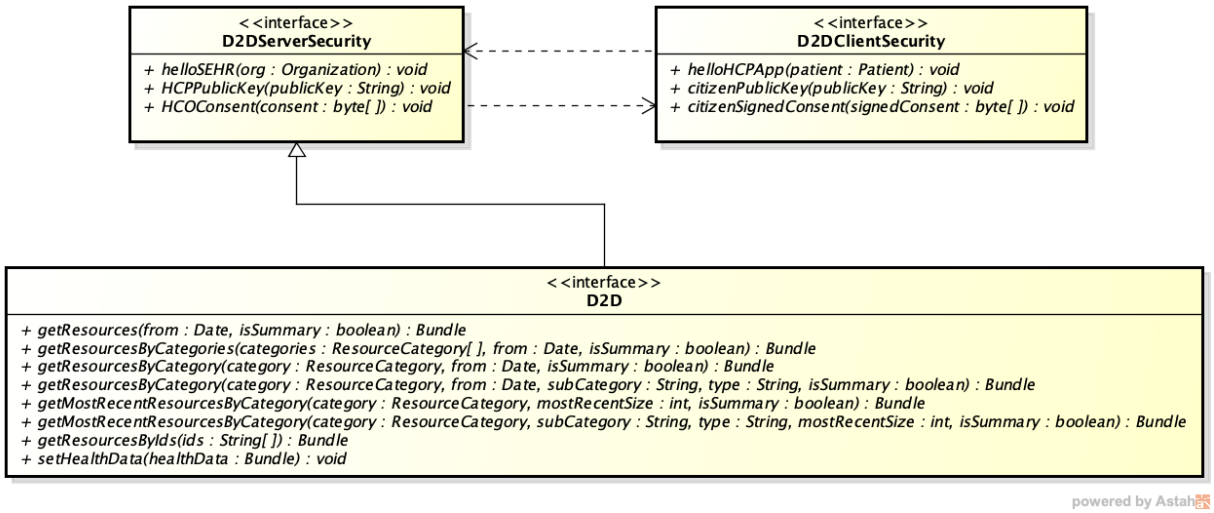

Figure 14 - D2D protocol interface operations

3.4.1 General Structure of Request Messages

The tables below provide a complete description of all the messages in order for the HCP to be able to request different types of resources based on the parameters that are defined in each operation whose invoke creates a specific request message. In detail, each different request creates a specific JSON request which is sent over the Bluetooth connection, it then triggers the creation of a different type of response from the side of the S-EHR app, and initiates the creation of a JSON response which is sent back to the requestor. It should be noted that each read operation and search operation (as specified in Section 3.3) should follow the JSON structure provided below:

```
{
  "id":"",
  "header":
  {
    "itemsPerPage":,
    "timeStamp":"",
    "agent":"",
    "protocol":"D2D",
    "version":""
  },
  "operation":"",
  "parameters":[{"name":"","value":""}]
}
```

Shortly, the JSON attributes represent the following:

- **id**: It is a random string in the form of identifier, created for both specifying and identifying the current request
- **header**: It is a JSON array containing the values for the itemsPerPage, timeStamp, agent, protocol, and version of the current request

- **itemsPerPage**: It is a non-negative integer that represents the total number of items that will be returned as a response for each different page that will be provided (i.e., pagination)
- **timeStamp**: It represents the current date time of the request following the pattern of yyyy-mm-dd'T'HH:mm:ss:SSSZ
- **agent**: It is a string that represents the current JRE as well as the Operating System being used along with its version.
- **protocol**: It is a fixed string having the value of "D2D"
- **version**: It is a string representing the current version of the library being used from the side of the HCP app

- **operation**: It is a string that represents the type of operation (i.e., request) that is provided having one of the following values: READ, SEARCH, CLOSE_CONNECTION
- **parameters**: It is a JSON array that includes the names and the values of the parameters that are provided through the request. These parameters may have the following values:
  - **name**: SUMMARY, **value**: true **|** false
  - **name**: ID, **value**: a String containing the ID of the requested resource
  - **name**: DATE, **value**: YYYY-MM-DD **|** null
  - **name**: MOST_RECENT, **value**: a non negative integer
  - **name**: CATEGORY, **value**: PATIENT_SUMMARY | IMAGE_REPORT | LABORATORY_REPORT | PATIENT | DOCUMENT_REFERENCE | DOCUMENT_MANIFEST | DIAGNOSTIC_REPORT | MEDICATION_REQUEST | CONDITION | IMMUNIZATION | ALLERGY_INTOLERANCE | OBSERVATION | ENCOUNTER | COMPOSITION | PROCEDURE
  - **name**: SUB_CATEGORY, **value**: It is a String, where in some cases it is code from LOINC or SNOMED with the following format: SYSTEM_NAME | CODE
  - **name**: TYPE, **value**: It is a String, where in some cases it is code from LOINC or SNOMED with the following format: SYSTEM_NAME | CODE

Moreover, It should be noted that each write request (as specified in Section 3.3) should follow the format provided below:

```
{
  "id":"",
  "header":
  {
    "itemsPerPage":,
    "timeStamp":"",
    "agent":"",
    "protocol":"D2D",
    "version":""
  },
  "operation":"WRITE",
  "body":""
}
```

Shortly, the JSON values represent the following:

- **id**: It is a random string in the form of identifier, created for both specifying and identifying the current request

- **header**: It is a JSON array containing the values for the itemsPerPage, timeStamp, agent, protocol, and version of the current request
    - **itemsPerPage**: It is a non-negative integer that represents the total number of items that will be returned as a response for each different page that will be provided (i.e., pagination)
    - **timeStamp**: It represents the current date time of the request following the pattern of yyyy-mm-dd'T'HH:mm:ss:SSSZ
    - **agent**: It is a string that represents the current JRE as well as the Operating System being used along with its version.
    - **protocol**: It is a fixed string having the value of "D2D"
    - **version**: It is an string representing the current version of the library being used from the side of the HCP app
- **operation**: It is a string that represents the type of operation (i.e., request) that is provided having the following value: WRITE
- **body**: it is an encrypted string containing the body of the write request that is provided from the HCP app

A detailed explanation of the **JSON-schema** for defining the structure of the D2D request messages can be seen in the ANNEX of the current document.

3.4.2 General Structure of Response Messages

The following tables provide a complete description of the structure that the response messages should follow for the S-EHR app. It must be mentioned that each different response creates a specific JSON response which is sent over the Bluetooth connection back to the requestor. It should be noted that each read operation and search operation (as specified in Section 3.3) should follow the JSON structure provided below:

```
{
  "body":"",
  "header":
  {
    "page":,
    "requestId":"",
    "totalPages":"",
    "agent":"",
    "protocol":"D2D",
    "timestamp":"",
    "version":""
  },
  "id":"",
  "status":""
}
```

Shortly, the JSON values represent the following:

- **body**: It is a string containing the encrypted content of the response with the healthcare data that will be provided to the HCP app

- **header**: It is a JSON array containing the values for the itemsPerPage, timeStamp, agent, protocol, and version of the current request
    - **page**: It represents a non-negative integer showing the current number of page that will be provided as a response
    - **requestId**: It is a string that has to be the same as the id of request in order to avoid possible conflicts of sending different responses to different requests
    - **totalPages**: It represents a non-negative integer that contains the total number of pages that are about to be provided to the requestor
    - **agent**: It is a string that represents the current JRE as well as the Operating System being used along with its version.
    - **protocol**: It is a fixed string having the value of "D2D"
    - **timestamp**: It represents the current date time of the response following the pattern of yyyy-mm-dd'T'HH:mm:ss:SSSZ
    - **version**: It is an integer representing the current version of the library being used from the side of the S-EHR app
- **id**: It is a random string in the form of identifier, created for both specifying and identifying the current response
- **status**: It is an integer showcasing the status of the current response, having one of the following values:
    - **200**: Request was successfully executed
    - **300**: Request was executed but ended with an exception
    - **400**: The received request does not have a valid format
    - **410**: The received response does not have a valid format
    - **500**: An exception was raised in D2D classes

A detailed explanation of the **JSON-schema** for defining the structure of the D2D response messages can be seen in the ANNEX of the current document.

3.4.3 Specification of D2D Requests and Responses

Below, an example of each different request is provided, through the explanation of the operation of the D2D interface that creates this message, being followed by an example of the corresponding response.

3.4.3.1 *Search all the resource types*

This is considered as a search operation that can be used for requesting all the available resources (of whatever type) from a specific date, considering whether it will be provided in the form of a summary or not.

Name	getResources
Description	A search operation that can be used for requesting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not.

Arguments	• from: a parameter containing the date from which the available resources will be provided • isSummary: a boolean showcasing whether the provided resources will be in the form of a summary or not
Return Value	FHIR Bundle containing the requested data
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

Example of JSON Request created through the **getResources**:


```
{
  "id":"f2336f76-3412-4a21-86c7-d5b95a250f54",
  "header":
  {
    "itemsPerPage":0,
    "timeStamp":"2021-06-18T12:19:40.693Z",
    "agent":"JRE 1.8.0_261 - Windows 10 10.0",
    "protocol":"D2D",
    "version":"1"
  },
  "operation":"SEARCH",
  "parameters":[{"name":"DATE","value":"2019-02-14"},{"name":"SUMMARY","value":false}]
}
```


On the request for getting the resources from a specific date, either in the form of a summary or not, an example of the corresponding response is provided below.


```
{
  "body":"SPBOFjJ//IHExMPpnW6c12Wxx7xzuBDBRRga6imC8AOvga+mApuWT+m8UcYsxBQQ##",
  "header":
  {
    "page":1,
    "requestId":"82cb82f8-8c54-4abd-8a47-785ddcfa87fd",
    "totalPages":1,
    "agent":"JRE 0 - Linux 4.4.111-21654000",
```

```
    "protocol":"D2D",
    "timeStamp":"2021-06-18T13:07:31.345Z",
    "version":"1"
  },
  "id":"cd0777b6-cfbe-45be-ba67-a15e9fc61749",
  "status":200
}
```


3.4.3.2 *Search the Resources that belong to specific categories*

This is considered as a search operation that can be used for requesting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the categories in which these resources belong to.

Name	getResourcesByCategories
Description	A search operation that can be used for requesting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the categories in which these resources belong to.
Arguments	• categories: a String that may be provided multiple times (depending on the total number of the requesting categories) and can have the values specified in Section 2as follows: ○ **name**: CATEGORY, **value**: PATIENT_SUMMARY | IMAGE_REPORT | LABORATORY_REPORT | PATIENT | DOCUMENT_REFERENCE | DOCUMENT_MANIFEST | DIAGNOSTIC_REPORT | MEDICATION_REQUEST | CONDITION | IMMUNIZATION | ALLERGY_INTOLERANCE | OBSERVATION | ENCOUNTER | COMPOSITION | PROCEDURE • from: a parameter containing the date from which the available resources will be provided • isSummary: a boolean showcasing whether the provided resources will be in the form of a summary or not
Return Value	FHIR Bundle containing the requested data
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

Example of JSON Request created through the **getResourcesByCategories**:


```
{
  "id":"af67b3e9-b7fc-43ce-82e0-6ce018c6082e",
  "header":
  {
    "itemsPerPage":0,
    "timeStamp":"2021-06-18T12:22:54.757Z",
    "agent":"JRE 1.8.0_261 - Windows 10 10.0",
    "protocol":"D2D",
    "version":"1"
 },
 "operation":"SEARCH",
 "parameters":
[{"name":"DATE","value":"2019-02-
14"},{"name":"SUMMARY","value":false},{"name":"CATEGORY","value":"LABORATORY_REPORT"},{"name
":"CATEGORY","value":"OBSERVATION"}]
}
```


On the request for getting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the categories in which these resources belong to, an example of the corresponding response is provided below.


```
{
  "body":"SPBOFjJ//IHExMPpnW6c12Wxx7xzuBDBRRga6imC8AOvga+mApuWT+m8UcYsxBQQ##",
  "header":
  {
    "page":1,
    "requestId":"82cb82f8-8c54-4abd-8a47-785ddcfa87fd",
    "totalPages":1,
    "agent":"JRE 0 - Linux 4.4.111-21654000",
    "protocol":"D2D",
    "timeStamp":"2021-06-18T13:07:31.345Z",
    "version":"1"
  },
  "id":"f3be5eb6-ff82-4810-a616-5206a23faa03",
  "status":200
}
```

3.4.3.3 *Search the Resources that belong to a single category*

This is considered as a search operation that can be used for requesting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to.

Name	getResourcesByCategory
Description	A search operation that can be used for requesting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to.
Arguments	• category: a String that may be provided a single time that can have the values specified in Section2as follows: ○ **name**: CATEGORY, **value**: PATIENT_SUMMARY | IMAGE_REPORT | LABORATORY_REPORT | PATIENT | DOCUMENT_REFERENCE | DOCUMENT_MANIFEST | DIAGNOSTIC_REPORT | MEDICATION_REQUEST | CONDITION | IMMUNIZATION | ALLERGY_INTOLERANCE | OBSERVATION | ENCOUNTER | COMPOSITION | PROCEDURE • from: a parameter containing the date from which the available resources will be provided • isSummary: a boolean showcasing whether the provided resources will be in the form of a summary or not
Return Value	FHIR Bundle containing the requested data
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

Example of JSON Request created through the **getResourcesByCategory**:

```
{
  "id":"af67b3e9-b7fc-43ce-82e0-6ce018c6082e",
  "header":
```

```
 {
   "itemsPerPage":0,
   "timeStamp":"2021-06-18T12:22:54.757Z",
   "agent":"JRE 1.8.0_261 - Windows 10 10.0",
   "protocol":"D2D",
   "version":"1"
 },
"operation":"SEARCH",
"parameters":
[{"name":"DATE","value":"2019-02-
14"},{"name":"CATEGORY","value":"PATIENT_SUMMARY"},{"name":"SUMMARY","value":"false"}]
}
```


On the request for getting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, an example of the corresponding response is provided below.


```
{
  "body":"SPBOFjJ//IHExMPpnW6c12Wxx7xzuBDBRRga6imC8AOvga+mApuWT+m8UcYsxBQQ##",
  "header":
  {
    "page":1,
    "requestId":"82cb82f8-8c54-4abd-8a47-785ddcfa87fd",
    "totalPages":1,
    "agent":"JRE 0 - Linux 4.4.111-21654000",
    "protocol":"D2D",
    "timeStamp":"2021-06-18T13:07:31.345Z",
    "version":"1"
  },
  "id":"f3be5eb6-ff82-4810-a616-5206a23faa03",
  "status":200
}
```


3.4.3.4 *Search the Resources that belong to a single category and subcategory*

This is considered as a search operation that can be used for requesting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, their subcategory and their type.

Name	getResourcesByCategory
Description	A search operation that can be used for requesting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, their subcategory and their type.
Arguments	• category: a String that may be provided a single time that can have the values specified in Section 2 as follows: ○ **name**: CATEGORY, **value**: PATIENT_SUMMARY | IMAGE_REPORT | LABORATORY_REPORT | PATIENT | DOCUMENT_REFERENCE | DOCUMENT_MANIFEST | DIAGNOSTIC_REPORT | MEDICATION_REQUEST | CONDITION | IMMUNIZATION | ALLERGY_INTOLERANCE | OBSERVATION | ENCOUNTER | COMPOSITION | PROCEDURE • from: a parameter containing the date from which the available resources will be provided • subCategory: It is a String, where in some cases it is code from LOINC or SNOMED with the following format: SYSTEM_NAME | CODE • type: It is a String, where in some cases it is code from LOINC or SNOMED with the following format: SYSTEM_NAME | CODE • isSummary: a boolean showcasing whether the provided resources will be in the form of a summary or not
Return Value	FHIR Bundle containing the requested data
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

Example of JSON Request created through the **getResourcesByCategory**:


```
{
  "id":"af67b3e9-b7fc-43ce-82e0-6ce018c6082e",
  "header":
  {
    "itemsPerPage":0,
    "timeStamp":"2021-06-18T12:22:54.757Z",
```

```
    "agent":"JRE 1.8.0_261 - Windows 10 10.0",
    "protocol":"D2D",
    "version":"1"
 },
 "operation":"SEARCH",
 "parameters":
[{"name":"CATEGORY","value":"OBSERVATION"},{"name":"SUB_CATEGORY","value":"vital-signs"},{"name":"DATE","value":"2019-02-04"},{"name":"SUMMARY","value":"false"}]
}
```


On the request for getting all the available resources from a specific date, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, their subcategory and their type, an example of the corresponding response is provided below:


```
{

"body":"SPBOFjJ//lHExMPpnW6c12Wxx7xzuBDBRRga6imC8AOoXVeEoqZ6gnox8eFN1zZlStZKf0BQ3/TR##kuTk+EPLvwxr3ZUdY1fA3v52E0++V72gtNO0If4uj2/EETMXI+FfEdwqqw9zt46UQ9foJxir6fPt##FSSAy09zJ+QnrjK+PNotksA+mMWsPAX269jIBoobx1EQNxx0UQlqgdEeu2uu6gsUSrcXaPDPUovt##dgGZhTOAWzPcQbhC0+johlFJpDPZFhchItQd6sxEHl1jRy69NtmpbOIKKtU+jMgfPlWQiAoJvh3a##FxYXQEkHOj08OCymyQJpMY13kQseyAB9f4AJ5zSYFveIIEbTMf",
  "header":
  {
    "page":1,
    "requestId":"82cb82f8-8c54-4abd-8a47-785ddcfa87fd",
    "totalPages":1,
    "agent":"JRE 0 - Linux 4.4.111-21654000",
    "protocol":"D2D",
    "timeStamp":"2021-06-18T13:07:31.345Z",
    "version":"1"
  },
  "id":"f66a46d4-ff51-488d-881e-008b45414be5",
  "status":200
}
```


3.4.3.5 *Search the most recent Resources that belong to a single category*

This is considered as a search operation that can be used for requesting the most recent resources with a specific size, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to.

Name	getMostRecentResourcesByCategory

Description	A search operation that can be used for requesting the most recent resources with a specific size, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to.
Arguments	• category: a String that may be provided a single time that can have the values specified in Section 2 as follows: ○ **name**: CATEGORY, **value**: PATIENT_SUMMARY | IMAGE_REPORT | LABORATORY_REPORT | PATIENT | DOCUMENT_REFERENCE | DOCUMENT_MANIFEST | DIAGNOSTIC_REPORT | MEDICATION_REQUEST | CONDITION | IMMUNIZATION | ALLERGY_INTOLERANCE | OBSERVATION | ENCOUNTER | COMPOSITION | PROCEDURE • mostRecentSize: an integer showcasing the total size of the requested resources • isSummary: a boolean showcasing whether the provided resources will be in the form of a summary or not
Return Value	FHIR Bundle containing the requested data
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

Example of JSON Request created through the **getMostRecentResourcesByCategory**:


```
{
  "id":"f2336f76-3412-4a21-86c7-d5b95a250f54",
  "header":
  {
    "itemsPerPage":0,
    "timeStamp":"2021-06-18T12:19:40.693Z",
    "agent":"JRE 1.8.0_261 - Windows 10",
    "protocol":"D2D",
    "version":"1"
  },
  "operation":"SEARCH",
```

```
  "parameters":
[{"name":"CATEGORY","value":"IMAGE_REPORT"},{"name":"MOST_RECENT","value":2},{"name":"SUMM
ARY","value":"true"}]
}
```


On the request for getting the most recent resources with a specific size, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, an example of the corresponding response is provided below.


```
{
  "body":"SPBOFjJ//IHExMPpnW6c12Wxx7xzuBDBRRga6imC8AOvga+mApuWT+m8UcYsxBQQ##",
  "header":
  {
    "page":1,
    "requestId":"82cb82f8-8c54-4abd-8a47-785ddcfa87fd",
    "totalPages":1,
    "agent":"JRE 0 - Linux 4.4.111-21654000",
    "protocol":"D2D",
    "timeStamp":"2021-06-18T13:07:31.345Z",
    "version":"1"
  },
  "id":"f3be5eb6-ff82-4810-a616-5206a23faa03",
  "status":200
}
```


3.4.3.6 *Search the most recent Resources that belong to a single category and subcategory*

This is considered as a search operation that can be used for requesting the most recent resources with a specific size, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, their subcategory and their type.

Name	getMostRecentResourcesByCategories
Description	A search operation that can be used for requesting the most recent resources with a specific size, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, their subcategory and their type.
Arguments	• category: a String that may be provided a single time that can have the values specified in Section 2 as follows: ○ **name**: CATEGORY, **value**: PATIENT_SUMMARY | IMAGE_REPORT | LABORATORY_REPORT | PATIENT |

	DOCUMENT_REFERENCE | DOCUMENT_MANIFEST | DIAGNOSTIC_REPORT | MEDICATION_REQUEST | CONDITION | IMMUNIZATION | ALLERGY_INTOLERANCE | OBSERVATION | ENCOUNTER | COMPOSITION | PROCEDURE • subCategory: It is a String, where in some cases it is code from LOINC or SNOMED with the following format: SYSTEM_NAME | CODE • type: It is a String, where in some cases it is code from LOINC or SNOMED with the following format: SYSTEM_NAME | CODE • mostRecentSize: an integer showcasing the total size of the requested resources • isSummary: a boolean showcasing whether the provided resources will be in the form of a summary or not
Return Value	FHIR Bundle containing the requested data
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

Example of JSON Request created through the **getMostRecentResourcesByCategories**:


```
{
  "id":"f2336f76-3412-4a21-86c7-d5b95a250f54",
  "header":
  {
    "itemsPerPage":0,
    "timeStamp":"2021-06-18T12:19:40.693Z",
    "agent":"JRE 1.8.0_261 - Windows 10",
    "protocol":"D2D",
    "version":"1"
  },
  "operation":"SEARCH",
  "parameters":
[{"name":"CATEGORY","value":"LABORATORY_REPORT"},{"name":"MOST_RECENT","value":2},{"name":"
SUB_CATEGORY","value":"vital signs"},{"name":"SUMMARY","value":"true"}]
}
```

On the request for getting the most recent resources with a specific size, considering whether it will be provided in the form of a summary or not, providing also the single category in which these resources belong to, their subcategory and their type, an example of the corresponding response is provided below.


```
{
  "body":"SPBOFjJ//IHExMPpnW6c12Wxx7xzuBDBRRga6imC8AOvga+mApuWT+m8UcYsxBQQ##",
  "header":
  {
    "page":1,
    "requestId":"82cb82f8-8c54-4abd-8a47-785ddcfa87fd",
    "totalPages":1,
    "agent":"JRE 0 - Linux 4.4.111-21654000",
    "protocol":"D2D",
    "timeStamp":"2021-06-18T13:07:31.345Z",
    "version":"1"
  },
  "id":"f66a46d4-ff51-488d-881e-008b45414be5",
  "status":200
}
```


3.4.3.7 *Read specific Resources of a specific Id*

This is considered as a read operation that can be used for requesting specific resources with a specific identifiers.

Name	getResourcesByIds
Description	A read operation that can be used for requesting specific resources with a specific identifier.
Arguments	• ids: a list of strings that provides the unique identifiers of the requesting resources
Return Value	FHIR Bundle containing the requested data
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

Example of JSON Request created through the **getResourcesById**:


```
{
  "id":"f2336f76-3412-4a21-86c7-d5b95a250f54",
  "header":
  {
    "itemsPerPage":0,
    "timeStamp":"2021-06-18T12:19:40.693Z",
    "agent":"JRE 1.8.0_261 - Windows 10",
    "protocol":"D2D",
    "version":"1"
  },
  "operation":"READ",
  "parameters":
[{"name":"ID","value":"4134134-341234-5674-76534"},{"name":"ID","value":"97979-2342342-776567-12345"}]
}
```


On the request for getting specific resources with specific identifiers, an example of the corresponding response is provided below:


```
{
  "body":"SPBOFjJ//IHExMPpnW6c12Wxx7xzuBDBRRga6imC8AOvga+mApuWT+m8UcYsxBQQ##",
  "header":
  {
    "page":1,
    "requestId":"82cb82f8-8c54-4abd-8a47-785ddcfa87fd",
    "totalPages":1,
    "agent":"JRE 0 - Linux 4.4.111-21654000",
    "protocol":"D2D",
    "timeStamp":"2021-06-18T13:07:31.345Z",
    "version":"1"
  },
  "id":"f66a46d4-ff51-488d-881e-008b45414be5",
  "status":200
}
```


3.4.3.8 *Write specific Resources*

This is considered as a write operation that can be used for sending specific resources with a specific identifier that should be written to the S-EHR app.

Name	setHealthData
Description	A write operation that can be used for sending specific resources with a specific identifier that should be written to the S-EHR app
Arguments	• healthData: a Bundle message containing the body of the resource to be written
Return Value	This operation is void
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

Example of JSON Request created through the **setHealthData**:


```
{
  "id":"f2336f76-3412-4a21-86c7-d5b95a250f54",
  "header":
  {
    "itemsPerPage":0,
    "timeStamp":"2021-06-18T12:19:40.693Z",
    "agent":"JRE 1.8.0_261 - Windows 10",
    "protocol":"D2D",
    "version":"1"
  },
  "operation":"WRITE",
"body":"CYCXDCwk09Wx4c3PdDutaNq1Npco3+RRHRFojAyCzEgnZssh/tyOiO2LyNhFRxmnYxZnKzIvGiKhiH
QT53dIoyoK/C5jTeiZbh1VnPcax3DQW6KrnWXzEe+MFNXYaYUOo25/Wy5nusu7dTc88//ZR6PV0R3JlBQFRB
1gck0uiNkhaN7j"
}
```


3.4.4 Management of Large Image Files

Complex diagnostic investigations that require images are not based on single jpeg files, but usually on a DICOM study (a series of large images compliant with the DICOM standard). The DICOM protocol is the standard for the communication and management of medical imaging information and related data, it is most commonly used for storing and transmitting medical images enabling the integration of medical

imaging devices and picture archiving and communication systems (PACS) from multiple manufacturers. HL7 FHIR specifications define the ImagingStudy resource for representing a DICOM study, where this resource can be referenced by an instance of DiagnosticReport. So, an instance of a DiagnosticReport may:

- contain one or more media files.
- reference to an instance of ImagingStudy.

In the latter case the ImagingStudy itself does not contain any image, it only contains all the needed references to access a WADO-RS [WADO-RS] server to download the images of the DICOM study. WADO-RS is an HTTP protocol that is part of the wider standard protocol called DICOMWeb [DICOMWEB], defining the operations for retrieving DICOM objects over the HTTP protocol.

In the case where the citizen has to provide large image files to the side of the HCP, the S-EHR app can provide the DiagnosticReport as defined in Section 3.4, and afterwards the HCP app can use the reference that is found into the imaging study to access a DICOM study following the process defined in the R2D Emergency protocol for the DICOM supporting (Section 6.4.7).

4 R2D ACCESS PROTOCOL: REMOTE HEALTH DATA EXCHANGE

While the D2D protocol allows to exchange health data between a S-EHR and a HCP App of a healthcare organisation at short distance, the R2D Access protocol (Figure 15) defines the set of operations used for downloading (in a standard way) health data from a healthcare organisation to an application (S-EHR app) acting on behalf of a patient. The healthcare organisation may be a healthcare provider (e.g. a hospital, a private laboratory or a general practitioner), or a national EHR provider. The R2D Access allows a S-EHR app to download health data of a citizen in a standard way from heterogeneous sources of data even pertaining to different countries.

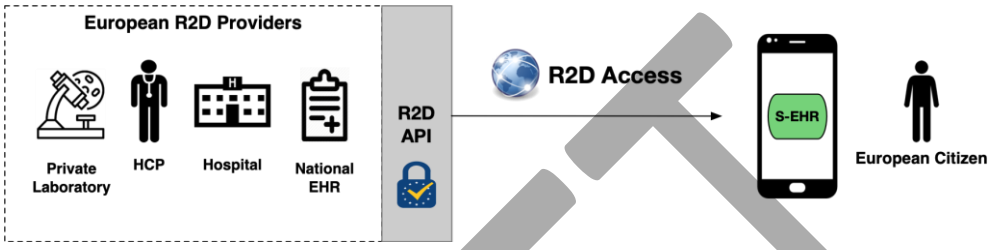

Figure 15 - R2D Access overview

Health data of a citizen is distributed across several different HCOs (mostly private laboratories and hospitals), so it is expected that a citizen will import health data from several HCOs, creating in this way an integrated EHR on his / her mobile (Figure 16).

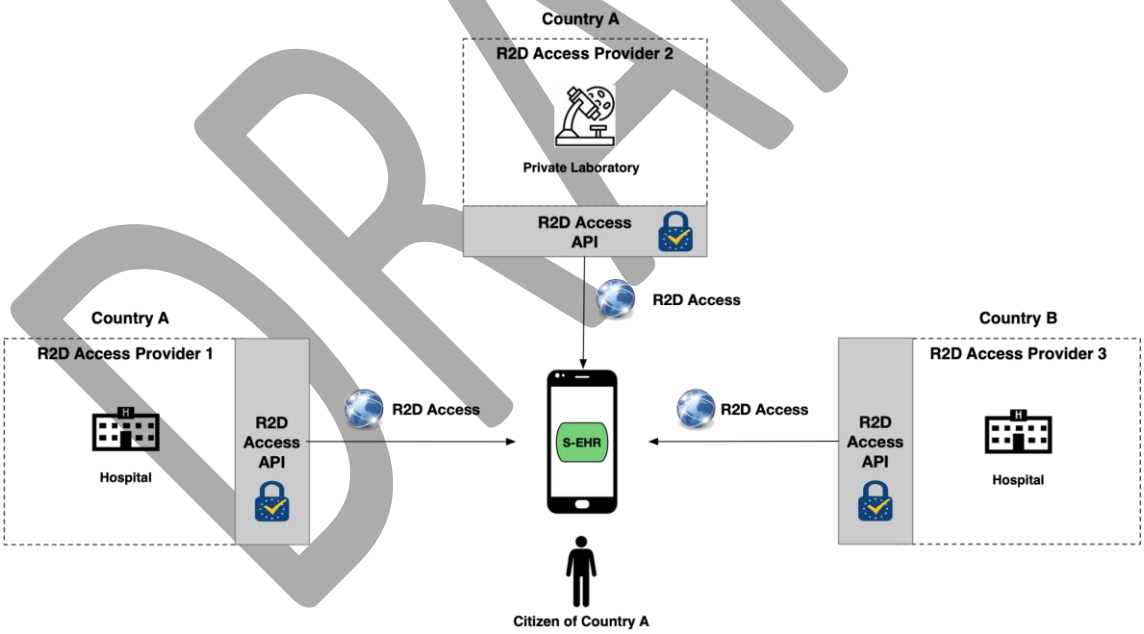

Figure 16 - R2D Access example

The remainder of this chapter is structured as following:

- R2D Access Related Works: state of the art of Cross Border Health Data Exchange related to R2D Access goals.
- R2D Access Overview: high level description of the R2D Access protocol, defining the building blocks of R2D Access and the overall structure of a R2D Access typical transaction.

- R2D Access Technical Specification: concrete specification of the R2D Access protocol (defines the list of the supported operations, the accepted parameters, the returned values and the possible exceptions).

4.1 R2D Access Related Works

This section describes four relevant initiatives that have goals similar to R2D Access.

4.1.1 eHDSI

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 [CEF].

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 17) is based on a closed and trusted federation of National Contact Points (NCP) one for each Member State named eHDSI Circle of Trust; an NCP is a service representing a Member State's EHR, it provides a reduced-but-common API designed for allowing EHRs of EU countries to InteropEHRate in order to exchange health data.

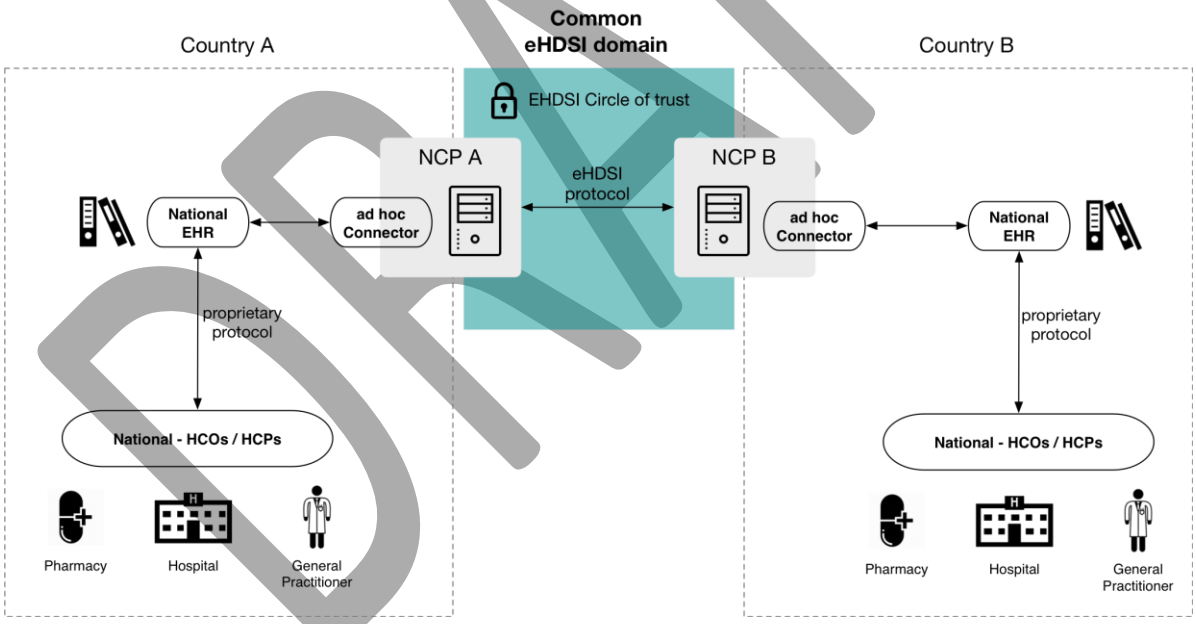

Figure 17 - eHDSI system architecture

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 of health data coming from its country to the NCPs of other countries that owns the requested data;
- handling incoming requests coming from other NCPs;
- converting data from eHDSI data representation to owner country data representation.

An HCP of Country A, that wants access to health data of a citizen of Country B , 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). 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 protocols share many objectives (mainly the cross-border exchange of health data in a standard way), however, they show also some relevant differences:

- eHDSI is available only to HCOs and HCPs, it has not been designed considering the citizen as a primary actor. In eHDSI, a citizen cannot submit requests to his/her or his reference NCP (NCP of his country). Differently, in InteropEHRate the citizen is at the centre of the data exchange process. They can download their health data from any (local or national) EHR that supports R2D Access (or D2D) to their smart mobile device, and provide the required health data directly to the 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-Access adopts a subset of the standard FHIR API and data model. It means that every application already interacting with a FHIR compliant repository, may be easily converted to use R2D Access.

Note that the adoption of FHIR is also under consideration of eHDSI. The European Commission outlined this 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©)”.

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 the 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.

The Patient Summary data model proposed by the IPS project is adopted by R2D Access protocol as the standard structured representation of a Patient Summary. The IPS project is focused on a Patient Summary standard representation in FHIR, while it does not define any standard operation to retrieve it from a FHIR server, so regarding this last topic the R2D Access provides several operations to search and retrieve the Patient Summary of a citizen.

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 Argonaut 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 its Health app. 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].

The Argonaut project is a U.S. initiative thus it is based on a care provisioning model that is different from the European one. This makes it difficult to adopt this model for European Healthcare Organizations. Apart from this, the strong support to the adoption of FHIR by important IT players members of the Argonaut initiative (the motto of the initiative is Accelerating FHIR) and the use of the smartphone as a citizen EHR proposed by Apple are important connecting points between Argonaut and R2D Access, promoting the technical choices made while designing R2D Access protocol.

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.

The 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 Access and IPA share a common approach mainly because they are both designed for the citizen, and they are both based on FHIR. Unlike R2D Access, IPA has the objective to define a minimum set of operations to access "patient records anywhere in the world" while the reference context of R2D Access is Europe. The wider scope of IPA obviously makes it a more generic protocol than R2D Access, R2D Access is tailored to European health care processes and adopts eIDAS, the European standard for citizen identification. At the moment (2021) it is not clear how the project is proceeding, specification is stuck at version 0.1.0 and this does not help to compare the two protocols.

4.2 R2D Access Overview

R2D Access is an internet based protocol, its operations are exposed as RESTful services. A strong requirement from the InteropEHRate consortium was to define R2D Access operations 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 previous state of the art section).

Unlike other protocols, FHIR not only defines a data model (how to represent data), but also defines a set of RESTful operations (how to manage data) [FHIR API SPEC] for creating, updating, deleting and searching health data. R2D Access has been defined applying constraints and restrictions to the FHIR RESTful API. The main reason behind these constraints is defining an easy to implement specification, and also facilitating healthcare organizations that have already adopted the FHIR RESTful API. The simplification of the protocol has been achieved by following guidelines:

1. Limiting the types of data (handled by the R2D Access) only to health data relevant for the citizen. FHIR defines, depending on the version, between 130 and 150 types of data, organized in the following categories: Foundation, Base, Clinical, Financial, Specialized. R2D Access is only designed to exchange health data of the citizen, so the types of health data managed by R2D Access has been limited only to the ones (Patient, Condition, AllergyIntolerance, Immunization, Procedure, Observation, MedicationRequest, DocumentReference, DocumentManifest, DiagnosticReport, Encounter, Composition) that have been marked as relevant by the InteropEHRate stakeholders. Despite this restriction, R2D Access handles all the types of data defined by eHDSI (Patient Summary, Prescription, Laboratory Reports, Medical Images and Discharge Reports) plus several others (i.e. DocumentReference, Observation, Encounter and every kind of Diagnostic Report).
2. Providing access only to citizens and authorizing citizens only to retrieve their own data. Narrowing searches to the current authenticated citizen (*Patient's Compartment* using FHIR geargon) avoids the need for the clients to explicitly add a search parameter specifying the subject of the search operation. Every operation provided by R2D Access works at the Patient level, so the conceptual operation "give me all the DiagnosticReport" means "give me all the DiagnosticReport of the authenticated citizen" (and not "give me all the DiagnosticReport of your database").
3. Limiting the number of parameters that can be used in the search operations to the ones required by the InteropEHRate scenarios. The most important parameters needed by a client are the parameters used for defining the time interval of a search operation (as stated before in every R2D Access operation there is always an implicit search parameter that is the authenticated citizen).
4. Limiting the possible values of search parameters regarding the status of a health data. Health data retrieved by R2D Access are for the citizen, this implies that only committed and final data must be provided through R2D Access. Temporary health data, or health data that must still be elaborated (by HCPs or by software) must not be provided through R2D Access. So, depending on the specific

type of health data, search narrowing is also applied on the *status* attribute's value. So, the conceptual operation "give me all the DiagnosticReport" means "give me all the DiagnosticReport of the authenticated citizen whose status is final" (and not "give me all the DiagnosticReport of your database with whatever status").

An R2D Access request is a FHIR compliant request, corresponding to an invocation of a RESTful service using the HTTP protocol and FHIR/JSON serialization. Other than the restrictions described before, there are additional points that makes R2D Access a specialised version of FHIR:

- The specific usage of asynchronicity for some of the R2D Access transactions. This is a new feature of R2D Access v.3, required by the InteropEHRate medical partners to support cases where the health data cannot be shared with the patient in real time.
- The usage of eIDAS as the reference identity provider for any R2D Access server. It means that R2D Access implementers MUST process requests that contain valid eIDAS tokens. European citizens authenticate to every instance of R2D Access everywhere in Europe using their unique eIDAS identity.
- The definition of an optional operation for citizen's registration exposed by healthcare organisations to receive identity attributes that are currently not supported by eIDAS and that are required by healthcare organisations that do not know yet the eIDAS identity of the citizen and need to associate it to a legacy patient's identity.
- The optional support of WADO-RS operations for retrieving images (DICOM studies), also integrated with eIDAS.

The following chapters describe in detail the above points.

4.2.1 Health data completeness and asynchronous interactions

The FHIR specification assumes that a search operation returns any health data matching the request and available, in electronic (FHIR) format, within the invoked FHIR repository. The R2D Access specification adds a strongest business constraint: *an HCO serving an R2D Access request must return all matching health data produced by the HCO that the patient has the legal right to obtain.*

Providing the R2D Access protocol, the HCO commits to satisfy the above constraint. If a portion of the patient's health data are still available only in paper format, the HCO cannot return an incomplete response containing only the health data that have been already converted in electronic format. The HCO will exploit the asynchronous feature of R2D Access to take the needed time to make all the required health data available in an electronic (structured or unstructured) format.
More in general, serving synchronous responses to a S-EHR may be difficult, because health data of the citizens resides in several legacy systems of the HCO and they need to be extracted and converted to the FHIR representation. Depending on the specific legacy systems and the specific business processes of the HCO, these operations may require a relevant amount of time. Moreover, for security and management reasons the healthcare organisation may prefer to maintain the health data to be shared with the patients in a repository separated from the internal EHR.

For these reasons, the traditional HTTP synchronous pattern may be too limiting. Likely, the current FHIR standard also supports an asynchronous interaction [FHIR ASYNC API], following the pattern specified by

IETF [rfc 7240]. In order to support the aforementioned needs, every R2D Access server MUST support the FHIR asynchronous interactions for specific data exchange operations.

Note that, although HTTP is based only on synchronous transactions, [rfc 7240] standardised how to perform asynchronous transactions by using several synchronous messages. In a traditional HTTP transaction a client sends a request to a server, and waits for the results. The transaction ends when data are returned to the client, or a timeout error is raised. Instead in the asynchronous HTTP pattern, a client sends an initial request to a server (asking for some data), the server does not reply with the requested data, but only notifying the client that the request has been accepted and will be processed as soon as possible. In this initial response, the server provides an additional URL that the client can use to check if the processing of the request is finished. The client can periodically send a GET request to the provided URL monitoring the returned HTTP code: 202 means that the request is still under processing, while 200 means the request has terminated. In this case the server provides the URLs that the client shall use to retrieve the requested data. The following sequence diagram (Figure 18) shows the overall structure of an asynchronous transaction between a S-EHR and a R2D Access server:

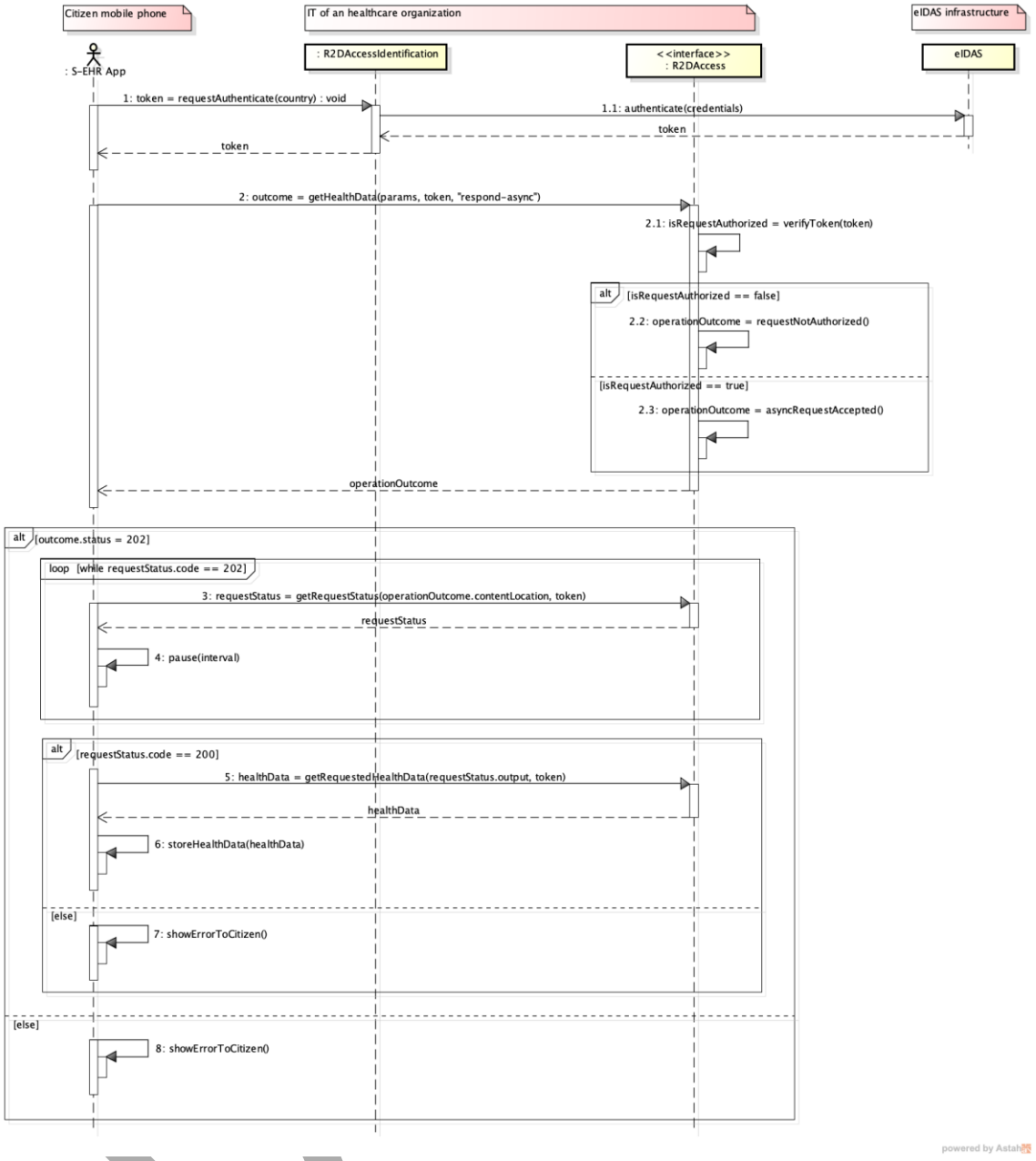

Figure 18 - Asynchronous request processing

- **step 1:** the citizen authenticates to eIDAS. The authentication process represented in this diagram is a simplified one. The details of the authentication process are reported in deliverable [D3.4].
- **step 2:** the citizen sends a request to the R2D Access server of a HCO to download his or her health data. This step is executed invoking one of the operations defined by R2D Access (defined later). The overall process is the same whatever is the invoked operation.
    - **step 2.1:** the R2D Access server checks if the provided eIDAS token is valid.
    - **step 2.2:** if the token is not valid, the server replies with a NOT AUTHENTICATED (HTTP STATUS 401) response and the processing of request is finished.
    - **step 2.3:** if the token is valid, the server replies with a REQUEST ACCEPTED outcome (HTTP STATUS 202). In case the client is submitting too many requests in the unit of time, the server can reject them using the following HTTP status: 429 Too Many Requests.

- **step 3 and step 4:** These steps are executed if the initial request has been accepted by the server (return code 202). In this case the S-EHR starts a polling loop for periodically checking the status of the request processing. When the request returns a code different from 200, the loop ends and the S-EHR can process the return code.
    - **step 5:** if the polling returns a status of 200, it means that the requested data is available, so the S-EHR uses the returned URLs to retrieve the health data.
    - **step 6**: the S-EHR app stores locally the retrieved health data.
    - **step 7**: if the polling returns a status not equals to 200, it means that the server raised an error while retrieving the requested data. The S-EHR app shows the error to the citizen.
- **step 8**: this step is executed if the first initial request was not accepted by the server. The S-EHR app shows the error to the citizen.

[rfc 7240] states that the asynchronous processing is requested by the client using the `Prefer` header parameter with the value `respond-async`. A server receiving the `Prefer` header parameter can choose if to honour or not the asynchronous processing requested by the client. This implies that servers should be able to use both patterns, synchronous and asynchronous depending on how clients request data. This complete freedom of choice (given to the client) does not suit well with the objective of easing the implementation of the R2D Access, because it implies that both the S-EHR and the R2D Access server should be able to process requests using both patterns (synchronous and asynchronous). An R2D Access server MUST support the asynchronous request processing pattern as the default one. Implementers of R2D Access are free to offer ALSO the synchronous processing pattern if they consider it advantageous to them, but if the `Prefer` header attribute specified by S-EHR requires an asynchronous interaction then the server MUST honour the request.

From the client point of view, it means that if a client requests data synchronously (without using the `Prefer` header parameter) it is NOT guaranteed that it will be processed synchronously. The R2D Access MAY choose to process the request asynchronously. When submitting a synchronous request to a R2D Access server, S-EHR apps have to evaluate the return code to determine if the request has been handled synchronously or asynchronously by the server. The 202 return code means that the request will be handled asynchronously, while the 200 return code means that the request has been (synchronous) served successfully.

4.2.2 Citizen identification

R2D Access requires authenticating any citizen by means of an eIDAS token transmitted in the HTTP request as an header parameter with name `Authorization` (the value of the parameter must be preceded by "`Bearer `"). An R2D Access server that supports all the operations defined in this deliverable, but does not use an identity manager other than eIDAS, MUST NOT be considered compliant to this specification.

The eIDAS SAML attribute profile specification [EIDAS_SAML] defines the following list of attributes for an eIDAS token identifying a natural person:

Attribute	Support
FamilyName	MANDATORY
FirstName	MANDATORY
DateOfBirth	MANDATORY
PersonIdentifier	MANDATORY
Gender	OPTIONAL
CurrentAddress	OPTIONAL
PlaceOfBirth	OPTIONAL
BirthName	OPTIONAL

Regarding the attribute PersonIdentifier, the [EIDAS_SAML] states the following declaration:

> "The uniqueness identifier consists of:
>
> 5. The first part is the Nationality Code of the identifier
>   a. This is one of the ISO 3166-1 alpha-2 codes, followed by a slash ("/"))
> 6. The second part is the Nationality Code of the destination country or international organization
>   a. This is one of the ISO 3166-1 alpha-2 codes, followed by a slash ("/")
> 7. The third part a combination of readable characters
>   a. This uniquely identifies the identity asserted in the country of origin but does not necessarily reveal any discernible correspondence with the subject's actual identifier (for example, username, fiscal number, etc.)
>
> Example: ES/AT/02635542Y (Spanish eIDNumber for an Austrian SP)"

eIDAS does not provide any national identifier (i.e. identifier already used at national level, such as the national social number of the ID card number) in the minimum mandatory data set, but uses its own code composed of three parts: the first identifies the nationality of the citizen, the second identifies the requesting country and the third is a constant unique code. The lack in the minimum dataset of a national identifier and the absence of an official European service for mapping national and eIDAS identifiers creates a potential problem of citizen identification for software services accepting eIDAS tokens. We expect that in

the future most healthcare organisations will know the eIDAS identifier of their patients, but this is not yet the current situation.

It is very likely that a citizen from Country A that receives care in a hospital of Country B, was registered by the HCO using demographic data (name, surname, date of birth, place of birth) and maybe passport number, therefore the mandatory dataset provided by eIDAS does not guarantee the exact match especially in some unlikely, but possible situations. This is the reason why, before allowing the access to the services of an R2D Access server, the providing HCO, MAY require some additional identification data to the citizen, to help the HCO to associate in a more secure way the requesting citizen to one of the patients in their EHR. This identification process IS NOT mandatory because some hospitals may already have collected the eIDAS identity of their patients.

Requiring additional identification data is a process executed by the HCO following the security policies defined by the HCO itself. The deployment (by two distinguished HCOs) of two R2D Access instances implementing two different identification policies is possible and completely reasonable.

So, the first time an authenticated (by eIDAS) citizen submit a request to an R2D Access server, it MAY be required to the citizen to perform a one-time operation to send additional identification data to help the HCO to associate the requesting citizen to a patient in the HCO's EHR. In this case the R2D Access server replies with the HTTP status code 403 (FORBIDDEN) adding to the response an header attribute named `requested-identification-attrs` containing the comma separated list of the requested identification attributes names. Following is the list of the defined (case sensitive) names for the identification attributes:

- `FiscalNumber`
- `Email`
- `MobilePhone`
- `IdentityCard.number`
- `IdentityCard.issuer`
- `IdentityCard.issueDate` (allowed format YYYY + "-" + MM + "-" + DD)
- `IdentityCard.expiryDate` (allowed format YYYY + "-" + MM + "-" + DD)
- `IdentityCard.frontPhoto` (represented as jpeg)
- `IdentityCard.backPhoto` (represented as jpeg)
- `Gender` (possible values: Male | Female | Not Specified)
- `PlaceOfBirth.name`
- `PlaceOfBirth.postCode`
- `PlaceOfBirth.district`
- `PlaceOfBirth.country`

The HCO is free to choose the set of attributes that should be required to the citizen depending only on its internal security policies and on the concrete data contained by the eIDAS token (mandatory attributes and values of optional attributes). The following sequence diagram (Figure 19) is a further development of the previous one (in red are shown the additions), showing the citizen identification process:

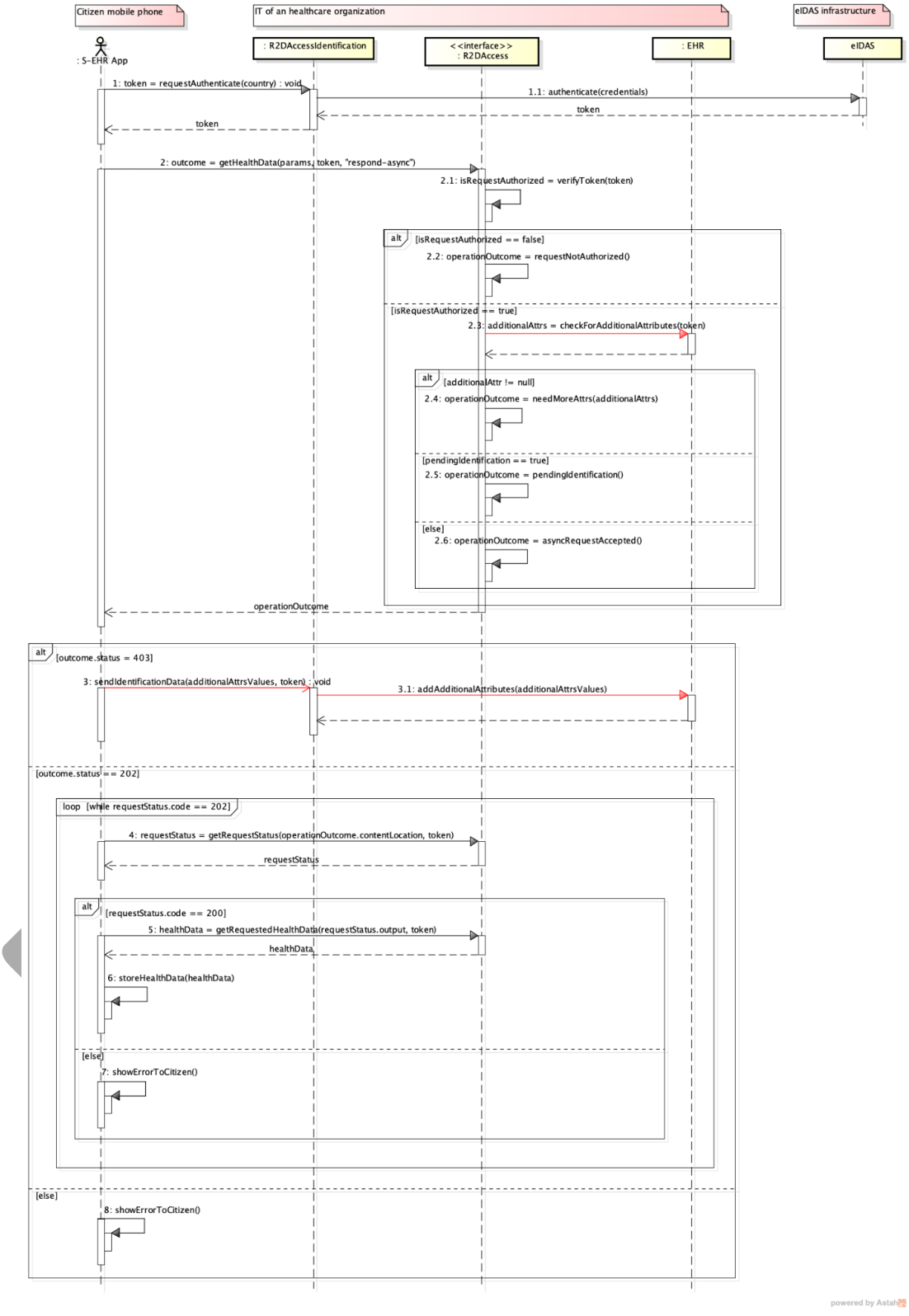

Figure 19 - Citizen identification process

- **step 1:** the citizen authenticates to eIDAS. The authentication process represented in this diagram is a simplified one, it does not represent the real process that is executed to authenticate the citizen. This objective is out of the scope of this deliverable, but it is a subject of the deliverable [D3.4].

- **step 2:** the citizen sends a request to the R2D Access server of a HCO to download his/her health data. This step is executed invoking one of the operations defined by R2D Access (defined later), for the purposes of this sequence diagram it is not useful to identify one specific operation, because invoking one service or another does not affect the overall structure of this process.
    - **step 2.1:** the R2D Access server checks if the provided eIDAS token is valid.
    - **step 2.2:** if the token is not valid, the server replies with the HTTP status code 401 (NOT AUTHENTICATED) optionally adding a FHIR OperationOutcome providing a human readable error message. The processing of the request is finished.
    - **step 2.3:** if the token is valid the R2D Access server sends a request to the EHR requesting the identification of the requesting citizen. In case the client is submitting too many requests in the unit of time, the server is able to reject them using the HTTP status code 429 (TOO MANY REQUESTS) optionally adding a FHIR OperationOutcome providing a human readable error message.
        - **step 2.4:** if it is not possible for the HCO to uniquely identify the citizen, or if the hospital needs additional identifying information, the server MUST reply with the HTTP status code 403 (FORBIDDEN) adding the response an header attribute named `requested-identification-attrs` containing the comma separated list of the requested identification attributes names and optionally adding a FHIR OperationOutcome providing a human readable error message.
        - **step 2.5:** The identification process executed by the HCO cannot be considered as completely automatic, but it is very likely that it requires human intervention. During the execution of this process the client may still try to access R2D Access services, in this case the server MUST reply with the HTTP status code 403 (FORBIDDEN) adding to the response the header attribute named `pending-identification` set to true and optionally adding a FHIR OperationOutcome providing a human readable error message.
        - **step 2.6:** if the token is valid, the server replies with the HTTP status code of 202 (REQUEST ACCEPTED) and optionally adding a FHIR OperationOutcome providing a human readable error message.
- **step 3:** if the initial request has been accepted by the server (return code 202), the S-EHR starts a polling loop for periodically checking the status of the request processing: Until the returned status is 202, the polling loop goes on.
- **step 4:** if the polling returns a status of 200, it means that the requested data are available, so the S-EHR uses the returned URLs to retrieve the health data.
    - **step 5**: the S-EHR app stores the retrieved health data.
- **step 6**: if the polling returns a status of 500, it means that the server raised an error while retrieving the requested data. The S-EHR app shows the error to the citizen.
- **step 7**: this step is executed if the first initial request was not accepted by the server. The S-EHR app shows the error to the citizen.

4.3 R2D Access Specifications

The objective of this section is to define the technical specification of the RESTful operations that compose the offered interfaces of an R2D Access server. These interfaces, introduced in the overall InteropEHRate architecture deliverable [D2.6], are called R2DAccess, R2DIdentification and R2DAccessDICOM as shown by the following diagram (Figure 20).

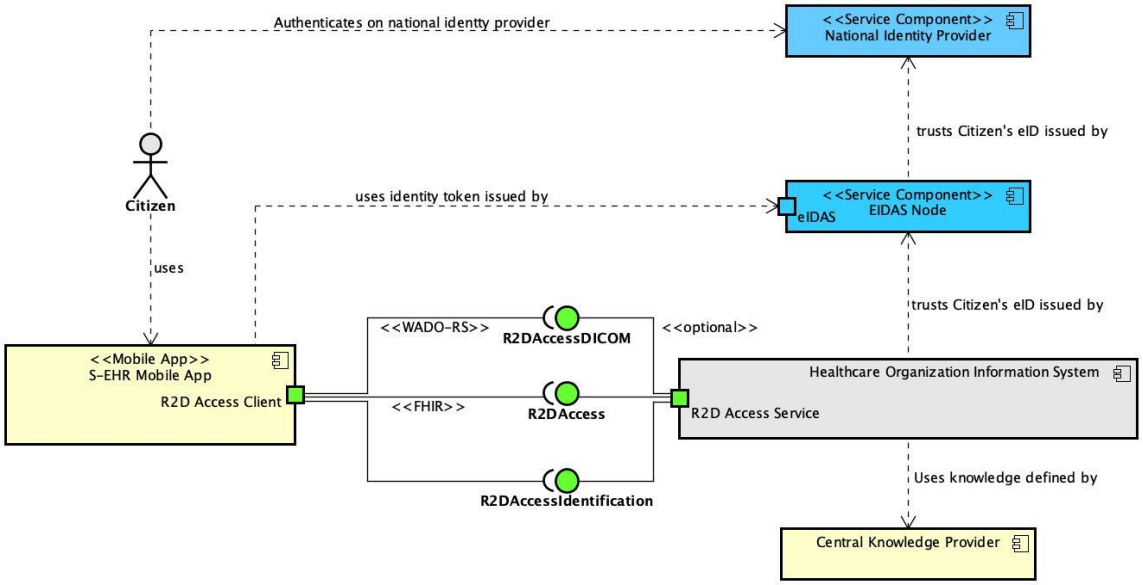

Figure 20 - R2DAccess Service interfaces

The next class diagram (Figure 21) shows a logical view of the R2DAccess interface providing a first initial list of the methods it defines:

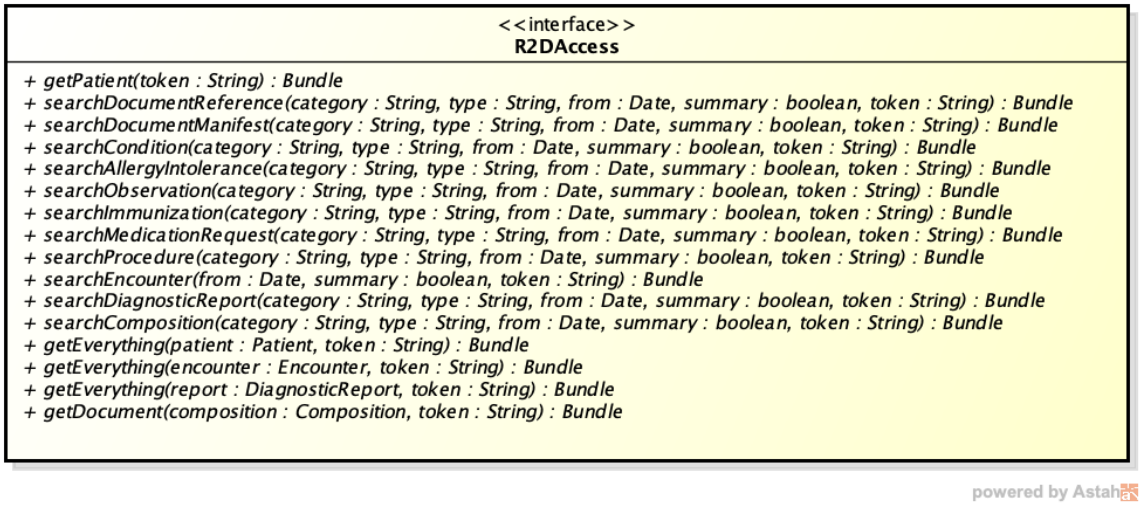

Figure 21 - The R2D Access interface

As already stated, R2D Access is a RESTful protocol. Every operation defined by the R2D Access protocol is described in a specific section of this chapter using an ad hoc template for RESTful services, the allows to describe the following features:

- the HTTP method to be used;
- the URL to be invoked;
- the type of the FHIR resource that is the subject of the request;
- the FHIR profile of the returned data;
- the allowed parameters that may be added to the URL and the constraints defined for the allowed values of each parameter;
- the set of allowed header parameters and the constraints defined for the allowed values of each parameter;
- the set of possible HTTP return codes.

Following is the list of the FHIR resources supported by R2D Access protocol. Every resource in the list MUST be queryable by any S-EHR using the search parameters defined in the next sections:

- Patient
- DocumentReference
- DocumentManifest
- DiagnosticReport
- MedicationRequest
- Condition
- Immunization
- AllergyIntolerance
- Observation
- Encounter
- Composition
- Procedure

The aforementioned list is only the list of the queryable resources, it is not the list of the resource types downloadable with R2D Access. For instance, while downloading an Encounter it will be possible to download also all the resources related by that instance of Encounter using specific operations to retrieve all the health data related to a main one with a single action. The entire set of exchangeable resources is defined by the InteropEHRate FHIR profiles [D2.9].

From a conceptual point of view, the set of resources queryable by R2D Access can divided into two main classes:

- Atomic resources: Condition, AllergyIntolerance, Immunization, MedicationRequest and DocumentReference. Most of these resource types are part of R2D Access, because they are considered relevant from a medical point of view in a cross border health data exchange context. In fact, they correspond also to required sections of the IPS specifications (Condition, MedicationRequest, AllergyIntolerance, Immunization, Procedure), or to health data exchanged by eHDSI (PatientSummary, MedicationRequest).
- Container resources: DocumentManifest, DiagnosticReport, Encounter, Composition and Patient. These resources allow an organized download of health data by an R2DAccess Server following a two-steps pattern: firstly it is submitted a query for retrieving the list of available resources of a specific content type (e.g. “give me all my Encounters”) then, using specific operations (that work at instance level), it is possible to retrieve all the related health data (e.g. “give me all the data related to the Encounter X”). This pattern allows the retrieval of health data of citizens giving them the possibility to choose what to download.

Observations can be either an atomic or a container resource depending on the specific usage.

The following table contains the set of the operations that compose the R2D Access protocol:

Operation
Retrieval of the Patient
Search of Condition of the patient
Search of AllergyIntolerance of the patient
Search of Immunization of the patient
Search of Procedure of the patient
Search of Observation of the patient
Search of MedicationRequest of the patient
Search of DocumentReference of the patient
Search of DocumentManifest of the patient
Search of DiagnosticReport of the patient
Search of Encounter of the patient
Search of Composition of the patient
Retrieval of every health data of the patient
Retrieval of every health data related to an Encounter of the patient
Retrieval of every health data related to an DiagnosticReport of the patient
Retrieval of every health data related to a Composition of the patient

These R2D Access specification mandates the use of the FHIR profiles defined by the InteropEHRate project, documented by deliverable [D2.9]. This deliverable and deliverable [D2.9] provide together all the technical information needed to implement the R2D Access protocol and to guarantee the interoperability among any compliant S-EHR app and any compliant R2D Access Service. They define respectively: i) how two systems interact to exchange health data, ii) what is the structure of the health data that can be exchanged between the two.

4.3.1 URI schemas

As stated before, R2D Access is a FHIR protocol specified by applying restrictions and constraints to the standard HL7 FHIR RESTFul API [FHIR API SPEC], without adding any new additional parameter. R2D Access 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 about the patient) and are used mainly for periodic import of citizen's health data from an EHR to their mobile device. These kinds of import operations do not require all the capabilities provided by the complete [FHIR API SPEC].

RESTful services can be invoked manually by specific HTTP clients (every known browser, CURL, Postman, etc. etc.), or programmatically from almost every programming language that provides support for web programming: Java, Javascript, Python, PHP. The most generic form of a R2D Access request is the following:

```
GET [base]/[type]{?paramsAndOptions}
```

names surrounded by [ ] are placeholders for mandatory parameters, names surrounded by { } are placeholders for repeatable parameters, while '?' prefixes placeholders for optional parameters.

- **GET**: is the HTTP method used to submit a read or search operation. it is the only method supported by R2D Access (it is a read only protocol).
- **base**: corresponds to the endpoint of the server providing the R2D Access service.
- **type**: must be one of the queryable FHIR types supported by R2D Access and listed in the previous section.
- The `{?paramsAndOptions}` tag is the optional part of a R2D Access request. It contains a series of encoded name-value pairs separated by the `&` char and represents the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value.

4.3.2 Supported HTTP methods

R2D Access is a read only protocol used by citizens, thus the only supported HTTP method for citizens is the GET method. Incoming requests submitted by citizens using one of the following HTTP methods POST, PUT and DELETE, MUST NOT BE AUTHORIZED.

R2D Access implementers are free to provide support also to POST, PUT and DELETE methods to provide WRITE operations to administrators or other privileged services.

4.3.3 Supported interactions

The FHIR specification [FHIR API SPEC] defines different kinds of conceptual interactions, concretely implemented by combining HTTP methods and specific standard URL schemas. The following table shows the level of support that R2D Access must provide for each kind of interaction defined by FHIR (Providing support also for parameters marked as HAVE NOT TO BE SUPPORTED is not considered a violation to the protocol):

Whole System Interactions		
Interaction	**Description**	**Support**
capabilities	Retrieve the capability statement of the R2D Access server	SUPPORTED
batch/transaction	Update, create or delete a set of resources in a single interaction	HAVE NOT TO BE SUPPORTED

search	Search across all resource types based on some filter criteria	HAVE NOT TO BE SUPPORTED
history	Retrieve the change history for all resources	NOT SUPPORTED

Type Level Interactions		
Interaction	**Description**	**Support**
create	Create a new resource with a server assigned id	HAVE NOT TO BE SUPPORTED
search	Search the resource type based on some filter criteria	SUPPORTED
history	Retrieve the change history for a particular resource type	HAVE NOT TO BE SUPPORTED

Instance Level Interactions		
Interaction	**Description**	**Support**
read	Read the current state of the resource	HAVE NOT TO BE SUPPORTED
vread	Read the state of a specific version of the resource	HAVE NOT TO BE SUPPORTED
update	Update an existing resource by its id (or create it if it is new)	HAVE NOT TO BE SUPPORTED
delete	Delete a resource	HAVE NOT TO BE SUPPORTED
patch	Update an existing resource by posting a set of changes to it	HAVE NOT TO BE SUPPORTED
history	Retrieve the change history for a particular resource	HAVE NOT TO BE SUPPORTED
operation	Execute an operation over a resource	SUPPORTED only for read-only operations defined below.

According to FHIR specifications [FHIR API SPEC], the name of these interactions is the same name that MUST be used in the capability statement of the R2D Access Server to state what operations are supported for every resource type (see section Capability Statement).

4.3.4 Supported common search parameter and search options

FHIR defines a set of common (to all the resource types) search parameters and search options (https://www.hl7.org/fhir/searchparameter-registry.html#common), this section lists which common parameters or options MUST be supported by any implementation of R2D Access. The following table show the level of support provided by R2D Access for common search options:

Search Parameter	R2D Access support
_sort	SUPPORTED
_count	SUPPORTED
_summary	SUPPORTED

4.3.5 Search Narrowing

R2D Access is designed to be used only by citizens that can only request their own data, they cannot request for data of another citizen. For this reason, all the R2D Access operations always imply an (eIDAS) authenticated citizen that must match to an existing patient of the healthcare organization providing the R2D Access server. For this reason, every R2D Access operation is always (and only) executed in the compartment of the authenticated citizen. The following request:

```
[base]/DiagnosticReport?code=XYZ
```

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

```
[base]/DiagnosticReport?subject=<auth citizen>&code=XYZ
```

This feature of the R2D Access protocol does not force S-EHR apps to provide the filtering criteria for the patient when invoking an R2D Access operation so that the query is simpler.

From the other side, the R2D Access server MUST process every incoming request according to the following possible scenarios:

- The incoming request does not contain a filter criteria for the patient: the R2D Access server MUST process it ONLY in the compartment of the authenticated citizen. R2D Access server implementers are free to develop a proprietary mechanism to narrow the search to the authenticated citizen without explicitly affecting the incoming query or as an alternative CAN pre-process the query adding the missing parameter and binding it to the identifier of the authenticated citizen.
- The incoming request contains a filter criteria for the patient: the R2D Access server MUST reject this request using the HTTP status code 403 (Forbidden). The reason why a request containing a filter parameter for the patient MUST be rejected is that the only way for a

client to communicate who is the authenticated citizen, is the eIDAS token, Any other option using the query parameter MUST be considered as a not secure and not compliant request, thus MUST be rejected.

- The incoming request does not contain the eIDAS token: the R2D Access server MUST reject this request using the HTTP status code 401 (Not Authorized).

Complete technical details about citizen identity and session management are specified in the deliverable [D3.4].

Search narrowing is not only applied to the authenticated citizen, but also to the status of the requested health data. Health data returned by R2D Access MUST be only those whose is a final one. For instance, temporary health data that still must be processed by clinicians MUST NOT be returned to the citizen. As the status attribute has different names and different values for each resource type, this detailed specification for every resource type can be found in the dedicated paragraph in the following sections.

4.3.6 Interface R2DAccess

This section contains the technical specifications of every operation of the R2D Access protocol introduced in the previous sections. The R2D Access is a protocol based on the HTTP specifications version 1.1, providing details about these standards is out of the scope of this document, complete knowledge of HTTP specifications [rfc 2616] [rfc 7540] is required for a complete understanding of the present deliverable.

4.3.6.1 *Search Patient*

The Patient resource in FHIR is used to represent demographics and other administrative information about an individual receiving care or other health-related services. Full information about the Patient resource can be found at this URL: https://www.hl7.org/fhir/patient.html.

The search operation over the Patient resource is part of the specifications of R2DAccess, but due to the search narrowing applied over the authenticated citizen, this search can return only the record of the Patient corresponding to the authenticated citizen. This is the reason why this search does define any search parameters available to the S-EHR app.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/Patient
```

Parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing one instance of Patient corresponding to the authenticated citizen. All the resources contained in the bundle are represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.2 *Search DocumentReference*

The DocumentReference is the FHIR resource used to represent a reference to a document of any kind, it provides metadata about the document so that the document can be discovered and managed. The scope of a document is any serialized object with a mime-type, so includes formal patient centric documents (CDA), clinical textual notes, scanned paper, PDF documents, images, etc. etc. Full information about the DocumentReference resource can be found at this URL: https://www.hl7.org/fhir/documentreference.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/DocumentReference{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the DocumentReference resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
category	token	NO	http://hl7.org/fhir/ValueSet/document-classcodes
type	token	NO	
date	date	NO	
_sort	text	NO	
_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	reference	YES	Authenticated citizen

status	token	YES	“current”

The presence of two implicit parameters under R2D Access control (subject and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of DocumentReference MUST be executed narrowing the results to DocumentReference whose `status` is equal to the literal “current” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/DocumentReference`
2. `[base]/DocumentReference?subject=<auth-ctz>&status=current`

R2D Access implementers MUST execute every search of DocumentReference considering that the `subject` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/DocumentReference?type=http://loinc.org|60591-5

[base]/DocumentReference?date=gt2019-01-01&date=lt2019-31-12&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

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: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.3 *Search DocumentManifest*

The DocumentManifest is the FHIR resource used to group a set of resources (pertaining to the same patient) that share a clinical context. Typically, DocumentManifest resources are used in document indexing systems like IHE XDS (https://wiki.ihe.net/index.php/Cross-Enterprise_Document_Sharing) or IHE MHD (https://wiki.ihe.net/index.php/Mobile_access_to_Health_Documents_(MHD)). Full information about the DocumentManifest resource can be found at this URL: https://www.hl7.org/fhir/documentmanifest.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentManifest resource:

```
GET [base]/DocumentManifest{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the DocumentManifest resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
type	token	NO	http://terminology.hl7.org/ValueSet/v3-ActCode
created	date	NO	
_sort	text	NO	
_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	reference	YES	Authenticated citizen
status	token	YES	“current”

The presence of two implicit parameters under R2D Access control (subject and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of DocumentManifest MUST be executed narrowing the results to DocumentReference whose `status` is equal to the literal “current” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/DocumentManifest`
2. `[base]/DocumentManifest?subject=<auth-ctz>&status=current`

R2D Access implementers MUST execute every search of DocumentManifest considering that the `subject` and `status` search parameters are mandatory and under the responsibility of the R2D Access

server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentManifest following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/DocumentManifest?created=gt2019-01-01&date=lt2019-31-12&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing instances of DocumentManifest represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.4 *Search Condition*

The Condition is the FHIR resource used to represent a record of detailed information about a condition, problem, diagnosis, or other event, situation, issue, or clinical concept that has risen to a level of concern. Full information about the Condition resource can be found at this URL: https://www.hl7.org/fhir/condition.html.

The resource Condition is part of the mandatory data of the International Patient Summary specifications, it has been added to the R2D Access protocol in order to let citizens download their set of Condition even if the HCOs used by the citizen do not provide a complete Patient Summary.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the Condition resource:

```
GET [base]/Condition{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the Condition resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
category	token	NO	http://hl7.org/fhir/ValueSet/condition-category
code	token	NO	http://hl7.org/fhir/ValueSet/condition-code
recorded-date	date	NO	
_sort	text	NO	

_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	Reference	YES	Authenticated citizen
verification-status	Token	YES	“confirmed”

The presence of two implicit parameters under R2D Access control (subject and verification-status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of DocumentReference MUST be executed narrowing the results to DocumentReference whose `verification-status` is equal to the literal “confirmed” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/Condition`
2. `[base]/Condition?subject=<auth-ctz>&verification-status=confirmed`

R2D Access implementers MUST execute every search of DocumentReference considering that the `subject` and `verification-status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/Condition?code=http://snomed.info/sct|45234553

[base]/Condition?category=problem-iten-list&date=gt2019-01-01&date=lt2019-31-12
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

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: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Some additional notes about the Condition resource:

- Condition defines two attributes for storing dates: one is `recorded-date` and the other one is `onset`. The attribute `recorded-date` stores the date in which the Condition was first recorded by the system, while the `onset` refers to when the Condition is supposed to begin, or was discovered. It is very difficult or almost impossible to determine with certainty the onset of a certain condition, this is the reason why this attribute may contain an incomplete date, or simply an year or the age (of the patient) when the condition was discovered. Due to this uncertainty this attribute cannot be used as a search parameter in R2D Access and the `recorded-date` was preferred.

- The Condition defines two status attributes, one is the `clinical-status` and the other one is the `verification-status`. The `clinical-status` states if the Condition is active or not, while the `verification-status` states if this condition was confirmed by a clinician. In R2D Access only Condition whose `verification-status` is equal to "confirmed" are returned to S-EHR, so it is the responsibility of the S-EHR to check the value of the `clinical-status` attribute to determine if the Condition is still active.

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.5 *Search AllergyIntolerance*

The AllergyIntolerance is the FHIR resource used to represent a record of a clinical assessment of an allergy or intolerance; a propensity, or a potential risk to an individual, to have an adverse reaction on future exposure to the specified substance, or class of substance. Full information about the AllergyIntolerance resource can be found at this URL: https://www.hl7.org/fhir/allergyintolerance.html.

Moreover, the resource AllergyIntolerance is part of the mandatory data of the International Patient Summary specifications, it has been added to the R2D Access protocol in order to let citizens download their set of AllergyIntolerance even if the HCOs used by the citizen do not provide a complete Patient Summary.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the AllergyIntolerance resource:

```
GET [base]/AllergyIntolerance{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the AllergyIntolerance resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
category	token	NO	http://hl7.org/fhir/ValueSet/allergy-intolerance-category

type	token	NO	http://hl7.org/fhir/ValueSet/allergy-intolerance-type
date	date	NO	
_sort	text	NO	
_count	integer	NO	
_summary	text	NO	"true"|"false"|"text"

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	token	YES	Authenticated citizen
verification-status	token	YES	"confirmed"

The presence of two implicit parameters under R2D Access control (subject and verification-status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of AllergyIntolerance MUST be executed narrowing the results to AllergyIntolerance whose `verification-status` is equal to the literal "current" and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/AllergyIntolerance`
2. `[base]/AllergyIntolerance?subject=<auth-ctz>&verification-status=confirmed`

R2D Access implementers MUST execute every search of AllergyIntolerance considering that the `subject` and `verification-status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/AllergyIntolerance?type=allergy

[base]/AllergyIntolerance?date=category=food&gt2019-01-01&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing instances of AllergyIntolerance represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Some additional notes about the AllergyIntolerance resource:

- AllergyIntolerance defines two attributes for storing dates: one is `recorded-date` and the other one is `onset`. The attribute `recorded-date` stores the date in which the AllergyIntolerance was

first recorded by the system, while the `onset` refers to when the AllergyIntolerance is supposed to begin, or was discovered. It is very difficult or almost impossible to determine with certainty the onset of a certain allergy, this is the reason why this attribute may contain an incomplete date, or simply an year or the age (of the patient) when the condition was discovered. Due to this uncertainty this attribute cannot be used as a search parameter in R2D Access and the `recorded-date` was preferred.

- The AllergyIntolerance defines two status attributes, one is the `clinical-status` and the other one is the `verification-status`. The `clinical-status` states if the AllergyIntolerance is active or not, while the `verification-status` states if this condition was confirmed by a clinician. In R2D Access only AllergyIntolerance whose `verification-status` is equal to "confirmed" are returned to S-EHR, so it is the responsibility of the S-EHR to check the value of the `clinical-status` attribute to determine if the AllergyIntolerance is still active.

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.6 *Search Observation*

The Observation is one of the most used FHIR resource and it is used to record measurements and simple assertions made about a patient, device or other subject. Observations are quite used for diagnosis, they can be grouped (one Observation may contain several other Observations) and moreover they can be referenced by a DiagnosticReport to represent, for instance, a laboratory report. Full information about the Observation resource can be found at this URL: https://www.hl7.org/fhir/observation.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the Observation resource:

```
GET [base]/Observation{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the Observation resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**

category	token	NO	http://hl7.org/fhir/ValueSet/observation-category
code	token	NO	http://hl7.org/fhir/ValueSet/observation-codes
date	date	NO	
_sort	text	NO	
_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
patient	token	YES	Authenticated citizen
status	token	YES	“final”

The presence of two implicit parameters under R2D Access control (subject and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of Observation MUST be executed narrowing the results to Observation whose `status` is equal to the literal “final” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/Observation`
2. `[base]/Observation?patient=<auth citizen>&status=final`

R2D Access implementers MUST execute every search of Observation considering that the `patient` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/Observation?code=http://loinc.org|29463-7

[base]/Observation?dcategory=vital-signs&date=gt2020-01-01&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing instances of Observation represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.7 *Search Immunization*

Immunization is the FHIR resource used to represent the recording of current and historical administration of vaccines to patients across all healthcare disciplines in all care settings and all regions. Full information about the Immunization resource can be found at this URL: https://www.hl7.org/fhir/immunization.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/Immunization{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the Immunization resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
date	date	NO	
_sort	text	NO	
_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**

patient	token	YES	Authenticated citizen
status	token	YES	“completed”

The presence of two implicit parameters under R2D Access control (patient and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of DocumentReference MUST be executed narrowing the results to DocumentReference whose `status` is equal to the literal “complete” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/Immunization`
2. `[base]/Immunization?patient=<auth citizen>&status=complete`

R2D Access implementers MUST execute every search of Immunization considering that the `patient` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/Immunization?date=gt2019-01-01&date=lt2019-31-12&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing instances of Immunization represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.8 *Search MedicationRequest*

MedicationRequest is the FHIR resource used to represent an order or request for both supply of the medication and the instructions for administration of the medication to a patient. Full information about the MedicationRequest resource can be found at this URL: https://www.hl7.org/fhir/medicationrequest.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the MedicationRequest resource:

```
GET [base]/MedicationRequest{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the MedicationRequest resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
category	token	NO	http://hl7.org/fhir/ValueSet/medicationrequest-category
date	date	NO	
_sort	text	NO	
_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	token	YES	Authenticated citizen
status	token	YES	“active”

The presence of two implicit parameters under R2D Access control (subject and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of DocumentReference MUST be executed narrowing the results to DocumentReference whose `status` is equal to the literal “active” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/MedicationRequest`
2. `[base]/MedicationRequest?subject=<auth citizen>&status=active`

R2D Access implementers MUST execute every search of DocumentReference considering that the `subject` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/MedicationRequest?category=inpatient

[base]/MedicationRequest?date=gt2019-01-01&date=lt2019-31-12&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

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: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.

- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.9 *Search Procedure*

Procedure is the FHIR resource used to represent the details of current and historical procedures performed on or for a patient. A procedure is an activity that is performed on, with, or for a patient as part of the provision of care. Full information about the Procedure resource can be found at this URL: https://www.hl7.org/fhir/procedure.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the Procedure resource:

```
GET [base]/Procedure{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the Procedure resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
category	token	NO	http://hl7.org/fhir/ValueSet/procedure-category
code	token	NO	http://hl7.org/fhir/ValueSet/procedure-code
date	date	NO	
_sort	text	NO	

_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	token	YES	Authenticated citizen
status	token	YES	“active”

The presence of two implicit parameters under R2D Access control (subject and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of Procedure MUST be executed narrowing the results to Procedure whose `status` is equal to the literal “completed” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/Procedure`
2. `[base]/Procedure?subject=<auth citizen>&status=completed`

R2D Access implementers MUST execute every search of Procedure considering that the `subject` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/Procedure?category=http://snomed.info|387713003

[base]/Procedure?date=gt2019-01-01&date=lt2019-31-12&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing instances of Procedure represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.10 *Search Encounter*

Encounter is the FHIR resource used to represent an interaction between a patient and healthcare provider for the purpose of providing healthcare service or assessing the health status of a patient. Full information about the Encounter resource can be found at this URL: https://www.hl7.org/fhir/encounter.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/Encounter{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the Encounter resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
date	Date	NO	
_sort	Text	NO	
_count	Integer	NO	
_summary	Text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	token	YES	Authenticated citizen

The presence of one implicit parameter under R2D Access control (subject), strongly modifies the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of Encounter MUST be executed narrowing the results to Encounter that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/DocumentReference`

2. `[base]/DocumentReference?subject=<auth citizen>`

R2D Access implementers MUST execute every search of Encounter considering that the `subject` search parameters is mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of Encounter following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/Encounter?date=gt2019-01-01&date=lt2019-31-12&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing instances of Encounter represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.

- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.11 *Search DiagnosticReport*

DiagnosticReport is the FHIR resource used to represent the findings and interpretation of diagnostic tests performed on patients. Full information about the DiagnosticReport resource can be found at this URL: https://www.hl7.org/fhir/diagnosticreport.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DiagnosticReport resource:

```
GET [base]/DiagnosticReport{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the DiagnosticReport resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
category	token	NO	http://hl7.org/fhir/ValueSet/diagnostic-service-sections
code	token	NO	
date	date	NO	
_sort	text	NO	

_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**
subject	token	YES	Authenticated citizen
status	token	YES	“final”

The presence of two implicit parameters under R2D Access control (subject and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of DocumentReference MUST be executed narrowing the results to DiagnosticReport whose `status` is equal to the literal “final” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/DiagnosticReport`
2. `[base]/DiagnosticReport?subject=<auth citizen>&status=final`

R2D Access implementers MUST execute every search of DiagnosticReport considering that the `subject` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DiagnosticReport following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/DiagnosticReport?category=RX|LAB

[base]/DiagnosticReport?date=gt2019-01-01&date=lt2019-31-12&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

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: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.12 *Search Composition*

Composition is the FHIR resource used to represent a set of healthcare-related information that is assembled together into a single logical document that provides a single coherent statement of meaning, establishes its own context and that has clinical attestation with regard to who is making the statement. The Composition resource provides the basic structure of a FHIR document. Full information about DocumentReference can be found at this URL: https://www.hl7.org/fhir/composition.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/Composition{?paramsAndOptions}
```

The `{?paramsAndOptions}` tag contains a series of encoded name-value pairs separated by the `&` char, representing the filter criteria for the query, as well as control parameters to modify the behaviour of the R2D Access server such as sorting the results or limiting the number of results to a specific value. Unlike the FHIR specifications, R2DAccess only supports a small subset of filtering criteria for the Composition resource. The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
category	token	NO	http://hl7.org/fhir/ValueSet/document-classcodes
type	token	NO	
date	date	NO	
_sort	text	NO	
_count	integer	NO	
_summary	text	NO	“true”|”false”|”text”

Search parameters under R2D Access control			
Name	**Type**	**Mandatory**	**Constraint**

subject	token	YES	Authenticated citizen
status	token	YES	“final”

The presence of two implicit parameters under R2D Access control (subject and status), strongly modify the semantic of the search operation as it is intended in the FHIR specifications. It implies that every search of Composition MUST be executed narrowing the results to Composition whose `status` is equal to the literal “final” and that belongs to the authenticated citizen. R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/Composition`
2. `[base]/Composition?subject=<auth citizen>&status=final`

R2D Access implementers MUST execute every search of Composition considering that the `subject` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Examples of valid queries:

```
[base]/Composition?type=http://loinc.org|60591-5

[base]/Composition?category=http://loinc.org|18842-5&sort=-date
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`

Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing instances of Composition represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

4.3.6.13 *Get everything of a Patient*

This operation is used to return all the information related to a Patient. This operation is considered the simplest way to import health data of a citizen using R2D Access. The response of this operation is a bundle of type "searchset" containing, at a minimum, the patient resource itself returned along with any other resources that the server has available for the given patient.

The basic idea behind this operation is to provide to the S-EHR a single way to retrieve all the health data produced during an Encounter in a single bundle without pagination. Moreover, using operations instead of a complex query on the Patient compartment moves the responsibility from the client to the server: it is a responsibility of the R2D Access server to search, select and group all the data related to the Patient that are considered relevant for the context of the encounter.

Full information about the operation Patient/$everything can be found at this URL: https://www.hl7.org/fhir/patient-operation-everything.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/Patient/$everything
```

The following tables show the list of the allowed parameters a S-EHR MAY use, and a R2D Access serve MUST process:

Search parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
start	date	NO	
end	date	NO	

The only instance of a Patient that can be used to invoke this operation is the one related to the authenticated citizen. For this reason R2D Access implementers MUST guarantee the semantic equivalence between these two queries:

1. `[base]/Patient/$everything`
2. `[base]/Patient/<PatientId>/$everything`

R2D Access implementers MUST execute every search of DocumentReference considering that the `subject` and `status` search parameters are mandatory and under the responsibility of the R2D Access server itself and not of the S-EHR. R2D Access implementers are free to develop their proprietary solution to restrict searches of DocumentReference following the criteria stated in these specifications.

Parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing heterogeneous instances of FHIR Resource (at least the Encounter resource). All the resources contained in the bundle are represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.14 *Get everything of an Encounter*

This operation is used to return all the information related to an instance of Encounter. Almost every health data of a citizen is produced during an encounter between the citizen and a healthcare organization, so using the Encounter resource to fetch the health data produced during an Encounter is considered one of the simplest ways to import health data with R2D Access. The response of this operation is an heterogeneous bundle of type "searchset" containing at a minimum, the encounter resource itself returned along with any other resources that the server has available for the given encounter.

The basic idea behind this operation is to provide to the S-EHR a single way to retrieve all the health data produced during an Encounter in a single bundle without pagination. Moreover, using an operation instead of a complex query moves the responsibility from the client to the server: it is a responsibility of the R2D Access server to search, select and group all the data related to the instance of the requested Encounter that are considered relevant for the context of the encounter itself.

Full information about the operation Encounter/$everything can be found at this URL: https://www.hl7.org/fhir/encounter-operation-everything.html.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/Encounter/<EncounterId>/$everything
```

Although the FHIR specifications define three optional search parameters (_since, _type, _count) to be used with this operation, R2D Access specifications define this operation without any parameters, avoiding in this way the retrieval of partial or not coherent results.

Parameters under the responsibility of the S-EHR are only the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing heterogeneous instances of FHIR Resource (at least the Encounter resource). All the resources contained in the bundle are represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.15 *Get everything of a Composition*

This operation is used to return all the information related to an instance of a Composition, packing them in a single Bundle with all the features defined by the FHIR specifications to represent CDA documents. The response of this operation is a bundle of type "searchset" that contains a bundle of type “document”. This second bundle embeds the instance of Composition itself and all the resources referred by it. Full information about the operation Composition/$document can be found at this URL: https://www.hl7.org/fhir/composition-operation-document.html.

This operation can be used to retrieve the patient summary of a patient when it is represented with a Composition.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the Composition resource:

```
GET [base]/Composition/{CompositionId}/$document
```

Although the FHIR specifications define three optional search parameters (id, persist, graph) to be used with this operation, R2D Access specifications define this operation without any parameters.

Parameters under the responsibility of the S-EHR are only the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing a bundle of type `document`. All the resources contained in the bundle are represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.6.16 *Get everything of a DiagnosticReport*

This operation is used to return all the information related to an instance of a DiagnosticReport packing them in a single Bundle. This is the only operation defined by the R2D Access protocol that IS NOT part of the FHIR API specifications [FHIR API SPEC].

In a clinical context, diagnostic reports can be used by HCPs in two ways: i) simply reading the final report compiled by the expert who performed the diagnostic investigations, ii) requiring the health data that were produced during the diagnostic investigation to analyse them. The purpose of this operation is to return all the health data directly referenced by an instance of DiagnosticReport, but also additional data that are considered relevant to understand the context of the diagnostic investigation performed.

The basic idea behind this operation is to provide to the S-EHR a single way to retrieve all the health data produced during a DiagnosticReport in a single bundle without pagination. Moreover, using an operation instead of a complex query moves the responsibility from the client to the server: it is a responsibility of the R2D Access server to search, select and group all the data related to the instance of the requested DiagnosticReport that are considered relevant for the context of the diagnostic report itself.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DiagnosticReport resource:

```
GET [base]/DiagnosticReport/{DiagnosticReportId}/$everything
```

Although the FHIR specifications define three optional search parameters (id, persist, graph) to be used with this operation, R2D Access specifications define this operation without any parameters.

Parameters under the responsibility of the S-EHR are only the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Default value is `application/json`
Prefer	string	YES	Default value is `respond-async`

Return Value

- Instance of Bundle of type `searchset` containing a bundle of type `document`. All the resources contained in the bundle are represented with `Content-Type` corresponding to requested Accept parameter (default value is JSON).
- Version: FHIR R4

Return Codes

- **202 Accepted**: asynchronous request accepted.
- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **429 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

R2D Access servers implementing this transaction shall provide a CapabilityStatement Resource stating the transaction has been implemented.

4.3.7 Interface R2DAccessIdentification

This section provides an initial view of the interface called R2DAccessIdentification.

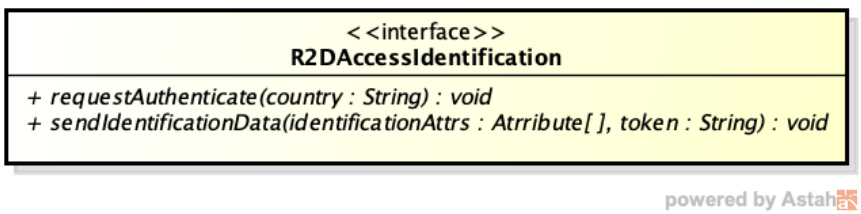

Figure 22 - R2DAccessIdentification interface

Usage of R2DAccessIdentification interface has already been shown in the sequence diagrams contained in sections Health data completeness and asynchronous interactions and Citizen identification. This interface defines two operations:

- requestAuthenticate: is the operation used by the S-EHR to start the eIDAS authentication process.
- sendIdentificationData: is the operation used by the S-EHR to send additional identification details of a citizen to grant access to a specific instance of R2D Access server.

Complete RESTful specifications of these methods are provided in the deliverable [D3.4].

4.3.8 Interface R2DAccessDICOM

The Digital Imaging and COmmunications in Medicine [DICOM] protocol is the standard for the communication and management of medical imaging information and related data, it is most commonly used for storing and transmitting medical images enabling the integration of medical imaging devices and Picture Archiving and Communication Systems (PACS) from multiple manufacturers. DICOMWeb [DICOMWEB] is the RESTful version of the DICOM protocol, this standard is composed by several sub-standard, the one used for retrieving DICOM studies is the Web Access to DICOM Objects by RESTful Services [WADO-RS] formerly referenced as WADO-RS.

Support for DICOM is NOT mandatory in R2D Access, it is fully acceptable that an implementation of R2D Access server does not provide support to DICOM and adopts the policy to embed significant key images represented with the Media resource type in a DiagnosticReport.

R2D Access implementers that choose to provide support for DICOM study are required to provide implementation also for the interface R2DAccessDICOM (the following picture (Figure 23) is taken from the architecture deliverable of the project [D2.6]).

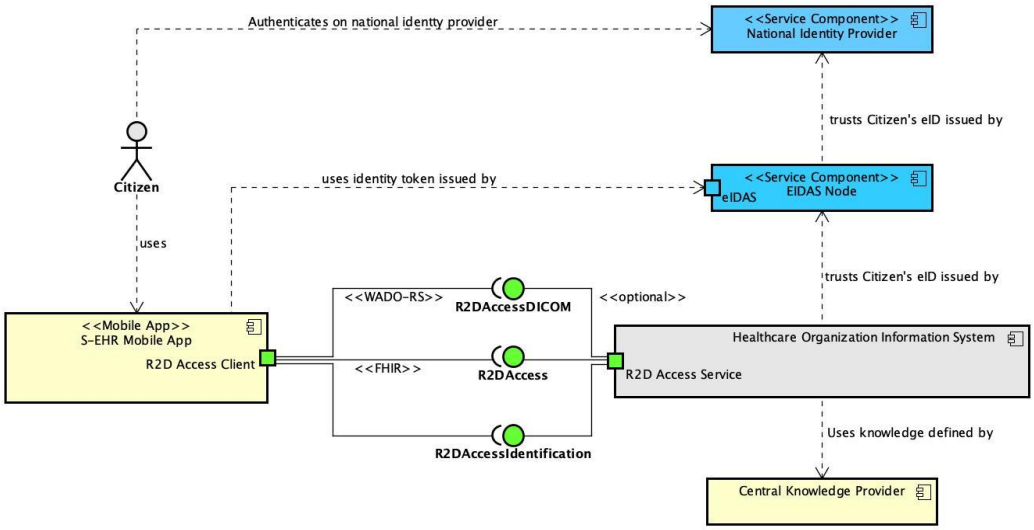

Figure 23 - DICOM support

The R2DAccessDICOM interface is a simplification of the WADO-RS interface, adopting only the basic methods to retrieve an entire DICOM study, a series of a study or an instance of a series. Moreover this implementation of WADO-RS server MUST adhere to the same security specifications defined by the R2D Access protocol, in particular those regarding the identification of the authenticated / requesting citizen by means of an eIDAS token stored as an header parameter with name `Authorization` in the HTTP

request. The token MUST be processed according to the process defined in section Citizen Identification and in the specific deliverable about security [D3.4]. The following diagram shows a logical view of the R2DAccessDICOM interface providing a first initial list of the methods it defines (Figure 24):

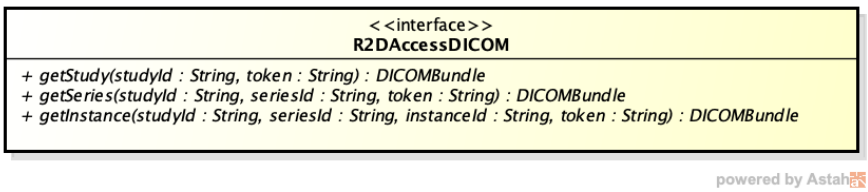

Figure 24 - R2DAccessDICOM interface

As for the R2DAccess interface, every method listed in the R2DAccessDICOM interface is described in a dedicated section using a template that fits better with RESTful services.
One important note for S-EHR developers: S-EHR are not required to compose the RESTful URLs for accessing DICOM objects, these URLS are already provided by the ImagingStudy itself using the following attributes:

- ImagingStudy.endpoint: WADO-RS endpoint for retrieving the entire study (see section Get a Study).
- ImagingStudy.series.endpoint: WADO-RS endpoint for retrieving an entire series of a study (see section Get a Series)

Complete specifications of WADO-RS protocol can be found at this page: http://dicom.nema.org/medical/dicom/current/output/html/part18.html.

4.3.8.1 *Get a Study*

This operation is used to retrieve an entire DICOM study in a single operation.

Endpoint and parameters

The following URL shows the basic structure of the query for requesting the study:

```
GET [base]/studies/{studyId}
```

Parameters under the responsibility of the S-EHR are only the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen

Accept	string	NO	Allowed media types for WADO-RS, default value is `application/dicom`

Return Value

- A payload containing one or more representations of the Target Resource with the `Content-Type` corresponding to the format set in the Accept parameter (default value is `application/dicom`). The payload may be single part or multipart depending on the media type.

Return Codes

- **200 Successful**: request successfully processed.
- **206 Partial Content**: The response payload contains representations for some, but not all, of the Target Resource(s).
- **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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **410 Gone**: the requested resource has been deleted.
- **413 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error.

4.3.8.2 *Get a Series*

This operation is used to retrieve an entire series of a DICOM study in a single operation.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the Composition resource:

```
GET [base]/studies/{studyId}/series/{seriesId}
```

Parameters under the responsibility of the S-EHR are only the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**

Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Allowed media types for WADO-RS, default value is `application/dicom`

Return Value

- A payload containing one or more representations of the Target Resource with the `Content-Type` corresponding to the format set in the Accept parameter (default value is `application/dicom`). The payload may be single part or multipart depending on the media type.

Return Codes

- **200 Successful**: request successfully processed.
- **206 Partial Content**: The response payload contains representations for some, but not all, of the Target Resource(s).
- **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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **410 Gone**: the requested resource has been deleted.
- **413 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error.

4.3.8.3 *Get an Instance*

This operation is used to retrieve a specific instance of a series of a DICOM study in a single operation.

This operation can be used to retrieve the patient summary of a patient when it is represented with a Composition.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the Composition resource:

```
GET [base]/studies/{studyId}/series/{seriesId}/instance/{instanceId}
```

Parameters under the responsibility of the S-EHR are only the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	eIDAS token of the authenticated citizen
Accept	string	NO	Allowed media types for WADO-RS, default value is `application/dicom`

Return Value

- A payload containing one or more representations of the Target Resource with the `Content-Type` corresponding to the format set in the Accept parameter (default value is `application/dicom`). The payload may be single part or multipart depending on the media type.

Return Codes

- **200 Successful**: request successfully processed.
- **206 Partial Content**: The response payload contains representations for some, but not all, of the Target Resource(s).
- **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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **410 Gone**: the requested resource has been deleted.
- **413 Too Many Requests**: client has submitted too many requests in the unit of time.
- **5XX Internal Server Error**: the server encountered an unexpected internal error.

4.3.9 Support for transactions

FHIR Transactions allow clients to submit FHIR Bundles containing several requests that are managed by the server in a transactional way. Due to the asynchronous and Read Only nature of the R2D Access protocol, support for FHIR transactions MUST NOT be provided by an R2D Access server.

Full details about FHIR transactions can be found at the following page: https://www.hl7.org/fhir/http.html#transaction.

4.3.10 Capability Statement

HL7 FHIR defines how a service can declare a CapabilityStatement resource describing the resources, transport, formats, and operations that can be performed on a series of resources for the service instance.

Full information about the CapabilityStatement can be found at the following URL: https://www.hl7.org/fhir/capabilitystatement.html.

The CapabilityStatement of a running FHIR server shall be retrievable at the following URL:

```
[base]/metadata
```

Additional information regarding the operations that must be listed in the capability statement of an R2D Access server are provided in the next sections.

5 R2D BACKUP PROTOCOL: BACKUP HEALTH DATA ON THE S-EHR CLOUD

The R2D Backup protocol defines the set of operations used for enabling the backup and the restore from a remote cloud of the health data repository stored locally on the mobile phone. It is most likely that the local health data repository is the result of the download (using R2D Access and / or D2D) of health data from several healthcare organizations therefore it is very important to give to the citizen to option to backup this *unique* repository on a remote cloud, and restore it (from a unique source) in case the mobile phone gets damaged, lost, broken or stolen.

The next section of this document, provides a detailed description on the technologies that have been studied and implemented towards the realization of the R2D Backup protocol, the context in which the R2D Backup protocol will be used, and of all the conditions that have influenced the decisions taken in the process carried out to specify the R2D Backup protocol.

5.1 R2D Backup Protocol scope

The purpose of the R2D Backup protocol is to allow citizens to securely back up their EHR in a remote repository (i.e. a S-EHR Cloud provider of their choice). The R2D Backup protocol defines a set of operations used for enabling (in a standardized way) the upload of encrypted health data on a S-EHR Cloud service along with the download of this encrypted health data to a S-EHR Mobile App from a S-EHR Cloud.

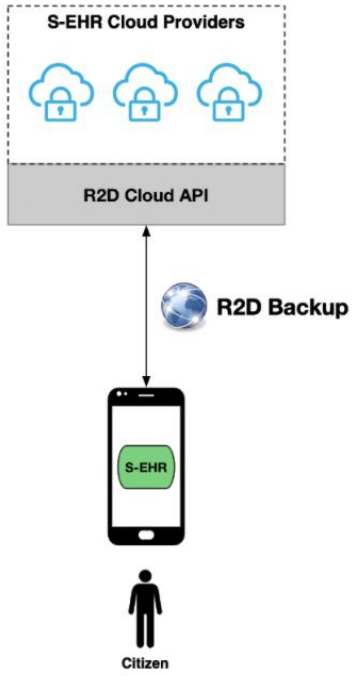

Figure 25 - R2D Backup protocol scope

5.2 Related Work

The communication between user applications and cloud providers is rapidly becoming a common occurrence since it facilitates the storage of vital personal data and its fast and constant retrieval. This section contains a list of technologies that implement similar operations, in the context of data exchange between cloud providers and other user services. In particular, it will cover the technical issues that are used by different providers in order to achieve this result.

To begin with, [Dropbox] is a typical example of this communication. It offers the possibility to make an app interact with the cloud and exchange data. This is achieved by *Dropbox API v2* [Dropbox API v2] which consists of a set of HTTP endpoints that are used to establish the communication between the app and the cloud. The Dropbox API v2 gives the developers the opportunity to manage files stored inside Dropbox through the app. Some functionalities that are included in this set of HTTP endpoints are full-text search, thumbnails and sharing. Regarding the format used by the endpoints, HTTP POST requests are called with parameters and responses applying the JSON format. Depending on the endpoint type there can be some alterations. For instance, *RPC endpoints (remote procedure call)* take JSON formatted arguments in the request body and return a JSON response in the response body. *Content-upload endpoints* are accepting files in the request body. This time, the arguments are sent as JSON in the *Dropbox-API-Arg* request header or *arg* URL parameter. Similarly, *Content-download endpoints* sent arguments as JSON in the *Dropbox-API-Arg* request header or *arg* URL parameter. The response body is composed of files, so the result will be formatted as JSON in the *Dropbox-API-Result* response header. Dropbox applies *OAuth 2.0* for authorizing API requests. Authorization is accomplished with the use of Authorization request header or authorization URL parameter. It is worth mentioning that Dropbox should not be used as an identity provider since OAuth 2.0 is an authorization protocol.
Furthermore, Dropbox provides some components that can be embedded inside an app and exploit all of its functionalities. To start with, *Chooser* [CHOOSER] is a small component which enables the fast file retrieval from the drive into the application by overcoming issues related to authentication, file browser implementation complications and uploads and storage control. It takes a single *options* object as a parameter and after a success callback function it returns a *files* parameter that consists of an array of file objects containing information about the requested files. Likewise, with the Saver *component* [SAVER] users can add files to their Dropbox drive and access them from different devices. The method which implements the saver takes as parameter the href URL of the file. The URI can be HTTP or HTTPS. Also, the name of the file is passed to the method along with a class that identifies the button which implements the method. Finally, the *Embedder component* [EMBEDDER] allows the preview of Dropbox files or folders and the interaction with them inside the app.

In a similar way, **Google Drive** also provides REST APIs in order to accomplish the communication with apps. With *Google Drive API* [GOOGLE DRIVE API] applications acquire additional functionality. Some of these capabilities are the simple or complex search queries for files and folders stored in the Google Drive, download and upload of files, share data with trusted users, etc. From a technical scope, the communication is taking place between three main components.

1. **Google Drive** is the main cloud file storage service which supplies users with a unique storage space. Each user has a separate storage slot called *My Drive*.
2. **Google Drive App** is the application that uses Google Drive to exploit its storage benefits.
3. **Google Drive API** guarantees that the app can use Google Drive's space for the sake of file storage and processing.

Another API provided by Google Drive is ***Activity API*** [ACTIVITY API]. Its main components are the *DriveActivity Resource* and the *query method*. The latter enables the user to collect information about changes made inside the drive. On the other hand, DriveActivity Resource keeps track of those changes.

The query method is exploited through a HTTP request in order to ask for a user's activity data. The request can be made either by the itemName, if the search is about a certain file, or by ancestorName if the desired result contains every item underneath a folder. Furthermore, the response is a list of DriveActivity resources that fulfil the terms of the given parameters.
Just like Dropbox, Google Drive API leverages OAuth 2.0 protocol in order to authenticate app users. For example, an application using *Google sign-in* controls the OAuth 2.0 flow and application access tokens.

In the same direction, **DataVaults** [DATAVAULTS] aims to give users the ability to fully manipulate their data and share it with other organisations or individuals by adopting adjustable sharing schemes. At the same time, it plans to overcome concerns related to confidentiality, ethics and intellectual property rights. Generally speaking, DataVaults recommends an innovative framework and architecture which collects personal information derived from several sources and provides them in a secure and interoperable manner to the users. DataVaults functionality can be considered as a large ever-expanding ecosystem where many groups of people can benefit from the constant production and circulation of data.

Lastly, **International Data Spaces** [IDS] vision is to create a secure network where data providers will take into account the value of their data and interact with other entities in an independent and controlled way. The International Data Spaces Association [IDSA] is in charge of the project and responsible for providing users with certification over their data. The main component that is responsible for the IDS functionality is the *IDS Connector* [IDS-CONNECTOR]. It provides multiple operations over the data and guarantees that every procedure is happening in a secure way.

5.3 R2D Backup Overview

The R2D Backup protocol defines the operations offered to S-EHR applications to safely back up health data of the citizen. Similarly to the R2D Access protocol and the R2D Emergency protocol, the R2D Backup protocol is an internet based protocol, and its operations are exposed as RESTful services. The following figure (Figure 26), extracted from the architecture deliverable [D2.6], shows the role of the R2DBackup interface and the two applications that uses it:

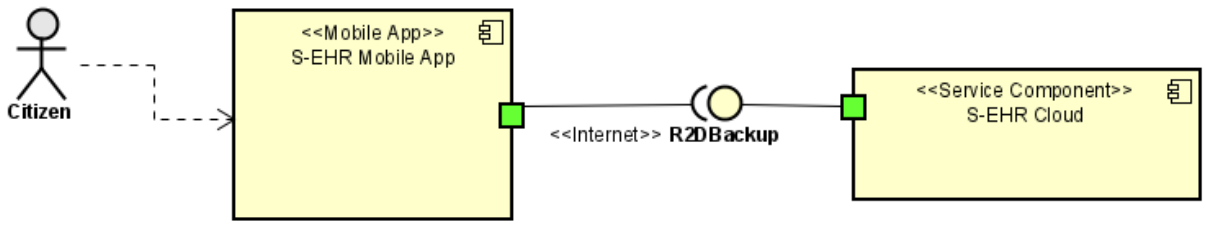

Figure 26 - R2D Backup protocol interface

The R2D Backup protocol defines the basic operations for the exchange of encrypted health data between the S-EHR mobile application and the S-EHR Cloud service. The following figure (Figure 27) shows a logical view of the R2DBackup interface, providing a first initial list of the methods it defines:

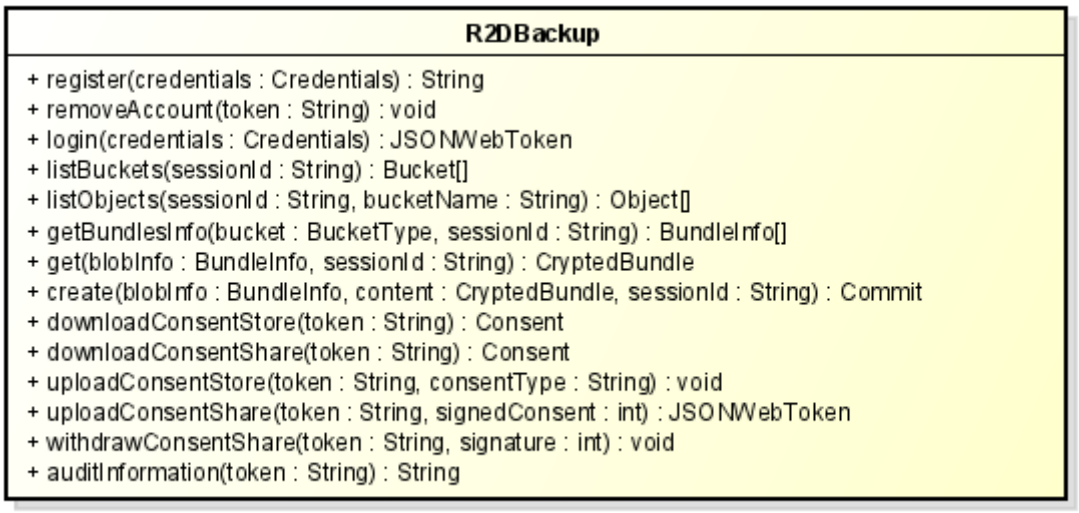

Figure 27 - R2D Backup protocol interface operations

Every operation defined by the R2D Backup protocol is described in a specific section of this chapter using an ad hoc template for RESTful services, the allows to describe the following features:

- the HTTP method to be used;
- the URL to be invoked;
- the type of the FHIR resource that is the subject of the request;
- the FHIR profile of the returned data;
- the allowed parameters that may be added to the URL and the constraints defined for the allowed values of each parameter;
- the set of allowed header parameters and the constraints defined for the allowed values of each parameter;
- the set of possible HTTP return codes.

Complete technical specifications of these methods are provided in the next sections of this chapter, but before going into technical details it is important to define the technical context in which the R2D Backup protocol operates and all the preliminary operations that must be executed by S-EHR to grant access to R2D Backup services. These operations, although not related to the exchange of health data, still remain crucial to the overall communication, thus they are defined as part of the R2D Backup protocol.

The remainder of this section is focused on the behavioural aspects of the R2D Backup providing sequence diagrams showing how the R2DBackup interface has to be used to upload and download data from a S-EHR Cloud. Moreover, it includes the messages that will be exchanged when invoking the R2D Backup operations, and the rules in which the exchanged messages should be compliant with.

5.3.1 Registration and upload of health data

The next figure (Figure 28) shows the sequence diagram for registering, and then uploading data to the S-EHR Cloud. In more detail, it describes the interactions between the S-EHR Mobile application and the S-EHR Cloud, over the R2D Backup protocol, with the usage of the MR2DBackup, a mobile library which implements the R2DBackup interface. A detailed description of the diagram is included afterwards.

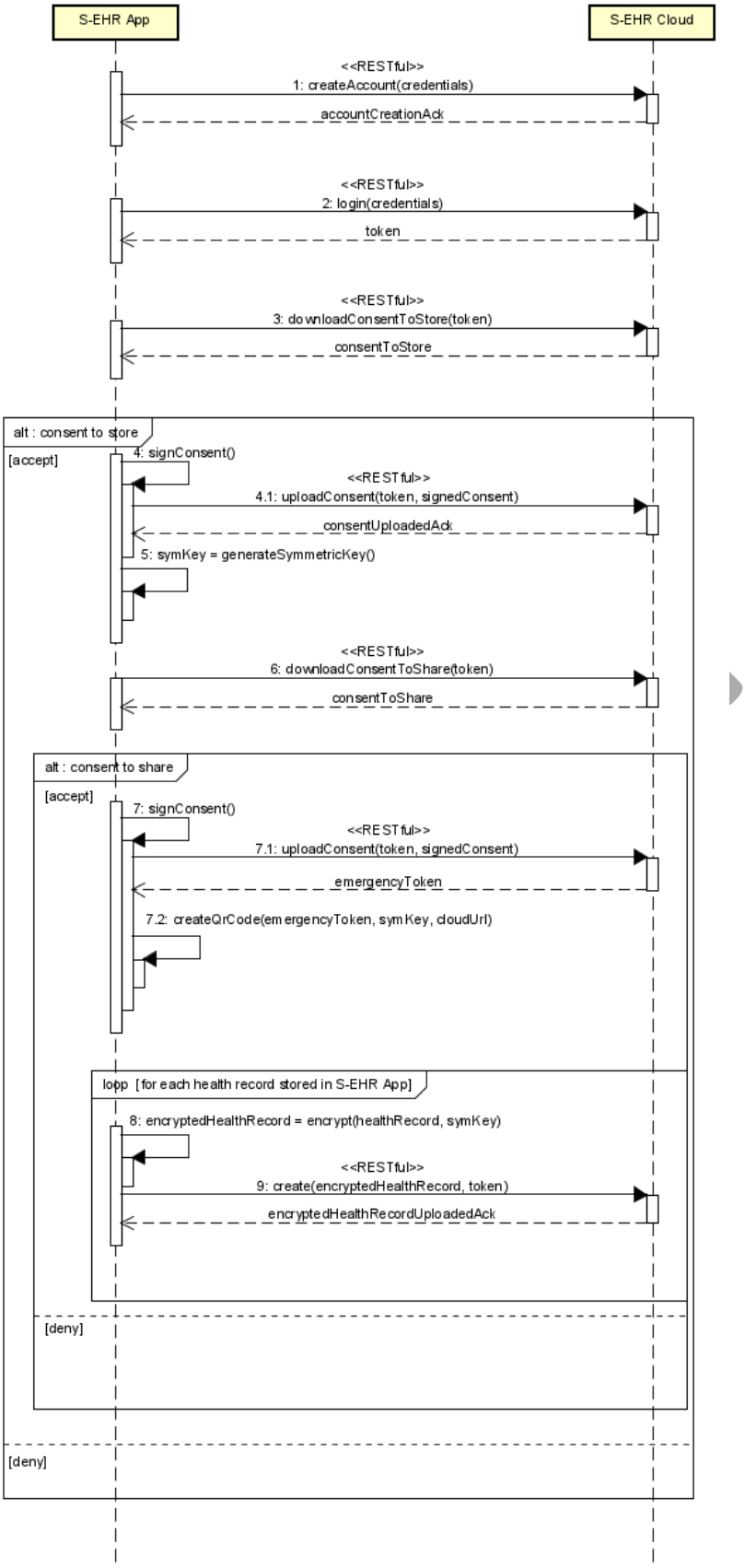

Figure 28 - R2D Backup protocol interactions with respect to the upload of encrypted health data to the S-EHR Cloud

Following, a detailed description of the sequence diagram takes place. It should be noted that the steps that are depicted in the above figure can be split into three major categories: (i) citizen's account creation and login, (ii) consent acceptance, and (iii) health data encryption and upload on the S-EHR Cloud.

(i) Citizen's account creation and login

- Step 1 - Create Account: This step is performed in order for the citizen to connect to his/her preferred S-EHR Cloud service and create an account to it. The citizen should include in the request their credentials in order to create the account on the S-EHR Cloud. The same credentials will be used also for the login of the citizen.
- Step 2 - Login to S-EHR Cloud: This step is performed in order for the citizen to login to their preferred S-EHR Cloud provider. The citizen should include his/her credentials on the request. If the credentials are correct, the S-EHR Cloud provider returns an authentication token to the S-EHR Mobile application that can be used to perform any health data exchange to and from the S-EHR Cloud.

(ii) Citizen's consents acceptance

- Step 3 - Accept consent to store encrypted health data on the S-EHR Cloud: The citizen in order to be able to upload encrypted health data on the S-EHR Cloud should at first accept a consent that is sent from the S-EHR Cloud. This consent is retrieved by the S-EHR mobile application, and upon citizen's acceptance is digitally signed. Afterwards, the consent including the citizen's digital signature is sent back to the S-EHR Cloud. In the case where the citizen declines this consent, he/she is not able to perform any requests to the S-EHR Cloud.
- Step 4 - Accept consent to share encrypted health data stored on the S-EHR Cloud with authorized HCPs and trusted Healthcare Institutions: This step regards the second consent that the citizen may accept, and concerns the ability of HCPs to access the citizen's health data that is stored in the S-EHR Cloud, if an emergency occurs. In contrast with the first consent, this is not a mandatory consent. If the citizen decides to decline this consent, it means that he/she has chosen to utilize the S-EHR Cloud only as a backup service. What is important to note here is, that even if the citizen initially declines this consent, he/she also has the ability to accept it at a later time.

(iii) Health data encryption and upload to the S-EHR Cloud

What is important to note here, before the steps description is, that the steps that will be described below can only take place if the citizen accepts the first consent (i.e. to allow the S-EHR Cloud to store encrypted health data).

- Step 5 - Encrypt and upload health data on the S-EHR Cloud: The health data that is stored locally on the S-EHR Mobile application is at first encrypted with a symmetric key that is generated on the S-EHR mobile application and then uploaded on the S-EHR Cloud. The upload of the health data to the S-EHR Cloud is done automatically by the S-EHR App if a change on a health record is applied. This aforementioned key is also used by the S-EHR Mobile application in order to decrypt at a later time the health data as it is downloaded from the S-EHR Cloud.

5.3.2 Download of health data

The next figure (Figure 29) shows the sequence diagram of the R2D Backup protocol as it is used for the download of the encrypted health data that is stored on the S-EHR Cloud to a S-EHR Mobile application. Similar to the previous sequence diagram, the steps can be split into two main categories: (i) the login of the citizen, and (ii) the download of the encrypted health from the S-EHR Cloud to the S-EHR Mobile application.

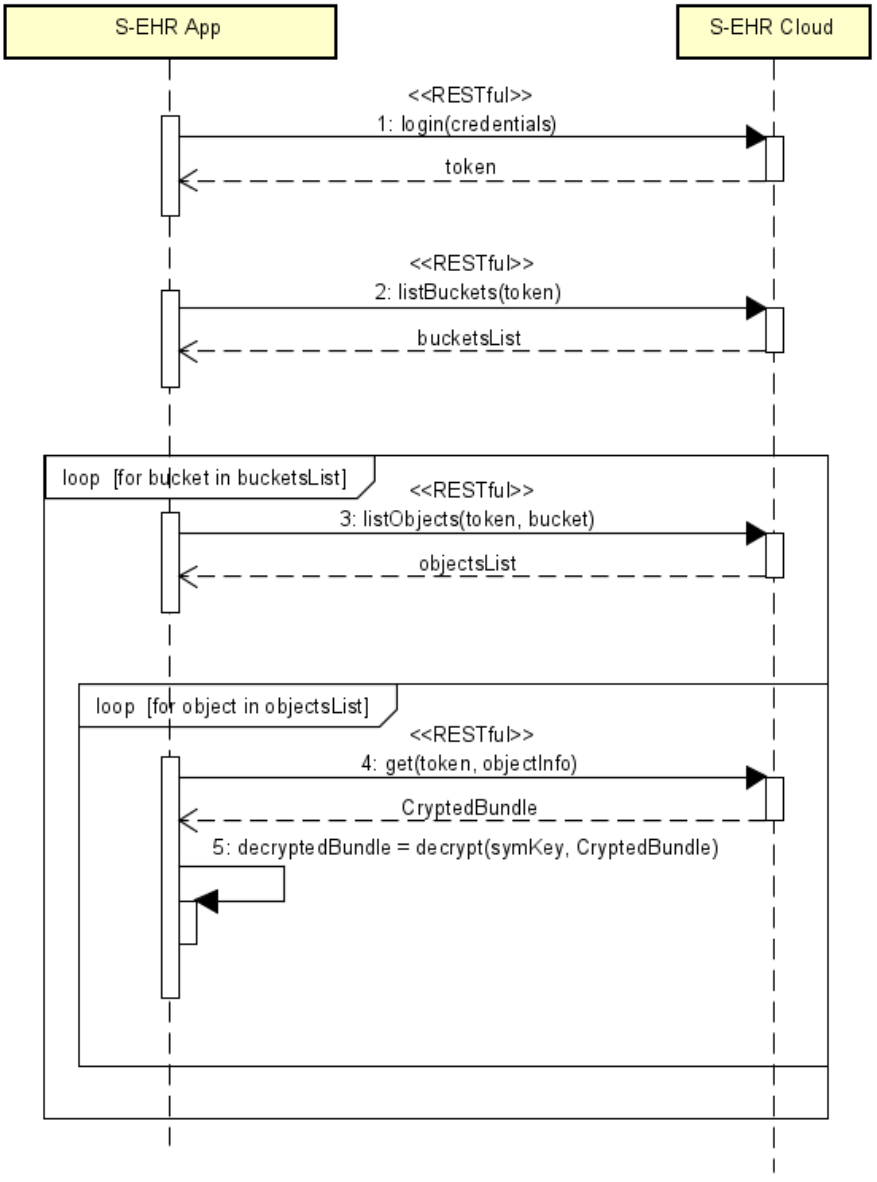

Figure 29 - R2D Backup protocol interactions with respect to the download of encrypted health data from the S-EHR Cloud

(i) Citizen's login

- Step 1 - Citizen's login: This step is performed in order for the citizen to login to their preferred S-EHR Cloud provider from a new S-EHR Mobile application. The citizen should include his/her credentials on the request, and if the credentials are correct, the S-EHR Cloud provider returns an authentication token to the S-EHR Mobile application that can be used to perform any health data

exchange to and from the S-EHR Cloud, which in this case regards the request to download encrypted health data from the S-EHR Cloud.

(ii) Download and decryption of health data from the S-EHR Cloud

- Step 2 - List buckets: The encrypted health data of a citizen is stored in the S-EHR Cloud in the form of objects within several buckets. For this reason, before the actual download of the health data on the mobile of the citizen a request to receive a list of all the buckets that are used to store the citizen's health data is performed.
- Step 3 - List objects: This step is performed in order to receive a list of the encrypted health data that is stored on a specific bucket on the S-EHR Cloud.
- Step 4 - Download and decrypt health data object: As soon as the list of the objects (i.e. the encrypted health data) is returned from the S-EHR Cloud, the request to download it is performed. The encrypted health data Bundle is downloaded from the S-EHR Cloud and then decrypted locally on the S-EHR Mobile app with the same encryption key that was used for the encryption in the first place. Afterwards, the health data is stored locally on the citizen's mobile device.

5.4 R2D Backup Specifications

This section contains the technical specifications of every operation of the R2D Backup protocol introduced in the previous sections. The R2D Backup is a protocol based on the HTTP specifications version 1.1, providing details about these standards is out of the scope of this document, complete knowledge of HTTP specifications [rfc 2616] [rfc 7540] is required for a complete understanding of the present deliverable.

5.4.1 Register to a S-EHR Cloud

This operation allows a citizen to send a request to create an account to a S-EHR Cloud provider. As soon as the account is created, a bucket linked to this account is created as well. This bucket is used to store the encrypted health data of the citizen.

Endpoint and parameters

The following URL shows the structure of the request for registering to a S-EHR Cloud:

```
POST [base]/citizen/register
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Basic Base64(username:password)

Return Value

- void operation.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.2 Login to the S-EHR Cloud

This operation allows a citizen to send a request to login to the S-EHR Cloud. If the login is successful, an authorization token is returned to the S-EHR Mobile application. This token can be used by the citizen owning the account in order to perform requests to the S-EHR Cloud.

Endpoint and parameters

The following URL shows the structure of the request for authenticating to a S-EHR Cloud:

```
POST [base]/citizen/login
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Basic Base64(username:password)

Return Value

- JSON Web Token, used for the authorization of the citizen in the body of the response.
- JSON Web Token, used by HCPs during emergency situations in order to perform a request to access the citizen's health data stored in the S-EHR Cloud.
  The JSON response is structured as follows:

```
{
      "token" : JSON Web Token: Citizen's Authorization
            token,
      "emergencyToken" : JSON Web Token: Healthcare
            Institution emergency token
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.3 List of available buckets

This operation allows a citizen to send a request to retrieve the list of the buckets that this citizen can gain access to. Different buckets may be created in emergency situations. The citizen may access all concerning buckets.

Endpoint and parameters

The following URL shows the structure of the request for retrieving the list of buckets:

```
GET [base]/citizen/buckets
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- List of buckets that the citizen's account has access to, in JSON format. The JSON response is structured as follows:

```
{
      "buckets" : [
            String: Bucket Name 1,
            String: Bucket Name 2,
            ...
      ]
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.4 List of encrypted health records stored in a bucket

This operation allows a citizen to send a request to retrieve the list of the objects (i.e. the encrypted health data) stored in a specific bucket.

Endpoint and parameters

The following URL shows the basic structure of the query for retrieving the list of objects in a bucket:

```
GET [base]/citizen/buckets/{$bucketName}
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- List of items contained in a bucket in JSON format. The JSON response is structured as follows:

```
{
      "bucket" : String: Bucket Name,
      "objects" : [
            String: Object Name 1,
            String: Object Name 2,
            ...
      ]
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.5 Retrieval of metadata information of an object

This operation allows a citizen to send a request to download the metadata of an encrypted health record from the S-EHR Cloud..

Endpoint and parameters

The following URL shows the basic structure of the query for requesting the metadata:

```
GET [base]/citizen/{$bucketName}/{$objectName}/metadata
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- The metadata of the health data resource requested in JSON format. The JSON response is structured as follows:

```
{
      "object" : String: Object Name,
      "metadata": {
            "objectName": String: Object Name,
            "bucketName": String: Bucket Name,
            "size": Float: Size of the object in KB,
            "type": String: Object type,
            "dateAdded": Date: Date stored in S-EHR Cloud
      }
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.6 Download of an object from the S-EHR Cloud

This operation allows a citizen to send a request to download an encrypted health data resource from the citizen's bucket and decrypt it locally on the S-EHR Mobile app.

Endpoint and parameters

The following URL shows the basic structure of the query for executing searches on the DocumentReference resource:

```
GET [base]/citizen/{$bucketName}/{$objectName}
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- Encrypted Binary file.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.7 Upload of a health record to the S-EHR Cloud

This operation allows a citizen to send a request to upload an encrypted FHIR Bundle containing health data of the authenticated citizen.

Endpoint and parameters

The following URL shows the basic structure of the query for uploading an encrypted bundle:

```
POST [base]/citizen/upload?objectName={$ResourceCategory}
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**

Authorization	string	YES	Authentication token - JSON Web token

Body value

- The encrypted bundle is stored in the body of the request.

Return Value

- void operation.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.8 Download of consent for S-EHR Cloud provider

This operation allows a citizen to send a request to download the consent that allows the S-EHR Cloud to store the citizen's health data.

Endpoint and parameters

The following URL shows the basic structure of the query for downloading the consent:

```
GET [base]/citizen/consent/cloud
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- The consent that after signing it and re-uploading it on the S-EHR Cloud allows the service to store the citizen's encrypted health data for backup purposes.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.9 Upload of signed consent for S-EHR Cloud provider

This operation allows a citizen to sign and upload the consent that allows the S-EHR Cloud to store the citizen's health data.

Endpoint and parameters

The following URL shows the basic structure of the query for uploading the consent:

```
POST [base]/citizen/consent/cloud
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Body value

- The signed consent.

Return Value

- void operation.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.10 Download of consent for trusted HCO

This operation allows a citizen to send a request to download the consent that allows the S-EHR Cloud to share the citizen's health data with HCPs from authorized Healthcare Organizations.

Endpoint and parameters

The following URL shows the basic structure of the query for downloading the consent:

```
GET [base]/citizen/consent/hco
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- The consent that after signing it and re-uploading it on the S-EHR Cloud allows the service to share the citizen's health data with HCPs from authorized Healthcare Organizations.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.

- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.11 Upload of signed consent for trusted HCO

This operation allows a citizen to sign and upload the consent that allows the S-EHR Cloud to share the citizen's health data with HCPs from authorized Healthcare Organizations.

Endpoint and parameters

The following URL shows the basic structure of the query for uploading the consent:

```
POST [base]/citizen/consent/hco
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Body value

- The signed consent.

Return Value

- JSON Web Token, that can be used by HCPs from trusted Healthcare Organizations in order to access the citizen's health data stored in the S-EHR Cloud.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.12 Download of auditing information

This operation allows a citizen to download audit information regarding the Healthcare Organizations that were granted access to the S-EHR Cloud, and any changes made to the citizen's health records.

Endpoint and parameters

The following URL shows the basic structure of the query for downloading the auditing information:

```
GET [base]/citizen/auditing
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- The auditing information in JSON format. The JSON response is structured as follows:

```
{
    "hcosGrantedAccess:" [
        "hco": Title of HCO": {
            "acceptedRequestOn": TimeStamp,
            "healthRecordsAccessed": [
                {"$healthRecordName": TimeStamp, ...}
            ],
            "healthRecordsGenerated": [
                {"$healthRecordName": TimeStamp, ...}
            ]
        },
        ...
    ],
    "healthRecords": [
        "$healthRecord: Health Record Name": {
            "uploadDate": TimeStamp,
            "updatedOn": [
                timeStamp1, ... : TimeStamp
            ],
            "downloadedOn": [
                timeStamp1, ... : TimeStamp
            ]
        },
        ...
    ]
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.13 Withdrawal of signed consent for trusted HCO

This operation revokes the previously signed consent, thus the S-EHR Cloud is no longer allowed to share the citizen's health data with HCPs from trusted Healthcare Organizations.

Endpoint and parameters

The following URL shows the basic structure of the query for withdrawing the consent:

```
DELETE [base]/citizen/consent/hco
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Body value

- The signed withdrawal request.

Return Value

- void operation.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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.

- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

5.4.14 Removal S-EHR Cloud account

This operation deletes the citizen's account from the S-EHR Cloud and removes all the data that is stored in the S-EHR Cloud related to this citizen.

Endpoint and parameters

The following URL shows the basic structure of the query for withdrawing the consent:

```
DELETE [base]/citizen/account
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Authentication token - JSON Web token

Return Value

- void operation.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

6 R2D EMERGENCY PROTOCOL: EMERGENCY ACCESS OF HEALTH DATA FROM THE S-EHR CLOUD

The R2D Emergency protocol defines the set of operations that allow authorized HCPs to access the encrypted health data that is backed up on a S-EHR Cloud of a citizen in need during an emergency situation over the internet.

The next section of this document, provides a detailed description on the technologies that have been studied and implemented towards the realization of the R2D Emergency protocol, the context in which the R2D Emergency protocol will be used, and of all the conditions that have influenced the decisions taken in the process carried out to specify the R2D Emergency protocol.

6.1 R2D Emergency Protocol scope

The purpose of the R2D Emergency protocol is to allow authorized HCPs to gain access to the health data the citizen backed up on the S-EHR Cloud during an emergency situation. The R2D Emergency protocol defines a set of operations used for enabling (in a standardized way) the download of encrypted health data by an authorized HCP from a S-EHR Cloud service to an HCP App, along with the ability upload to the S-EHR Cloud of that specific citizen in need, new health data regarding that emergency, and reports at patient discharge, once the emergency is over.

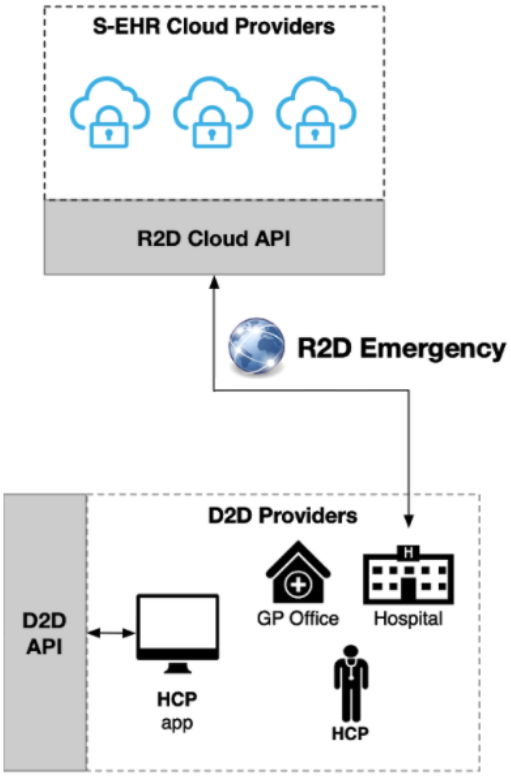

Figure 30 - R2D Emergency protocol scope

6.2 Related Work

Many cloud providers offer the opportunity to handle emergencies and other situations where the data access from third parties has a vital significance. It is interesting to see the ways its provider handles such challenges regarding the variety of the technologies used to grant and revoke access from involved entities.

The most popular Cloud providers don't offer functionalities that aim to cope with emergency scenarios but they have developed technologies which handle occasions when data needs to be shared with one or more stakeholders.

In the first place, **Dropbox** uses Dropbox API in order to manage file permissions and share data with stakeholders [DBX SHARING GUIDE]. They provide three different ways that data can be distributed depending on the scenario and restrictions. The first option is a **shared link** that gives admission to non-Dropbox users. By default, these links are accessible publicly but there are certain restrictions that can be added such as password or expiration time. The operation which is used to create the shared links takes as input a "path" and a "settings" parameter. Path is responsible for identifying the requested file while settings manages the actions and the availability of the file or folder. If, for example, the link points to an online editable file (e.g. Google docs) it can give modification abilities to the third party. This is achieved by setting the "editor" value to the "access" parameter. The second option is to add users to a **shared folder** which contains all the needed files. Generally speaking, shared folders have an owner and multiple members that can be either viewers or editors. The last alternative is **shared files** and can be exploited for giving access to a specific content. Every file can be shared with a single or various accounts and also with groups of people. Responsible for giving access to third parties are file owners and editors.

In a similar manner, **Google Drive** uses permissions to declare accessibility to a variety of entities (user, group, domain, etc.). The concept behind the permission is that every file or folder inherits the rules of its parent. A **Files Resource** is a list of boolean *capabilities* fields that is used to determine the available operations for every file. This goes hand in hand with the *user's Permission Resource* concerning a data source. When creating a permission, there are two parameters that need to be declared. The first one is "*type"* which defines the targeting entities such as user, group, domain or anyone. The second parameter is the "role" and offers corresponding capabilities to the above mentioned entities.

International Data Spaces' [IDS] concept focuses on the unrestricted data exchange inside its secure environment. This initiative offers a software architecture which ensures freedom of data sharing and secure interaction between parties. From a technical scope, *Data Owners* are providing their data through a *Data Provider* device. This operation is happening with the help of the *IDS Connector*. It allows entities to attach usage policies into the data spaces environment. It also offers a trusted manner for applications to exploit the IDS functionalities. On the other hand, *Data Recipients* retrieve the data through *Data Consumer* devices. Users that are certified by the IDS Association establish constraints on their data before going public and accessible to other parties. However, when it comes to vital data such as Electronic Health Records, the data exchange procedure cannot be restricted to plain permission restrictions. Cloud services may offer a ubiquitous access to information but the exposure of such data in public storage raises a plethora of concerns. The above mentioned cloud drives are not specialized for emergency situations. As a result, many surveys have been conducted in order to propose ways to overcome these challenges.

One survey [OLIVEIRA 2020] suggests the ***Red Alert Protocol*** *(RAP)* to address situations when the patient is not conscious. First of all, users are registered through a central Registration Authority and are assigned with *user attributes* and a unique identifier that will be used when needed. RAP is divided into four basic stages. In this phase, the patient stores his EHR which is encrypted as a ciphertext in the cloud. Next, in the *Emergency Session*, the Master Authority (MA) links the patient to every treatment team that needs to be

involved. The procedure starts by the time someone informs the health center for the incident. Each treatment team needs to prove its involvement in the operation. When the attributes are validated, the MA generates a global emergency key and a control token for each team. In this way, the Master Authority can stop sharing data with a certain team when its contribution has ended without revoking access from everyone. The third stage is the *Process Data Phase.* One of the leaders of the team sends the access token to the cloud in order to receive the data. The HealthCare Practitioners then decrypt the EHR by using the emergency key and finally read the data through an application. Finally, the *Leave Session* phase is happening when the patient is no longer under care of a treatment team. The MA is informed and the access tokens are either revoked or expired. Consequently, no more entities can access the patient's information.

Another study [LI 2010], proposed a patient-centric *Personal Health Record system* which supports multiple owners over the stored data. The framework exploits the attribute-based encryption (ABE) to certify that every user has control of their data. Data users, meaning the entities who want to access the data for a variety of reasons, are classified according to their attributes (researcher, professional, etc.). Therefore, data owners enable the access only to specific groups that satisfy those requirements. As a second security policy, the whole user ecosystem is divided into two security domains. *Personal Domain (PSD)* is composed of users that are personally connected to the PHR owner who is in charge of the group. On the other hand, *Public Domain (PUD)* may contain entities from healthcare, education, government and insurance fields. This domain also involves *public attribute authorities (PAA)* that are responsible for providing credentials to authorized users. Except for the above-mentioned domains, there is a third subdomain called *Emergency Department (ED)* and its goal is to manage situations when the regular access polishes are not useful. Access rights to PHR's are also assigned to the appropriate EDs. The emergency teams need to contact the ED and validate their identity along with the emergency situation. After a successful validation, the team obtains temporary read-only keys which are revoked after the completion of the emergency.

One more survey [BANERJEE 2013] talks about a centralized cloud database that will store every user's medical history in a single standard-formatted and interoperable document. The storage system can be accessed by hospitals and health centers to receive information and health records of the desired patient. The main concept is to achieve the authentication of the patient with the use of biometric data such as fingerprints. Additionally, a unique state-owned identification number (social security number) can be used. Every time new data is generated, the cloud center is updated with the recent information. Taking into account all the available documents, the system creates a single summarised document that contains the complete medical history of that person. The creation of this document is implemented with the multi-document summarization technique. First, a summary of each single document is created by the use of a graph based ranking algorithm. From all these summaries a meta-document is produced and with the execution of the page ranking method on it the final result is generated. Finally, if the patient is unconscious, the health center personnel uses the biometric information to access and query the storage system to find a match and finally retrieves the appropriate data in order to use it for the operation.

The next survey [LOUNIS 2016] proposed an architecture consisting of four main entities. A wireless sensor network (WSN) that gathers health data from the patients, the monitoring applications which are used by the HCPs to access stored data, the Healthcare Authority that manages security issues and the cloud servers. Emergencies can be detected either by human intervention or by the advanced WSN. The scenarios

are divided into *proactive* and *passive*. In the first case, the system detects the emergency by analyzing collected health data and informs the appropriate entities. Additionally, it modifies access parameters in order to enable the staff to access the patient's health data. The healthcare authority is also alerted and is responsible for providing further access privileges given the emergency situation. On the other hand, if the emergency is detected by human intervention, the group of people that needs to act first is requesting limited access to the health data so that they can offer the basic medical help. Furthermore, some additional attributes are used such as Emergency Case (EC) which facilitates the identification of the emergency in order to address proper medical assistance. Also, the exclusion of unauthorised access to medical data is achieved with *Patient Identity (PI)* and *Emergency Keys*. These keys should be prebuilt and easily retrieved in order to make emergency response accelerated. Eventually, access to medical data should be temporary and emergency keys should be revoked after the end of the procedure.

6.3 R2D Emergency Overview

The R2D Emergency protocol defines the operations represented by the interfaces that are offered by the Healthcare Organization Information System (i.e. the HCP web application). These interfaces are included in the HCP application, and will be used by authorized HCPs who need to gain access to the S-EHR Cloud provider of a citizen in need, during an emergency, in order to download their health data, and upload new health data related to the emergency, such as the Patient Discharge Report. The HCPs will be the only actors involved in the overall interaction, for the exchange of encrypted healthcare related data with the S-EHR Cloud (Figure 31).

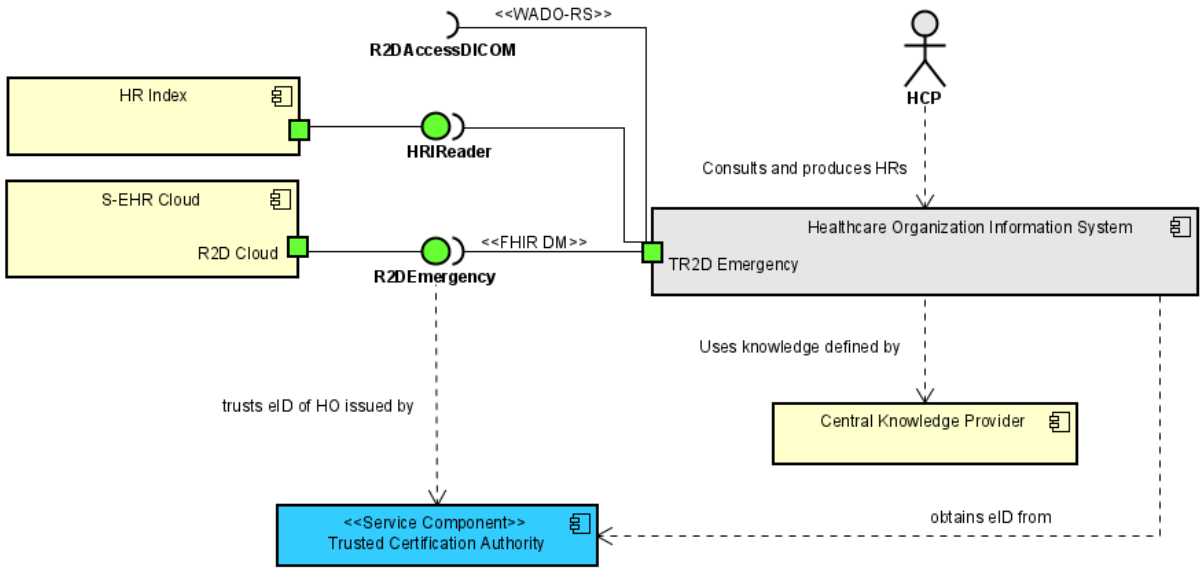

Figure 31 - R2D Emergency protocol interface

The R2D Emergency protocol defines the basic operations that the protocol should contain, for the ability of HCPs to access the citizen's health data that is stored in a S-EHR Cloud, as well as for exchange of

encrypted health data between an HCP application and a S-EHR Cloud service. The next class diagram shows a logical view of the R2DEmergency interface providing a first initial list of the methods it defines:

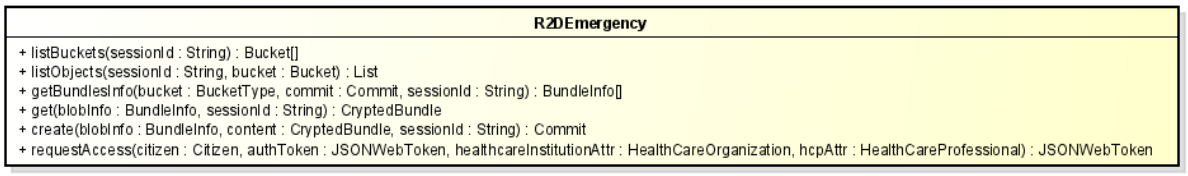

Figure 32 - R2D Emergency protocol interface operations

Every operation defined by the R2D Emergency protocol is described in a specific section of this chapter using an ad hoc template for RESTful services, the allows to describe the following features:

- the HTTP method to be used;
- the URL to be invoked;
- the type of the FHIR resource that is the subject of the request;
- the FHIR profile of the returned data;
- the allowed parameters that may be added to the URL and the constraints defined for the allowed values of each parameter;
- the set of allowed header parameters and the constraints defined for the allowed values of each parameter;
- the set of possible HTTP return codes.

Complete technical specifications of these methods are provided in the next sections of this chapter, but before going into technical details it is important to define the technical context in which the R2D Emergency operates. The remainder of this section is focused on the behavioural aspects of the R2D Emergency protocol providing sequence diagrams showing how the R2DEmergency interface has to be used to upload and download data from a S-EHR Cloud. Moreover, it includes the messages that will be exchanged when invoking the R2D Emergency operations, and the rules in which the exchanged messages should be compliant with.

6.3.1 R2D Emergency download of encrypted health data from a S-EHR Cloud

This section describes the use of R2D Emergency protocol as a medium protocol for the communication and data exchange of encrypted health data between an HCP application and a S-EHR Cloud. In more detail, it describes the interactions between the HCP application and the S-EHR Cloud, over the R2D Emergency protocol to download encrypted data from the S-EHR Cloud of a patient to the computer of the authorized HCP that is managing the emergency situation. The sequence diagram also shows the preliminary steps that need to take place in order for an HCP to gain access to a citizen's health data during emergency situations.

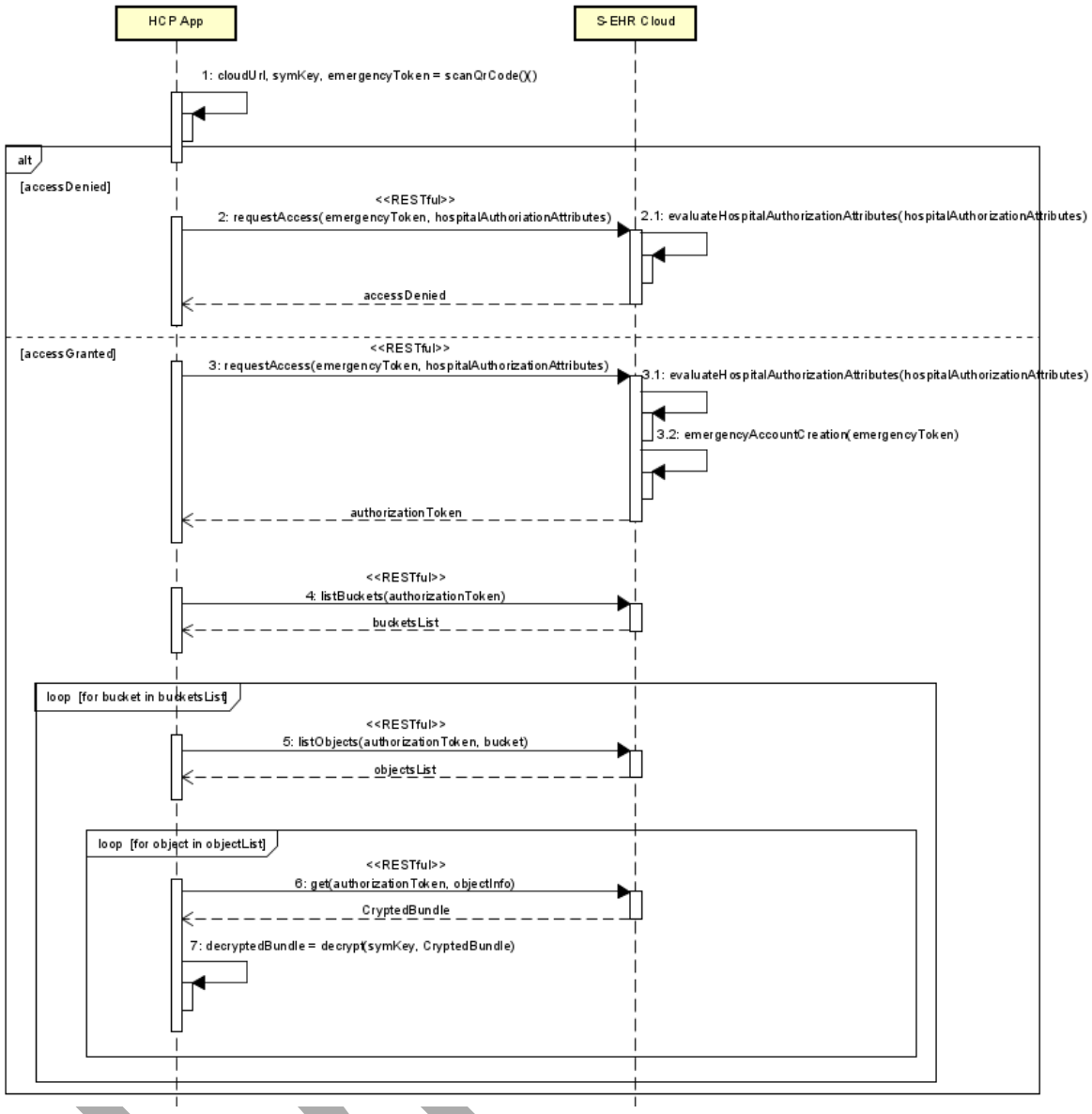

Figure 33 - R2D Emergency protocol interactions with respect to the download of encrypted health data from the S-EHR Cloud to the HCP app during an emergency

Following, a detailed description of the sequence diagram takes place. It should be noted that the steps that are depicted in the above figure can be split into three major categories: (i) retrieving information regarding the citizen's preferred S-EHR Cloud service, (ii) requesting to access the citizen's health data, and (iii) download and decryption of health data from the S-EHR Cloud.

(i) Retrieving information regarding the Citizen's S-EHR Cloud

- Step 1 - Scan QR-code and extract S-EHR Cloud information: The HCP uses the QR-code that is obtained from the citizen and scans it with the use of the HCP application. This QR-code contains information with respect to the citizen's preferred S-EHR Cloud service (i.e. which S-EHR Cloud service the citizen uses), an emergency access token which can be used by HCPs in order to request access to the citizen's health data, and the symmetric key which will be used for the decryption of

downloaded health data as well as for the encryption of health data that will be uploaded to the S-EHR Cloud after the emergency is ended.

(ii) Requesting to access the citizen's health data that is stored in the S-EHR Cloud

- Step 2 - Request to access the citizen's health data: The HCP performs a request to the citizen's selected S-EHR Cloud provider in order to access the citizen's health data. This request includes the citizen's emergency access token, along with the Healthcare Institution's attributes, and the HCP's personal identification information. The S-EHR Cloud, upon the receiving of the HCP's request, may either approve or decline it. As a first step the S-EHR Cloud may decline any incoming request that is not coming from a trusted healthcare institution (i.e. trusted by the S-EHR Cloud provider). If the incoming request is coming from a known healthcare institution, the S-EHR Cloud contacts a trusted authorization service hosted by the S-EHR Cloud, named HCP Attributes Evaluation service, in order to evaluate the attributes that were sent with the request in the first place. This trusted service may be an Attribute-Based Access Control (ABAC) Engine. If the evaluation is successful the S-EHR Cloud creates a temporary account that may be used by the HCP's of that specific Healthcare Institution and can be used to access the citizen's health data that is stored in the S-EHR Cloud, as well as uploading health data to the S-EHR Cloud once the emergency is over. If the evaluation is not successful, the request to access the S-EHR Cloud is rejected.

(iii) Download of encrypted health data from the S-EHR Cloud and decryption locally on the HCP application

- Step 3 - List buckets: The encrypted health data of a citizen is stored in the S-EHR Cloud in the form of objects within several buckets. For this reason, before the actual download of the health data on the HCP app a request to receive a list of all the buckets that are used to store the citizen's health data is performed.
- Step 4 - List objects: This step is performed in order to receive a list of the encrypted health data that is stored on a specific bucket on the S-EHR Cloud.
- Step 5 - Download and decrypt health data object: As soon as the list of the objects (i.e. the encrypted health data) is returned from the S-EHR Cloud, the request to download it is performed. The encrypted health data Bundle is downloaded from the S-EHR Cloud and then decrypted locally on the HCP app with the encryption key that was obtained when the QR-code was scanned. Afterwards, the health data is stored locally on the HCP app.

6.3.2 R2D Emergency upload of encrypted health data to a S-EHR Cloud

This section describes the use of R2D Emergency protocol as a medium protocol for the communication and data exchange of encrypted health data between an HCP application and a S-EHR Cloud. In more detail, it describes the interactions between the HCP application and the S-EHR Cloud, over the R2D Emergency protocol to upload encrypted data to the S-EHR Cloud of a patient.

A prerequisite of this sequence diagram (Figure 34) (shown in the previous one) is the successful execution of the preliminary steps that need to take place in order for an HCP to gain access to a citizen's health data during emergency situations.

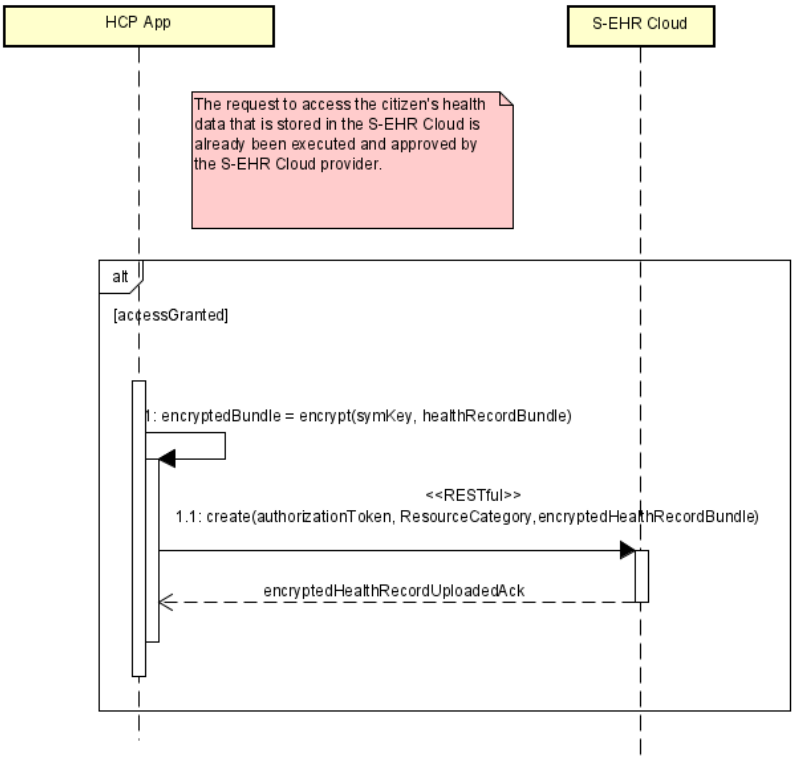

Figure 34 - R2D Emergency protocol interactions with respect to the upload of encrypted health data related to the emergency to the S-EHR Cloud from the HCP app during or after an emergency

Figure 34 depicts the sequence diagram of the R2D Emergency protocol as it is used for the upload of encrypted health data related to the emergency as soon as it ends from the HCP app.

Upload of encrypted health data related to the emergency to the S-EHR Cloud

- Step 1 - Encrypt and upload health data related to the emergency on the S-EHR Cloud: The health data that is created during an emergency or after its completion (i.e. the Patient Discharge Report) on the HCP application is encrypted with the symmetric key that is obtained from the QR-code and then uploaded on the S-EHR Cloud. The upload of this new content is stored on a bucket to the citizen's S-EHR Cloud that is specific to this emergency.

The following tables provide a complete description of all the operations of the R2D Emergency protocol's exposed methods on the R2DEmergency interface, along with the translation of those requests from the citizen's side to REST HTTP requests in order to communicate with a RESTful service such as the S-EHR Cloud.

6.4 R2D Emergency Specifications

This section contains the technical specifications of every operation of the R2D Emergency protocol introduced in the previous sections. The R2D Emergency is a protocol based on the HTTP specifications version 1.1, providing details about these standards is out of the scope of this document, complete knowledge of HTTP specifications [rfc 2616] [rfc 7540] is required for a complete understanding of the present deliverable.

6.4.1 Request to access health data

This operation allows an HCP to send a request to access the citizen's bucket during an emergency. An account is created for the health care institution that is providing care for the patient. This temporary account may be used by the HCPs of that specific institution during the emergency in order to download the citizen's health data, and/or upload new health data in a bucket dedicated for that specific emergency.

Endpoint and parameters

The following URL shows the structure of the request for gaining access to the patient's S-EHR Cloud:

```
POST [base]/hcp/requestaccess
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Emergency token (in JWT format) collected by the QR-code provided by the citizen
HCO-attrs	string	NO	Healthcare institution's attributes used to evaluate the request

Return Value

- JSON Web Token used by the HCPs of the Healthcare institution in order to access the citizen's health data of the citizen in the body of the response. The JSON response is structured as follows:

```
{
      "hcoEmergencyToken" : Healthcare Institution
      Authorization JSON Web Token
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Preconditions

- The citizen should have agreed to share their health data with authorized HCPs during emergency situations.
- The HCP should scan the QR-code found on the citizen in order to obtain the HCP authorization token, and the S-EHR Cloud URI information.

6.4.2 List of available buckets

This operation allows an HCP to retrieve the list of the buckets that the HCPs of this institution can gain access to. Different buckets may be created in emergency situations. The HCPs may access all concerning buckets.

Endpoint and parameters

The following URL shows the structure of the request for retrieving the available buckets:

```
GET [base]/hcp/buckets
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Health care institution authentication JSON Web token

Return Value

- List of buckets that the citizen's account has access to, in JSON format. The JSON response is structured as follows:

```
{
      "buckets" : [
            String: Bucket Name 1,
            String: Bucket Name 2,
            ...
      ]
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Preconditions

- The Healthcare Institution was granted access to the citizen's S-EHR Cloud.

6.4.3 List of health records in a bucket

This operation allows an HCP to retrieve the list of the objects (i.e. the encrypted health data) stored in a specific bucket.

Endpoint and parameters

The following URL shows the structure of the request for retrieving the objects stored in a specific bucket:

```
GET [base]/hcp/buckets/{$bucketName}
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Health care institution authentication JSON Web token

Return Value

- List of items contained in a bucket in JSON format. The JSON response is structured as follows:

```
{
      "bucket" : String: Bucket Name,
      "objects" : [
            String: Object Name 1,
            String: Object Name 2,
            ...
      ]
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Preconditions

- The Healthcare Institution was granted access to the citizen's S-EHR Cloud.

6.4.4 Metadata retrieval

This operation allows an HCP to retrieve the metadata of encrypted health data's information from the S-EHR Cloud.

Endpoint and parameters

The following URL shows the structure of the request for retrieving the metadata of an encrypted health data file:

```
GET [base]/hcp/{$bucketName}/{$objectName}/metadata
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Health care institution authentication JSON Web token

Return Value

- The metadata of the health data resource requested in JSON format. The JSON response is structured as follows:

```
{
      "object" : String: Object Name,
      "metadata": {
            "objectName": String: Object Name,
            "bucketName": String: Bucket Name,
            "size": Float: Size of the object in KB,
            "type": String: Object type,
            "dateAdded": Date: Date stored in S-EHR Cloud
      }
}
```

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Preconditions

- The Healthcare Institution was granted access to the citizen's S-EHR Cloud.

6.4.5 Health record download

This operation allows an HCP to download an encrypted health data record from the S-EHR Cloud and decrypt it locally on the HCP App.

Endpoint and parameters

The following URL shows the structure of the request for downloading an encrypted health data file:

```
GET [base]/hcp/{$bucketName}/{$objectName}
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Health care institution authentication JSON Web token

Return Value

- Encrypted Binary file.

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Preconditions

- The Healthcare Institution was granted access to the citizen's S-EHR Cloud.

6.4.6 Health record upload

This operation allows an HCP to upload an encrypted health data's information to the S-EHR Cloud.

Endpoint and parameters

The following URL shows the structure of the request for uploading an encrypted health data file:

```
POST [base]/hcp/upload?objectName={$ResourceCategory}
```

Additional parameters under the responsibility of the S-EHR are the header parameters that can be added to a request and that are used to provide contextual information affecting the request processing:

Header parameters available to S-EHR			
Name	**Type**	**Mandatory**	**Constraint**
Authorization	string	YES	Health care institution authentication JSON Web token

Body value

The encrypted bundle is stored in the body of the request.

Return Value

- void operation

Return Codes

- **200 Successful**: request successfully processed.
- **4XX 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**: requested resource not found.
- **406 Not Acceptable**: client requested a not supported content-type format.
- **5XX Internal Server Error**: the server encountered an unexpected internal error

Preconditions

- The Healthcare Institution was granted access to the citizen's S-EHR Cloud.

6.4.7 DICOM Support

This section describes the way HCPs can retrieve DICOM images from other Healthcare Organizations during emergency situations. The process starts with the citizen storing to the S-EHR Cloud a set of information for each Healthcare organization that has generated DICOM images for this citizen and supports the exchange of those images via a REST-ful WADO-RS service.

This set information includes the following:

- An authorization token that is obtained from the WADO-RS service. The WADO-RS Service comes with an authorization mechanism that generates such tokens, usually in the form of JWT (JSON Web Token).

- A consent signed by the citizen that allows the Healthcare Organization to share with other Healthcare Organizations the DICOM images it generated during emergency situations.

What is important to note is that since there might exist several Healthcare Organizations that contain citizen's DICOM Medical Images, multiple tuples of such tokens and consents are stored in the S-EHR Cloud.

When the emergency occurs the HCP of the Healthcare Organization that the citizen is taken to, performs a request to the S-EHR Cloud to obtain the above-mentioned information. With the use of this information, the HCP can perform requests to a WADO-RS server of the Healthcare Organization that contains DICOM Images of this specific citizen, to download a Study, a Series of a Study of a specific Instance of a Series. The procedure that is followed in order to perform those requests is the same as already described in the sections Get a Study, Get a Series, and Get an Instance accordingly.

More information regarding the authorization of Healthcare Organizations to access DICOM images stored in other Healthcare Organization during emergency situations, can be found on [D3.8].

7 CONCLUSIONS AND NEXT STEPS

The objective of this report was to deliver the final version of the design and the specification of the D2D and the R2D protocols. With the implementation of D2D it became 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 the R2D Access protocol works over the internet and defines the set of operations for acquiring the health data of a citizen from an EHR (e.g. a National EHR, or the EHR of a hospitals). Moreover, it is used for initial import of health data to the mobile device and for the subsequent periodic synchronization operations. The R2D Backup protocol defines the set of operations used by the citizens in order to safely back up and store their health data on the S-EHR Cloud and then restore it on another S-EHR App. In addition, the R2D Emergency protocol is also used for the communication with the S-EHR Cloud but from the HCP's perspective, allowing authorized HCPs to communicate with S-EHR Cloud providers during emergency situations and gain access to the health data of the citizen in need.

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. In the version of the D2D protocol that was specified in [D4.2], the D2D protocol specification was 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 his/her own operations, in order to implement the corresponding data exchange between a S-EHR and an HCP application. In the final version of the protocol, the independent nature of the D2D protocol has been continued, while the way that the messages are being exchanged between the S-EHR app and the HCP app have been restructured for providing a client-server nature to the protocol, where the HCP app can send requests of specific resources (client), and the S-EHR app will provide specifically designed responses (server). As for our next goals, we plan to perform additional evaluation of the overall D2D protocol specification, and finalize or redesign specific messages that may violate or delay the overall process.

As for the R2D Access protocol, the current version (version 3) represents a concrete step towards the completeness and maturity of the protocol. The boundaries and the characteristics of the protocol are well defined as well as the main requirement: define an easy to adopt protocol to enable citizens to download their health data. R2D Access provides several ways to allow S-EHR app developers to adopt different patterns to import data from an R2D Access server, choosing the one that best fits their needs: for low users that accept to import any health data, or for more experienced users that wants to download only specific health data required. Version 1 and 2 of the protocol supported only limited kinds of data and basic interactions, while version 3 now supports a wider set of managed health data types and also adopts more advanced FHIR operations that simplify the import with one operation a complex set of health data that share a common context. As for our next goals for the R2D Access Protocol, it is within our plans to continue the implementation of the libraries that may implement the functionality of the protocol.

Regarding the design and specification of the R2D Backup protocol, the exchange of encrypted health data between a S-EHR mobile application of a citizen and a S-EHR Cloud provided was made feasible over the internet. Moreover, it should be mentioned that the protocol has been specified in order to allow a citizen

to exchange all health data types that are supported by the other protocols. Regarding the future steps with respect to the R2D Backup protocol, we plan on continuing the implementation of the protocol in order to give the citizens additional abilities such as the capability to choose their preferred S-EHR Cloud provider.

Finally, this document also described the design and specification of the R2D Emergency protocol giving the ability of HCPs to access health data of a citizen that is stored in the S-EHR Cloud and access it on their HCP app over the internet. The R2D Emergency protocol has been specified in that way in order to also allow an HCP to have the ability to upload encrypted health data related to the emergency to a S-EHR Cloud. Similar to the R2D Backup protocol, the R2D Emergency protocol's specification allows an HCP the download of all health data types that are supported by the other data exchange protocols defined by the InteropEHRate project. In this context, regarding the future steps with respect to the R2D Emergency protocol, we plan on continuing the implementation of the protocol in a way that will minimize the response times and accelerate the data exchange process, something greatly crucial especially when in emergency situations.

REFERENCES

- **[D2.3]** InteropEHRate Consortium, *D2.3-User Requirements for cross-border HR integration - V3*, 2021. https://www.interopehrate.eu/resources/#dels
- **[D2.6]** InteropEHRate Consortium, *D2.6 - InteropEHRate Architecture - V3*, 2021. https://www.interopehrate.eu/resources/#dels
- **[D2.9]** InteropEHRate Consortium, D2.9-FHIR profile for EHR interoperability - V3. Note that D2.9, final version of interoperability profiles, is due at M36, December 2021. The previous version, D2.8-FHIR profile for EHR interoperability – V2, 2020, can be found at https://www.interopehrate.eu/resources/#dels
- **[D3.8]** InteropEHRate Consortium, D3.8-Specification of consent management and decentralized authorization mechanisms for HR Exchange - V2, 2021. https://www.interopehrate.eu/resources/#dels
- **[Marco]** Nalin, Marco, et al. "The European cross-border health data exchange roadmap: Case study in the Italian setting." Journal of biomedical informatics 94 (2019): 103183
- **[Europa]** Commission makes it easier for citizens to access health data securely across borders, https://ec.europa.eu/commission/presscorner/detail/en/IP_19_842
- **[eHealth]** eHealth Information Exchange for Citizens, https://www.dhs.pa.gov/contact/DHS-Offices/Pages/eHealth-HIE%20for%20Citizens.aspx
- **[SMR]** Shared Medical Records, https://integracareclinics.com/shared-medical-records/
- **[MedCom]** Medcom, https://www.medcom.dk/medcom-in-english/about-medcom
- **[NPO]** Swedish Medical Record, https://www.swedish.org/patient-visitor-info/medical-records
- **[Diraya]** Diraya, https://joinup.ec.europa.eu/collection/junta-de-andalucia/solution/diraya/about
- **[Direct]** Direct Secure Messaging, https://directtrust.org/what-we-do/direct-secure-messaging
- **[Carequality]** Carequality, https://www.himss.org/resource-environmental-scan/carequality
- **[CloudFax]** Cloud Fax, https://apps.centurylink.com/email-and-collaboration/cloud-fax#:~:text=Cloud%20Fax%20lets%20you%20securely,%2Dbased%20toll%2Dfree%20number.
- **[KONFIDO]** KONFIDO, https://konfido-project.eu/
- **[OpenNCP]** OpenNCP, https://openncp.atlassian.net/wiki/spaces/ncp/overview

- **[epSOS]** Cross-border health project epSOS: What has it achieved? , https://ec.europa.eu/digital-single-market/en/news/cross-border-health-project-epsos-what-has-it-achieved

- **[eIDAS]** eIDAS, https://ec.europa.eu/digital-single-market/en/policies/trust-services-and-eidentification

- **[Gavrilov]** Gavrilov, Goce, et al. "Cloud-Based Electronic Health Record for Health Data Exchange." Proceedings/8 th International conference on applied internet and information technologies. Vol. 8

- **[Masud]** Masud, M., & Hossain, M. S. (2017). Secure data-exchange protocol in a cloud-based collaborative health care environment. *Multimedia Tools and Applications, 77*(9), 11121-11135. doi:10.1007/s11042-017-5294-5

- **[Basjaruddin]** Basjaruddin, N. C., Rakhman, E., Kuspriyanto, & Renardi, M. B. (2017). Developing electronic medical record based on NFC. *Proceedings of the 2017 International Conference on Computer Science and Artificial Intelligence - CSAI 2017*. doi:10.1145/3168390.3168420

- **[Khan]** Khan, M. A., Cherif, W., Filali, F., & Hamila, R. (2017). Wi-Fi direct Research - current status and FUTURE PERSPECTIVES. *Journal of Network and Computer Applications, 93*, 245-258. doi:10.1016/j.jnca.2017.06.004

- **[Jin]** Jin, C., Choi, J., Kang, W., & Yun, S. (2014). Wi-Fi direct data transmission for wireless medical devices. *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*. doi:10.1109/isce.2014.6884461

- **[Mockel]** Mockel, Rico, et al. "An easy to use bluetooth scatternet protocol for fast data exchange in wireless sensor networks and autonomous robots." 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2007.

- **[Qiu]** Qiu, Han, et al. "Secure health data sharing for medical cyber-physical systems for the healthcare 4.0." IEEE journal of biomedical and health informatics 24.9 (2020): 2499-2505.

- **[Chong]** Shao, Chong, and Shahriar Nirjon. "Imagebeacon: Broadcasting color images over connectionless bluetooth le packets." 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI). IEEE, 2017.

- **[Chung]** Chung, M., & Ko, I. (2015). Data-Sharing method For multi-smart devices at close range. *Mobile Information Systems, 2015*, 1-11. doi:10.1155/2015/931765

- **[D3.4]** InteropEHRate Consortium, D3.4-Specification of remote and D2D IDM mechanisms for HRs Interoperability - V2, 2021. https://www.interopehrate.eu/resources/#dels

- **[D3.6]** InteropEHRate Consortium, D3.6- Specification of data encryption mechanisms for mobile and web applications - V2, 2021. https://www.interopehrate.eu/resources/#dels

- **[D3.8]** InteropEHRate Consortium, D3.8-Specification of consent management and decentralized authorization mechanisms for HR Exchange - V2, 2021. https://www.interopehrate.eu/resources/#dels
- **[D4.1]** InteropEHRate Consortium, D4.1-Specification of remote and D2D protocol and APIs for HR exchange - V1, 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, 2020. https://www.interopehrate.eu/resources/#dels
- **[D4.8]** InteropEHRate Consortium, *D4.8-Specification of protocol and APIs for research health data sharing - V1*, 2020. https://www.interopehrate.eu/resources/#dels
- **[D6.7]** InteropEHRate consortium. *D6.7: Design of a service for cloud storage of S-EHR content (S-EHR Cloud) - V1,* 2021. www.interopehrate.eu/resources
- **[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
- **[GOEP] Generic Object Exchange Profile,** https://www.mpirical.com/glossary/goep-generic-object-exchange-profile
- **[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 ASYNC API]** FHIR Asynchronous RESTful API, Website: https://www.hl7.org/fhir/async.html
- **[FHIR IPS SPECS]** FHIR International Patient Summary, https://build.fhir.org/ig/HL7/fhir-ips/index.html
- **[FHIR R4]** FHIR Data Model R4, http://hl7.org/fhir/R4/
- **[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/
- **[rfc 2616]** Hypertext Transfer Protocol -- HTTP/1.1, Internet Engineering Task Force (IETF), Jun 1999, https://datatracker.ietf.org/doc/html/rfc2616
- **[rfc 7540]** Hypertext Transfer Protocol -- HTTP/2.0, Internet Engineering Task Force (IETF), May 2015, https://datatracker.ietf.org/doc/html/rfc7540
- **[rfc 7240]** Prefer Header for HTTP, Internet Engineering Task Force (IETF), June 2014, https://tools.ietf.org/html/rfc7240#section-4.1
- **[DICOM]** Digital Imaging and Communications in Medicine, https://www.dicomstandard.org
- **[DICOMWEB]** DICOMweb, https://www.dicomstandard.org/dicomweb

- **[WADO-RS]** Web Access to DICOM Objects by RESTful Services, http://dicom.nema.org/medical/dicom/current/output/html/part18.html#sect_10.4 https://www.dicomstandard.org/dicomweb/retrieve-wado-rs-and-wado-uri
- **[CEF]** Connecting European Facilities, https://ec.europa.eu/inea/en/connecting-europe-facility
- **[DROPBOX API V2]** Dropbox API v2, https://www.dropbox.com/developers/documentation/http/documentation
- **[CHOOSER]** Chooser Dropbox, https://www.dropbox.com/developers/chooser
- **[SAVER]** Saver Dropbox, https://www.dropbox.com/developers/saver
- **[EMBEDDER]** Embedder Dropbox, https://www.dropbox.com/developers/embedder
- **[GOOGLE DRIVE API]** Introduction to Google Drive API | Google Developers, https://developers.google.com/drive/api/v3/about-sdk
- **[ACTIVITY API]** Google Drive Activity | Google Developers, https://developers.google.com/drive/activity
- **[DATAVAULTS]** DataVaults Empowering Secure Data Storage, Sharing and Monetisation, https://www.datavaults.eu
- **[IDS]** International Data Spaces | The future of data economy is here, https://internationaldataspaces.org/
- **[IDS-ASSOCIATION]** The association | International Data Spaces, https://internationaldataspaces.org/we/the-association/
- **[IDS-CONNECTOR]** IDS Components | International Data Spaces, https://internationaldataspaces.org/use/ids-components/
- **[DBX SHARING GUIDE]** DBX Sharing Guide - Dropbox, https://developers.dropbox.com/dbx-sharing-guide
- **[IDS]** International Data Spaces | The future of data economy is here, https://internationaldataspaces.org/
- **[OLIVEIRA 2020]** Tuler de Oliveira, Marcela & Bakas, Alex & Frimpong, Eugene & Groot, Adrien & Marquering, Henk & Michalas, Antonis & Olabarriaga, Silvia. "A break-glass protocol based on ciphertext-policy attribute-based encryption to access medical records in the cloud." 2020 Annals of Telecommunications.
- **[LI 2010]** Li, Ming & Yu, Shucheng & Ren, Kaili & Lou, Wenjing, "Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-Owner Settings." 2010 in Security and Privacy in Communication Networks, (2010): 89-106
- **[BANERJEE 2013]** Banerjee, Arindam & Agrawal, P. & Rajkumar, Rajasekaran, "Design of a cloud based emergency healthcare service model." 2013 in International Journal of Applied Engineering Research, (2013): 2261-2264.
- **[LOUNIS 2016]** Ahmed Lounis, Abdelkrim Hadjidj, Abdelmadjid Bouabdallah, Yacine Challal, "Healing on the cloud: Secure cloud architecture for medical wireless sensor networks." 2016 in Future Generation Computer Systems, vol 55, (2016): 266-277

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 centimetres between the 2 devices, or in a distance of no more than 10 meters?	There is not any preference

Table 5 - 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 his/her EHR with the newly derived data (i.e. the data that will emerge from his/her visit to the Physician)?	Yes

Table 6 - 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 7 - 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 he/she wants to access? How would the Citizen understand the Physician's demands?	By providing an initial list of attribute names

Table 8- User Interaction Requirements

JSON-schema for the D2D requests

The JSON-schema for the D2D requests is specified below:

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "description": "JSON schema defining a D2DRequest",
  "examples": [
    {
      "id": "ece6f044-addc-422d-8f48-7894f65d67ca",
      "header": {
        "itemsPerPage": 100,
        "timeStamp": "2021-07-24T10:55:44.700Z",
        "agent": "JRE 1.8.0_121 - Mac OS X 10.16",
        "protocol": "D2D",
        "version": "1"
      },
      "operation": "SEARCH",
      "parameters": [
        {
          "name": "CATEGORY",
          "value": "OBSERVATION"
        },
        {
          "name": "DATE",
          "value": "2021-07-24"
        }
      ]
    }
  ],
  "required": [
    "id",
    "header",
    "operation"
  ],
```

```
  "type": "object",
  "properties": {
    "id": {
      "$id": "#/properties/id",
      "default": "",
      "type": "string"
    },
    "header": {
      "$id": "#/properties/header",
      "description": "Header of a D2DRequest, contains descriptive data about the request",
      "examples": [
        {
          "itemsPerPage": 100,
          "timeStamp": "2021-07-24T10:55:44.700Z",
          "agent": "JRE 1.8.0_121 - Mac OS X 10.16",
          "protocol": "D2D",
          "version": "1"
        }
      ],
      "required": [
        "itemsPerPage",
        "timeStamp",
        "agent",
        "protocol",
        "version"
      ],
      "type": "object",
      "properties": {
        "itemsPerPage": {
          "$id": "#/properties/header/properties/itemsPerPage",
          "default": 0,
          "description": "A non negative value states that the requestor is asking for a paged result and it is
specifying how many items every page should contain",
          "examples": [
            100
          ],
          "type": "integer"
        },
        "timeStamp": {
          "$id": "#/properties/header/properties/timeStamp",
          "examples": [
            "2021-07-24T10:55:44.700Z"
          ],
          "type": "string"
        },
        "agent": {
          "$id": "#/properties/header/properties/agent",
          "examples": [
            "JRE 1.8.0_121 - Mac OS X 10.16"
          ],
          "type": "string"
        },
        "protocol": {
          "$id": "#/properties/header/properties/protocol",
          "default": "D2D",
          "examples": [
            "D2D"
          ],
```

```
            "enum": [
               "D2D"
            ],
            "type": "string"
         },
         "version": {
            "$id": "#/properties/header/properties/version",
            "default": "1",
            "examples": [
               "1"
            ],
            "type": "string"
         }
      },
      "additionalProperties": false
   },
   "operation": {
      "$id": "#/properties/operation",
      "default": "",
      "description": "An explanation about the purpose of this instance.",
      "examples": [
         "SEARCH"
      ],
      "title": "The operation schema",
      "enum": [
         "SEARCH",
         "READ",
         "WRITE",
         "CLOSE_CONNECTION"
      ],
      "type": "string"
   },
   "parameters": {
      "$id": "#/properties/parameters",
      "description": "Array of the parameters needed for the processing of the request",
      "examples": [
         [
            {
               "name": "CATEGORY",
               "value": "OBSERVATION"
            },
            {
               "name": "DATE",
               "value": "2021-07-24"
            }
         ]
      ],
      "type": "array",
      "additionalItems": false,
      "items": {
         "$id": "#/properties/parameters/items",
         "anyOf": [
            {
               "$id": "#/properties/parameters/items/anyOf/0",
               "description": "A parameter is composed by a couple of a name and a value.",
               "examples": [
                  {
                     "name": "CATEGORY",
```

```
                        "value": "OBSERVATION"
                    }
                ],
                "required": [
                    "name",
                    "value"
                ],
                "type": "object",
                "properties": {
                    "name": {
                        "$id": "#/properties/parameters/items/anyOf/0/properties/name",
                        "description": "An explanation about the purpose of this instance.",
                        "examples": [
                            "CATEGORY"
                        ],
                        "enum": [
                            "CATEGORY",
                            "SUB_CATEGORY",
                            "DATE",
                            "TYPE",
                            "SUMMARY",
                            "MOST_RECENT",
                            "ID"
                        ],
                        "type": "string"
                    },
                    "value": {
                        "$id": "#/properties/parameters/items/anyOf/0/properties/value",
                        "description": "The value of a parameter. In case it is a date it MUST  have this format 'YYYY-
MM-DD', in case it is the CATEGORY parameter, it MUST contains one of the following value:
'PATIENT_SUMMARY', 'IMAGE_REPORT', 'LABORATORY_REPORT', 'PATIENT',
'DOCUMENT_REFERENCE', 'DOCUMENT_MANIFEST', 'DIAGNOSTIC_REPORT',
'MEDICATION_REQUEST', 'CONDITION', 'IMMUNIZATION', 'ALLERGY_INTOLERANCE',
'OBSERVATION', 'ENCOUNTER', 'COMPOSITION', 'PROCEDURE'",
                        "type": "string"
                    }
                },
                "additionalProperties": false
            }
        ]
    },
    "body": {
        "$id": "#/properties/body",
        "type": "string",
        "description": "optional encrypted string containing health data sent to the S-EHR."
    }
},
"additionalProperties": false
}
```


JSON-schema for the D2D responses

The JSON-schema for the D2D responses is specified below:


```
{
  "$id": "http://example.com/example.json",
```

```
"$schema": "http://json-schema.org/draft-07/schema",
"description": "The root schema comprises the entire JSON document.",
"examples": [
    {
        "id": "bf0a4f17-62c6-4199-ba9e-55325794c138",
        "header": {
            "requestId": "21a67a11-524d-44ab-834b-936e13f14b28",
            "page": 1,
            "totalPages": 1,
            "timeStamp": "2021-07-24T14:25:42.407Z",
            "agent": "JRE 1.8.0_121 - Mac OS X 10.16",
            "protocol": "D2D",
            "version": "1"
        },
        "body": "encrypted body value",
        "status": 200,
        "message": "Successful"
    }
],
"required": [
    "id",
    "header",
    "body",
    "status"
],
"type": "object",
"properties": {
    "id": {
        "$id": "#/properties/id",
        "description": "Unique ID of the response",
        "examples": [
            "bf0a4f17-62c6-4199-ba9e-55325794c138"
        ],
        "type": "string"
    },
    "header": {
        "$id": "#/properties/header",
        "description": "An explanation about the purpose of this instance.",
        "examples": [
            {
                "requestId": "21a67a11-524d-44ab-834b-936e13f14b28",
                "page": 1,
                "totalPages": 1,
                "timeStamp": "2021-07-24T14:25:42.407Z",
                "agent": "JRE 1.8.0_121 - Mac OS X 10.16",
                "protocol": "D2D",
                "version": "1"
            }
        ],
        "required": [
            "requestId",
            "page",
            "totalPages",
            "timeStamp",
            "agent",
            "protocol",
            "version"
        ],
```

```
"type": "object",
"properties": {
  "requestId": {
    "$id": "#/properties/header/properties/requestId",
    "description": "ID of the request that generates this response",
    "examples": [
      "21a67a11-524d-44ab-834b-936e13f14b28"
    ],
    "type": "string"
  },
  "page": {
    "$id": "#/properties/header/properties/page",
    "default": 0,
    "description": "current page",
    "type": "integer"
  },
  "totalPages": {
    "$id": "#/properties/header/properties/totalPages",
    "description": "Total number of pages that compose the result",
    "type": "integer"
  },
  "timeStamp": {
    "$id": "#/properties/header/properties/timeStamp",
    "examples": [
      "2021-07-24T14:25:42.407Z"
    ],
    "type": "string"
  },
  "agent": {
    "$id": "#/properties/header/properties/agent",
    "examples": [
      "JRE 1.8.0_121 - Mac OS X 10.16"
    ],
    "type": "string"
  },
  "protocol": {
    "$id": "#/properties/header/properties/protocol",
    "default": "D2D",
    "description": "Name of the protocol",
    "examples": [
      "D2D"
    ],
    "enum": [
      "D2D"
    ],
    "type": "string"
  },
  "version": {
    "$id": "#/properties/header/properties/version",
    "default": "1",
    "description": "version of the protocol",
    "examples": [
      "1"
    ],
    "type": "string"
  }
},
"additionalProperties": false
```

```
    },
    "body": {
      "$id": "#/properties/body",
      "description": "Contains an encrypted string representing the requested health data",
      "type": "string"
    },
    "status": {
      "$id": "#/properties/status",
      "default": 200,
      "description": "Return code of the requested operation",
      "examples": [
        200
      ],
      "enum": [
        200,
        300,
        400,
        410,
        500
      ],
      "type": "integer"
    },
    "message": {
      "$id": "#/properties/message",
      "description": "An optional human readable message describing the outcome of the operation",
      "type": "string"
    }
  },
  "additionalProperties": false
}
```


R2D over eHDSI

This section is not part of the InteropEHRate specification, but defines an additional architectural hypothesis for extending eHDSI. It is 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 cannot 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 (Figure 35) shows a Citizen of Country A downloading his health data using the S-EHR app and downloading data from the NCP of his country.

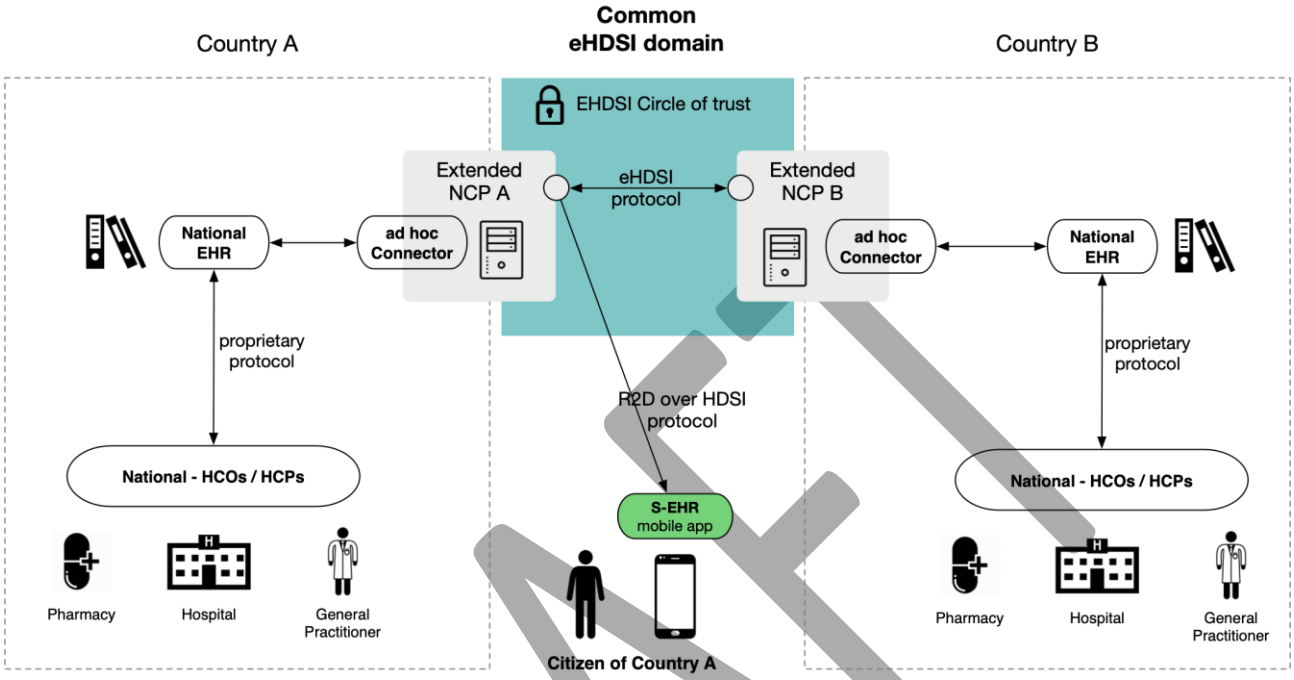

Figure 35 - Common eHDSI domain

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 retrieval of 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 retrieval of 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