



SHAPES

Smart and Healthy Ageing through People Engaging in supportive Systems

D 4.1: SHAPES Technological Platform (TP) Requirements and Architecture

Project Title	Smart and Healthy Ageing through People Engaging in Supportive Systems
Acronym	SHAPES
Grant Number	857159
Type of instrument	Innovation Action
Topic	DT-TDS-01-2019
Starting date	01/11/2019
Duration	48

Work package	WP4 – SHAPES Technological Platform
Lead author	Artur KRUKOWSKI (ICOM)
Contributors	ICOM: Eleni ZAROGIANNI & Ilia Pietri TREE: Tatiana Silva, David González & Joaquín Garcia HMU: Yannis Nikoloudakis VICOM: Luis Unzueta, Manex Serras, Gorka Epelde, Jordi Torres, Naiara Muro, Jon Kerexata, Garazi Artola, Gorka Epelde & Eduardo Carrasco GNO: Fotis Gonidis & Alexander Berler EDGE: Marco Manso, José Pires & Barbara Guerra FINT: George Bogdos & Anargyros Sideris
Version	1.3 (final, peer-reviewed)
Due date	M18 – 30/04/2021
Submission date	M18 – 28/04/2021
Dissemination Level	PU Public dissemination

Revision History

This section refers to incremental changes made from one release to another. Editors list refers to those providing such sections for integration in case anyone needs to get back to them for discussing changes and corrections.

Note that names in the table below do **NOT** represent a complete list of contributors. The full list of contributors is provided in the table on the front page of this deliverable.

Rev. #	Date	Editor	Comments
0.1	13/04/20	A. Krukowski (ICOM)	ToC & initial content
0.2	14/04/20	A. Krukowski (ICOM)	Requirement categories list
0.3	15/04/20	A. Krukowski (ICOM)	Corrected D8.4 references
0.4	05/02/2021	A. Krukowski (ICOM)	Template modification
0.5	09/03/2021	A. Krukowski (ICOM)	Template – new version
0.6	23/03/2021	A. Krukowski (ICOM)	Updated template
0.7	30/03/2021	A. Krukowski (ICOM) J. TORRES (VICOM) N. MURO (VICOM)	Added Table in Annex I CWDSS CWDSS
0.8	2/04/2021	A. Krukowski (ICOM)	Added section 5.3 Marketplace
0.9	6/4/2021	E. Carrasco (VICOM) B. Guerra, J. Pires & M. Manso (EDGE) I. Pietri, E. Zarogianni & A. Krukowski (ICOM) Y. Nikoloudakis (HMU) T. Silva (TREE), D. Gonzalez (TREE), G. Epelde (VICOM)	Section 3.2.1, 5.2.9 & 7.x Section 4.1 & 5.2.8 Section 5.2, 5.2.1, 5.2.7 & 6.x Section 5.2.6 Section 5.2.5
0.92	9/4/2021	A. Krukowski (ICOM) A. Krukowski (ICOM) Y. Nikoloudakis (HMU)	Added Section 2.2 Removed 7.1-7.5 (moved to D5.3) Added 5.2.5.2 & 5.3
0.931	12/4/2021	T. Silva (TREE) A. Krukowski (ICOM)	Updated 5.2.4 Fixed 5.3 -> 5.2.4.3 (datasets)
0.932	14/4/2021	E. Zarogianni (ICOM)	Sections 4.x, 5.2.1, 5.2.7, 6.x & Annex 3
0.933	15/4/2021	E. Carrasco (VICOM) F. Gonidis(GNO)	Section 3.2.1 update Section 6.2.4 & 4.1.1
0.934	19/4/2021	A. Sideris (FINT) S. Sari Sarlio-Siintola (LAU) T. Silva (TREE) + (PAL)	Sections 6.2.3, 6.2.2 & Annex 2 Section 3.1 Annexes 3-9
0.935	20/4/2021	E. Zarogianni & I. Pietri (ICOM)	Sections 5.4, 5.5 and 7
1.0	20/4/2021	A. Krukowski (ICOM)	First integrated version for peer review
1.1	22/4/20-21	F. Jesús Villanueva (UCLM) E. Carrasco (VICOM) A. Krukowski (ICOM)	Section 4.1.4 & 2.2 Corrections to section 3.2 Amendments after peer-reviews
1.2	26/4/2021	E. Zarogianni (ICOM)	Section 4 and 5.5.3
1.3	27/4/2021	M. Manso (EDGE)	Peer review of section 2.2

Keywords

Technological Platform, Requirements, Specifications

Disclaimer

This document contains information which is proprietary to the SHAPES consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or parts, except with the prior written consent of the SHAPES coordinator.

Table of Contents

Table of Acronyms and Abbreviations.....	vi
Executive Summary	vii
1. Introduction and Methodologies	1
1.1 Partners Involved in Task 4.1	2
1.2 Field of Application.....	2
1.3 Structure and Scope of the Document.....	3
1.4 Relation to other work in the project.....	4
2 System Technical Specifications.....	5
2.1 Methodology for Deriving Specifications and Architecture	5
2.2 Technical requirements and system specifications.....	6
2.3 Definition of the SHAPES Missions	44
3 Data Management and Privacy of Data.....	45
3.1 Generic Approach to SHAPES IoT and Medical Data Management	45
3.2 Management of Biometric Data	50
4 Standards Compliance in SHAPES	55
4.1.1 Introduction to FHIR	55
4.1.2 Introduction to Open mHealth Standard.....	57
4.1.3 Medical Device CE Certification.....	57
4.1.4 Cyber Security good practices for digital solutions	60
5 Semantic Interoperability for IoT & medical data	61
5.1 Introduction to semantics and interoperability	61
5.2 Semantic Interoperability in IoT.....	61
5.3 Semantic interoperability in the medical domain	63
5.4 Technologies in semantic interoperability	63
5.5 Information models for semantic interoperability.....	65
5.5.1 SHAPES Information Model for IoT data	65
5.5.2 The symbloTe Core Information Model.....	66
5.5.3 Forming the SHAPES Core Information Model	70
6 SHAPES Architecture.....	80
6.1 Conceptual SHAPES Architecture	80
6.2 Operational SHAPES Architecture	82
6.2.1 The symbloTe orchestration middleware.....	86
6.2.1.1 Introduction to symbloTe	86
6.2.1.2 The symbloTe architecture.....	86
6.2.1.3 Integration of symbloTe with ASAPA.....	94
6.2.1.4 Dependencies on other tools	94
6.2.1.5 Concluding Remarks	94
6.2.2 Gateway.....	95
6.2.3 Component: FINoT IoT platform.....	98
6.2.3.1 FINoT API	101
6.2.4 FHIR Medical Interoperability.....	101
6.2.5 Big Data Platform: Data Lakehouse & Analytics Engine	106
6.2.5.1 Communication between the Big Data Platform and the other components.....	111
6.2.5.2 Big Data Platform API details.....	112

6.2.5.3	Access to SHAPES datasets	113
6.2.5.4	Big Data Platform Stack	113
6.2.6	Component: ASAPA Authentication and Authorisation	115
6.2.6.1	Purpose and Scope	115
6.2.6.2	Basic Use Case	116
6.2.6.3	Component Architecture	117
6.2.6.4	Authentication	119
6.2.6.5	Security	120
6.2.6.6	Privacy Assurance	122
6.2.6.7	Component Deployment and Prerequisites	124
6.2.7	Component: Message Broker	124
6.2.8	Component: Front-end App	125
6.2.9	Add-on Service: Adilib Chatbot Building Platform	128
6.3	Marketplace	130
7	<i>Integration Strategy</i>	<i>132</i>
7.1	Components integration	132
7.2	Tool chain overview	132
7.2.1	Version control system	133
7.2.2	Code repository	133
7.2.3	Container management	134
7.2.4	Image repository	135
7.3	Integration testing	135
8	<i>Summary of Outcomes and Conclusions</i>	<i>137</i>
9	<i>Ethical Requirements Check</i>	<i>138</i>
	<i>References</i>	<i>139</i>
	<i>Annex 1 FInoT Middleware API for SHAPES</i>	<i>140</i>
	<i>Annex 2 Instructions for symbloTe compliance</i>	<i>159</i>
	<i>Annex 3 Digital Solutions - Technical Supplement</i>	<i>171</i>

Table of Acronyms and Abbreviations

Acronym	Description
AIV	Assembly, Integration and Verifications
AI	Artificial Intelligence
CO	Consortium only dissemination level
PU	Public Dissemination level
DoA	Description of the Action
EHR	Electronic Health Record
EC	European Commission
ICD	Interface Control Document
GDPR	General Data Protection Regulation
MRN	Medical record Number
PHR	Personal Health Record
PM	Person Month
QA	Quality Assurance
RAMS	Reliability, Availability, Maintainability, Safety of Means & People
RIA	Research and Innovation Action
SaaS	Software as a Service
SAaaS	Security Assessment as a Service
STC	Scientific Technical Coordinator
TP	Technological Platform
TRD	Technical Requirements Document
WP	Work Package

Executive Summary

The complete D4.1 deliverable reports on the results produced in Task 4.1. “*SHAPES Technological Platform (TP) Requirements and Mapping a Reference Architecture*”, containing the following sections that correspond to specific objectives of Task 4.1:

1. Translation of user needs and requirements from D3.7 “*Draft User Requirements for the SHAPES Platform*” and its subsequent iterations D3.8 and D3.9 into precise system specifications, taking into account also use cases as defined in D2.5 “*SHAPES Personas and Use Cases*” (due M6) and the definition of Pilot Themes as detailed in D6.1 “*SHAPES Pan-European Pilot Campaign Plan*”. Specifications will also consider suggestions from D8.4 “*Ethics Framework for Shapes solution*”.
2. Definition of the Reference Architecture of the core SHAPES Technological Platform, identifying its main components, their functionalities and interdependencies. The architecture definition is based on existing architecture models such as the IoT Reference Architecture proposed within the ISO IEC 30141:2018 standard and SHAPES partners’ experience from EU-funded *symbloTe* project and the IoT European Large-Scale Pilots Programme. It includes the identification of core components and categorises system specification and functionalities into devices, smart spaces, clouds and application domains. The resulting architecture has been designed to be fully modular, thus enabling high level of customisation to specifics of each Pilot Theme and its inherent use cases, as well as enabling support to additional pilots to be brought in through Open Calls, to be launched in Task 9.5: “*SHAPES Open Calls for Innovation and Collaboration*”.

Part 1 of the D4.1 report outlines the system specifications, as indicated above) including functional and non-functional specifications with references to user requirements from D3.7, D3.8 and D3.9.

Part 2 of the D4.1 report details the system architecture, describes each of its components in terms of functionalities, design, form, interfaces, placement in the architecture, allocation to the core or customised part of the system etc.

The **Appendices** provide supplementary info regarding technical aspects of Digital Solutions from SHAPES that have not been included in deliverable D5.2, such that they can be made available to developers before the extended version of D5.3 will be submitted by WP5 by month M24. They also provide API and integration instructions for integration of Digital Solutions with FINoT and symbloTe interoperability platforms.

NOTE: considering that deliverable D4.1 is the main and only document produced in WP4 that describes technical SHAPES solution and no other document of such types is expected to be produced, the general agreement among partners was to consider the D4.1 as a living document, to be evolving throughout the project via co-creative process among users, pilot hosts and technology providers, as well as external entities engaged through Task 9.5, thus to lead to the most universal architecture, able to support scenarios anticipated in the project, while being flexible enough to support such that have not been predicted yet. As such a final version of D4.1 can be expected by M30 of the project when WP4 will be expected to conduct formal technical tests of the integrated SHAPES solutions.

1. Introduction and Methodologies

The D4.1 “*SHAPES TP Requirements and Architecture*” deliverable has been elaborated in response to the user requirements acquired in WP3, use case defined in WP2, pilot themes defined in WP6, as well as ergonomics and ethical recommendations produced in WP8.

The first objective of D4.1 is to translate User Requirements into technical system specifications, taking into account the techniques planned to be used for the SHAPES system, to cover the needs, and more generally to offer a range of remote Medical Care services to the identified target Users. Hence the first sections of this document respond to the needs expressed by the WP3, first in D3.7 and later D3.8 and D3.9 revisions. of the ultimate solution and will be successively updated throughout all project phases.

It expresses the functions and performance of the product in the way the WP5 as a supplier of Digital Solutions perceives them, while respecting the expected use cases defined in WP2 and Pilot Themes in WP6. Services to be provided to users and constraints that should be respected while taking account of the know-how of the SHAPES consortium are provided, brought in as Digital Solutions in WP5 and technology enablers in WP4. System specifications are clearly labelled, such that design and architectural decision can be traced back to them, not to mention the ability to verify compliance of the system with specification during technical validation testing once the SHAPES system is ultimately implemented. The text that is not labelled shall be considered as an explanation/clarification needed better readers’ understanding.

As compared to user needs and requirements in D3.7-D3.9, the specifications describe generic functions as far as possible, and state generic functional requirements, together with strong configurability requirements. The high degree of configurability which is required will allow to develop a set of core modules, which can be easily adapted to the different test cases without additional software development. This general approach is deemed to be the most efficient to cover the different test cases of the SHAPES project, and thus to address the widest market of the medical care services for a wide range of professional applications.

The second objective of Task 4.1 is to develop the concept architecture of the SHAPES system in response to the system requirements and specifications produced earlier in Task 4.1. This activity started with acquisition of design information about all components brought into the project by technical partners, collecting inputs from both WP4 (regarding Technical Solutions) and WP5 (regarding Digital Solutions). A concept architecture describes both already available components and those to be developed and organises them into a concept architecture. It clearly identifies core components that need to be provided for any of the current and future Pilot Themes (WP6) and use cases (WP2), as well as custom components that need only be available for specific TPs and UCs. Furthermore, the document outlines value-added SHAPES interoperability mechanisms, the primary results expected from SHAPES

project. Lastly interfaces and protocols among SHAPES components are defined as well as for interoperability with external systems.

1.1 Partners Involved in Task 4.1

The list of consortium partners involved in Task 4.1 and deliverable D4.1 follows:

ID	Short Name	Role
1	NUIM	Contributes to user needs translation to specifications
4	AIAS	Contributes to ensuring applicability of the SHAPES architecture for Integrated Care Delivery
8	EDGE	Contributes to Open mHealth standard interoperability and SHAPES front-end Application, integration of eCare platform in SHAPES. Document peer-review
12	FINT	Leader of Task 4.7 Contributes with Gateway and FINoT IoT platform
13	GNO	Contributes with FHIR interoperability mechanisms as well as integration of its eHealthPass platform
15	ICOM	Leader of WP4, Tasks 4.1-3,8 and deliverable D4.1 Contributes with symbloTe interoperability platform
18	MedSyn	Contributes to integration of its IT Healthcare platform
20	OMN	Contributes to integration of its NOTiFY platform
22	PAL	Contributes to specific architectural considerations with respect to integration of robotics Digital Solutions
26	SciFy	Contributes to Marketplace integration into SHAPE architecture (referring to Task 7.4 to start on M19)
27	HMU	Leads Task 4.6. Contributes to ASAPA authentication mechanisms and Marketplace (as its task leader)
28	TREE	Leader of Task 4.4 Contributes with Data Lakehouse and Analytics Engine
29	UCLM	Contributes to integration of open-source gaming platform (StepMania) with UAvero & EDGE
35	VICOM	Leader of Task 4.5 Contributes with biometrics authentication mechanisms

Table 1 Partners involved in WP4 and their responsibilities

1.2 Field of Application

This document is applicable to the remaining work within WP4 aiming at defining the SHAPES system design, and preparing its AIV. It will be further derived in modules and sub-systems designs, and interface documents applicable for development of

these components. The technical system specifications can be used from the following perspectives:

- For teams in charge of developing solutions, it reflects the trade-offs made between performance, costs, time limits, and industrial feasibility. The specs table makes it possible to work out the conformity matrix in relation to the user requirements.
- For the teams in charge of AIV, this document makes it possible to build plans and procedures for integration and verification tests.
- For teams interacting with users, it serves as an input to produce User Manuals.

As stated in the Description of Work (DoW), the objectives were to:

- translate user needs and relevant operational and legal frameworks into clear technical system specifications for driving further project work on design, implementation, integration, testing and evaluations with end users
- develop a reference architecture for the core SHAPES TP, identifying its main elements, their functionality and their interdependencies
- identify core components and elaborate on how the required functionality can be categorized into the device, smart space, cloud and application domains

1.3 Structure and Scope of the Document

The structure of the D4.1 document is composed of the following Sections:

- **Section 1:** Introduction and Methodologies
Describes principles of SHAPES core platform operations
- **Section 2:** System Technical Specifications
Lists system specifications derived from user requirements in D3.9
- **Section 3:** Data Management and Privacy of Data
Described methodologies applied in SHAPES for the management of medical data and information compliant with GDPR regulation
- **Section 4:** Standards Compliance in SHAPES
Describes core standardisation considerations applied in SHPES
- **Section 5:** Semantic Interoperability for IoT & medical data
Describes two-path IoT and Medial Data interoperability mechanism approach adopted in the SHAPES architecture
- **Section 6:** SHAPES Architecture
Describes the overall approach to defining the SHAPES system architecture, with details of its sub-components and their current state of development
- **Section 7:** Integration Strategy
Outlines the strategy for seamless development of sub-components and the provisions for ensuring smooth integration

- **Section 8:** Summary of Outcomes and Conclusions
- **Annexes:** provide additional reference information regarding:
 - Annex 1: *Digital Solutions – Technical Supplement*, outlining additional information that has not been included in D5.6 and describing forms of their delivery & deployment, as well as deployment pre-requisites
 - Annex 2: *FINoT Middleware API for SHAPES*, detailing preliminary API used by FINoT platform for communication with other core components
 - Annex 3: *Instructions for symbloTe compliance*, providing preliminary step-by-step integration instructions for communicating with symbloTe platform from other core components and Digital Solutions

1.4 Relation to other work in the project

This deliverable is based on results from:

- **D2.5** “*SHAPES Personas and Use Cases*” (due M6)
User persona definitions were used in deriving specs and architecture
- **D3.9** “*Draft User Requirements for the SHAPES Platform*” (due M18)
Draft version with inputs from D3.7 and D3.8 was used to capture user needs
- **D6.1** “*SHAPES Pan-European Pilot Campaign Plan*” (due M6)
Pilot Theme definition were used to building a compliant form of an architecture
- **D8.4** “*Ethics Framework for Shapes solution*” (due M6)
Offered Ethical recommendations and oversight for architectural decisions

The results from D4.1 will then contribute to future work expected in:

- **WP5:** “*SHAPES Digital Solutions*”, including tasks and deliverables:
Tasks 5.2-8: proving technical platform for deployment of their Digital Solutions
D 5.3-4: final form of Digital Solution to fit SHAPES architectural provisions
- **WP6:** “*SHAPES Pan-European Pilot Campaign*”, including tasks and deliverables:
Task 6.2-8: providing them with architectural support for conducting pilot trials
D 6.2-10: helping to re-focus pilot trials to fit SHAPES architectural provisions
- **WP7:** “*Market Shaping, Scale-up Business Models and Socio-Economic Impact*”:
Task 7.4: impacting integration of SHAPES Marketplace into its architecture
D 7.4: impacting form of Marketplace to fit adopted SHAPES architecture

2 System Technical Specifications

2.1 Methodology for Deriving Specifications and Architecture

For a complex and sophisticated system, operational descriptions might be too tedious to handle for rapid prototyping and analysis of a system's behaviour. In such cases, it is more convenient to express the system on a higher level, somehow in a functional manner. This approach yields formal specifications that emphasize the system's general behavioural properties rather than its operational details. Moreover, it has a practical significance if the desired description can be derived or synthesized in a systematic way from the user requirements on system functions.

For the purpose of SHAPES, we adopt a process of deriving system specifications that follows the established industrial methodology for the description of system requirements and the synthesis of formal specifications from system requirements. The formal specifications can be taken as models of the system requirements. More generally, the main objective is to be able to derive an implementable or operational system description from a given high-level description on system functions. The proposed methodology can be fully automated, hence may improve both productivity and quality of system development. We have implemented a support system based on our approach and applied several practical system designs such as a complex platform built out of apparently disassociated and independent systems, as is the case of the SHAPES solution.

Literature on communicating systems, Formal Description Techniques (FDT), e.g. SDL [1], Estelle [2] and LOTOS [3], have been proposed as high-level specification languages. The conventional state machine-oriented approaches such as SDL and Estelle and algebraic approach such as LOTOS are suitable for the purpose of description and investigation of the total behaviour of systems. However, these approaches might be not suitable for rapid prototyping and flexible software development. Because we must enumerate and/or determine all system behaviours from an early stage of system design. Our objective is to give theoretical foundations and proposal of a flexible approach on the synthesis of formal specifications from user requirements written in an early stage of system design.

The methodology proposed for SHAPES differs from previous approaches mainly in theoretical discussions such as soundness and completeness and formal treatment, rather than practical methodology for description and use. Another related work is a synthesis of communicating processes from temporal logic specification by Manna and Wolper in [4]. Their approach is based on tableau-like method and completely different from our technical point of view, whereby SHAPES approach required applying both bottom-up and top-down methodologies simultaneously, having been based on existing solutions, some of those developed with Medical Services in mind while other ones for other domains and hence requiring adaptations, while user needs and expectations combined with use cases and Pilot Theme definitions need to drive the overall system requirements analysis and subsequently the architecture definition, ultimately leading to overall system development, integration and evaluation.

Therefore, our methodology starts from two opposite sides, on one hand we analyse user needs and expectations from WP3, combining them with use cases from WP2

and Pilot Themes from WP6. At the same time, we focus on building a database of Digital Solutions and other technological enablers, paying attention of both their current form, functionalities and interfaces, as well as on the initial analysis of needs for their extension to comply with expected objectives. By establishing links between such technological solutions and interfaces among them, we aim to ensure high-level of interoperability, even before reaching the stage of deriving the concept architecture. Such an approach is expected to enable SHAPES to be built such that it will be easily adaptable, extendable and interoperable with other external systems, some of them to be integrated and evaluated as part of Open Call for experiments in Task 9.5.

2.2 Technical requirements and system specifications

The technical requirements and system specifications detailed below aim to enable easy cross-referencing and manageability. They allow an easy extraction of subsets of requirements that are relevant to each of the core components and Digital Solutions, making it easier for developers, integrators and pilot hosts to comply with such requirements. Specifications are associated with core architectural components for easier traceability by their developers. We assumed compliance with Pilot Themes and Use Cases implied if explicitly mentioned in specs coming from user needs.

The specifications are primarily based on the outcomes of the user needs and requirements analysis performed in WP3 and detailed in D3.9 submitted on M18. As such, the table below follows a similar structure of classification groups.

Requirements from D3.9 have been translated into precise technical specifications where necessary while keeping a traceability back to the original table in D3.9. Many requirements had to be split into several functional components. In other cases requirements pointing to same technical concepts have been merged together, whereby references to all relevant user needs have been kept for traceability.

Several new specifications have been added to the list that originated from discussions in WP4 and describing commonly agreed design and implementation methodologies. Since not all specifications and user requirements apply to every SHAPES component, additional references to each of the sub-system of the extended SHAPES architecture have been added to the table. This way each of the components is able to extract a list of specs that apply specifically to such a component.

NOTE that the table below is still subject to changes as the development of the SHAPES system architecture and implementation of each of the component progresses, as part of the SHAPES co-creation process. Certainly, further discussions with end users in WP2 and WP3, as well as scopes and form of new solutions being brought through Open Calls in WP9 will further extend the list of SHAPES system requirements.

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
Business Specifications (B)														
Customer Service (CS)														
SPS-001	The SHAPES Platform shall adopt a customer logic (B2C and B2B) in its design and development.	High	X	X	X	X	X	X	X	X	X	X		B-CS-001 (D3.9)
SPS-002	The SHAPES Platform shall have its own Terms of Use and Services Policy.	High			X						X	X		B-CS-002 (D3.9)
SPS-003	The SHAPES Platform shall have its own Privacy Policy, observing applicable regulations, including the GDPR.	High	X	X		X	X	X		X	X			B-CS-003 (D3.9)
SPS-004	The SHAPES Platform should implement a customer support service.	Medium			X						X			B-CS-004 (D3.9)
Pricing (P)														

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-005	The SHAPES Platform shall be modular and configurable	High	X	X	X	X	X	X	X	X	X	X	Component will support operation as stand-alone and in various configurations	B-P-001 (D3.9)
SPS-006	The SHAPES Platform shall support various business models	High									X		Models to include: direct sales, licensing, subscription, PaaS	B-P-002 (D3.9)
SPS-007	The SHAPES Platform should support multiple subscription models	Medium									X		Subscriptions to include: free, standard, premium	B-P-003 (D3.9)
Marketplace (M)														
SPS-008	SHAPES Platform shall offer online marketplace.	High	X											B-M-001 (D3.9)
SPS-009	SHAPES Platform Marketplace should support registration of suppliers (supply) and clients (demand)	Medium	X											B-M-002 (D3.9)
SPS-010	SHAPES Platform Marketplace should select its suppliers based on their offer's effectiveness, affordability and added-value to SHAPES Platform.	Medium	X											B-M-003 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-011	SHAPES Platform Marketplace should support monetisation	Medium	X									e.g., fee per transaction	B-M-004 (D3.9)	
SPS-012	Suppliers of SHAPES Platform Marketplace shall abide and follow SHAPES Platform's Terms of Reference, privacy policy and ethics.	High	X										B-M-005 (D3.9)	
SPS-013	SHAPES Platform Marketplace should encourage transparent competitiveness.	Medium	X										B-M-006 (D3.9)	
SPS-014	SHAPES Platform Marketplace should contribute to building economies of scale	Medium	X									e.g., create supply chains	B-M-007 (D3.9)	
SPS-015	SHAPES Platform Marketplace should prevent monopolisation by vendors	Medium	X										B-M-008 (D3.9)	
SPS-016	SHAPES Platform Marketplace should contribute to dynamics of local economies	Medium	X									Include: aggregation of offers based on location and geographical reach	B-M-009 (D3.9)	

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
Sustainability (S)														
SPS-017	SHAPES Platform shall support its inherent sustainability	High	X	X	X	X	X	X		X	X	e.g., economic, financial, social and environmental	B-S-001 (D3.9)	
SPS-018	SHAPES Platform shall comply with universal accessibility policies	High									X	e.g., consider the public authorities' role wrt subsidising schemes	B-S-002 (D3.9)	
SPS-019	SHAPES Platform shall be based on common and open standards	High	X			X	X				X		B-S-003 (D3.9)	
SPS-020	SHAPES Platform should observe a cradle-to-cradle approach.	Medium											B-S-005 (D3.9)	
SPS-021	SHAPES Platform shall offer open APIs and use interoperability standards	High	X			X	X	X		X	X		B-S-006 (D3.9)	
Functional requirements (FR)														
General (G)														
SPS-022	The SHAPES platform shall facilitate integrated care	High								X			FR-G-2 (D3.9)	
SPS-023	SHAPES platform shall support multilingual user interface	High	X								X	X	FR-G-3 (D3.9)	

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-024	Shapes shall support vocational well-being	High									X		FR-G-4 (D3.9)	
Health support (HS)														
SPS-025	The SHAPES platform may provide a technical support on 24/7 bases	Low									X		FR-HS-1 (D3.9)	
SPS-026	SHAPES platform shall offer usage tutorials and help cards	High	X						X	X			FR-HS-2 & FR-IS-7a (D3.9)	
SPS-027	Digital Solutions shall provide usage tutorials and help cards including devices they use.	High								X	X		FR-HS-2 (D3.9)	
SPS-028	SHAPES platform may comply with best practices from ProACT project	Low	X							X	X		FR-HS-2 (D3.9)	
SPS-029	SHAPES platform should support health data management (collection, sharing and processing)	Medium	X	X			X	X		X	X		FR-HS-3 (D3.9)	
SPS-030	SHAPES platform shall comply with private data protection of the GDPR regulation	High	X	X	X		X	X		X	X		FR-HS-3 (D3.9)	

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
SPS-031	SHAPES platform shall ensure that private data is stored only within EU Member States and other countries considered as GDPR compliant	High	X	X			X			X	X		FR-HS-3 (D3.9)
SPS-032	Wrist monitors and/or smart watches with emergency function may be supported	Low							X		X		FR-HS-3 (D3.9)
SPS-033	All classes of users shall be able to review the historical data	High									X	X	FR-HS-3 (D3.9)
SPS-034	Data management should be integrated into the patients dashboard	Medium										X	FR-HS-3 (D3.9)
SPS-035	Devices counting steps as part of daily activity monitoring should be supported	Medium	X					X			X		Devices such as pedometers, preferably wrist-wearable. FR-HS-3a (D3.9)
SPS-036	Devices monitoring daily activity including sports ones should be supported	Medium	X					X			X		Devices such as fitness trackers (e.g. Garmin Vivosmart 4 on eHealthPass or Xiaomi Mi 3 Band on eCare) FR-HS-3b (D3.9)

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
SPS-037	Devices measuring and/or manually-entering water intake should be supported	Medium	X					X			X	Devices such as Smart drinking bottles connected to an app.	FR-HS-3c (D3.9)
SPS-038	Devices recording sleep quality should be supported	Medium	X					X			X	Devices such as wrist-wearable fitness trackers recording sleep quality and advising how to improve sleep quality.	FR-HS-3d (D3.9)
SPS-039	Devices tracking nutrition should be supported	Medium	X					X			X	Devices allowing either scanning a barcode using a mobile phone and/or manually entering types of consumed meals. Optionally to specify if meal was consumed in whole or partially.	FR-HS-3e (D3.9)
SPS-040	Medication tracking should be supported	Medium	X								X	Is to some extent part of Digital Solution under PT-03_all	FR-HS-3f (D3.9)
SPS-041	SHAPES shall support monitoring of vital signs (weight, temperature, blood pressure, blood glucose, bio impedance, heart rate, blood oxygen level, etc.)	Medium	X					X			X		FR-HS-3g (D3.9)
SPS-042	SHAPES should support manual data entry	Medium									X		FR-HS-3h (D3.9)
SPS-043	SHAPES shall support risk assessment and action plans	High								X	X		FR-HS-4 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FIoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
	as part of its data processing of health data													
SPS-044	SHAPES should support use of questionnaires as self-assessment tools	Medium									X		FR-HS-5 & FR-IS-3 (D3.9)	
SPS-045	SHAPES may offer support in dealing with legal issues may be offered (e.g. care plans)	Low									X		FR-HS-6 (D3.9)	
SPS-046	SHAPES should support reminders	Medium									X	e.g. medication, clinical readings, appointments, activity reminders	FR-HS-7 (D3.9)	
SPS-047	SHAPES may support wearable sensor devices for pain management	Low						X			X		FR-HS-8 (D3.9)	
SPS-048	SHAPES should offer alerting mechanisms about medical risks & emergencies	Medium									X	Targeting care receiver, care givers and health care professionals and linked to vital health sign monitoring, e.g. blood pressure, heart rate, blood glucose, oxygen saturation. To be categorised as red/amber/green	FR-HS-9 (D3.9)	
SPS-049	SHAPES may support maintenance of clinical devices	Low						X			X	e.g. calibration, make, model number	FR-HS-10	

ID	Specification	Priority	Applicability								Clarification	Reference	
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FIoT	Gateway	Message Broker	Data Lake & Analytics			Digital Solutions
SPS-050	SHAPES access devices may be ergonomic	Low								X	X		FR-HS-11
SPS-051	SHAPES should support Predictive Medicine	Medium							X	X		Prediction of risk of health events such as decompensations in patients with heart failure, exacerbations of COPD, and hypo/hyperglycaemia in patients with diabetes. Use of smart data analytics and predictive algorithms and Ambient Intelligence Health and Wellness Platform should be considered.	FR-HS-12
SPS-052	SHAPES should support monitoring of appliances	Medium	X			X	X			X		e.g. on/off, duration of use	FR-HS-13
Information services (IS)													
SPS-053	SHAPES access devices may be user friendly	Low								X	X		FR-IS-1 (D3.9)
SPS-054	Video conferencing should be supported in SHAPES	Medium								X			FR-IS-2 (D3.9)
SPS-055	Voice and chat interaction should be supported using chat bots	Medium								X	X		FR-IS-3 (D3.9)
SPS-056	SHAPES should support allocating tasks to healthcare professionals via dashboard	Medium								X	X	Includes marking as ‘ongoing’, ‘complete’, ‘further follow-up required’ or ability to leave notes e.g. handovers/updates	FR-IS-4 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference	
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App			
SPS-057	Scheduling of tasks for different users should be supported in SHAPES	Medium										X	e.g. care receiver, care giver, etc.	FR-IS-4a (D3.9)	
SPS-058	SHAPES may motivate care receiver	Low										X	To be provided on a regular basis or according to other data, completion of tasks, etc.	FR-IS-4b (D3.9)	
SPS-059	User friendly dashboard should be offered to care receivers and care takers	Medium										X	X	Option to store preferences, personalise the use.	FR-IS-5 & FR-IS-6 (D3.9)
SPS-060	Management of patient profile data should be supported in SHAPES	Medium										X		e.g. age, gender, degree of dependence, individual top three challenges as well as info about health condition based on 1) a medical report; 2) told by care receiver (who can be more or less sure); 3) told by the caregiver (again, who can be more or less sure)	FR-IS-6a & FR-IS-6d (D3.9)
SPS-061	Management of care giver profiles should be supported in SHAPES	Medium										X		e.g. age, educational degree, distance to reach care receiver, access to internet, technological skills, type of care, duration and frequency of care, number of caregivers, relationships etc.	FR-IS-6b & FR-IS-6c (D3.9)
SPS-062	SHAPES should offer social and information space	Medium	X										X	Connecting care receivers with peers and networks of interest to exchange info and socialise.	FR-IS-7 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-063	SHAPES should offer tutorials on healthy habits	Medium	X									X		FR-IS-7b (D3.9)
SPS-064	SHAPES should offer info where and how to receive organisational, administrative and/or legal support	Medium	X									X		FR-IS-7b (D3.9)
SPS-065	SHAPES may offer environmental, pollution and weather info, relevant to older population in a given area	Low	X									X		FR-IS-7c (D3.9)
SPS-066	SHAPES may offer info about cooking on TV programme, relevant to older population in a certain area	Low	X									X		FR-IS-7c (D3.9)
SPS-067	SHAPES may offer search for contacts	Low	X									X	e.g. to share the lunch break with to break loneliness	FR-IS-7d (D3.9)
SPS-068	SHAPES may offer info about activities in the community	Low	X									X		FR-IS-7e (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-069	SHAPES may offer info about support services (mental and physical help)	Low	X								X	Targets care receivers and (informal) care givers, linking them with services provided under FR-HM-1.	FR-IS-8 (D3.9)	
SPS-070	SHAPES should offer social games	Medium	X								X	This could for example relate to the number of activities conducted per week.	FR-IS-9 (D3.9)	
SPS-071	SHAPES should offer reminders and suggestions for games/activities	Medium	X								X		FR-IS-9a (D3.9)	
SPS-072	SHAPES should keep logs of access to personal data	Medium		X							X	All data changes/views should be retrievable. Includes e.g. who has seen/modified personal data and when.	FR-IS-10 (D3.9) & ET20 (D8.4 Privacy & DP 5)	
SPS-073	SHAPES should support authorisation management to data	Medium		X							X	Specifying who has access to what data. Only owners of data and those already authorised to access it can set such permissions.	FR-IS-10a (D3.9)	
SPS-074	SHAPES should implement roles to manage group authentication and authorisation	Medium		X							X			
SPS-075	SHAPES should offer possibility to delete access to the platform	Medium		X							X	X	Option to request deleting the account - compliant with GDPR regulation	FR-IS-11 (D3.9)

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
SPS-076	SHAPES shall offer option to view personal data stored in the platform	High		X					X	X	Compliance with GDPR regulation	FR-IS-11b & TS-DP-5 (D3.9), ET4 (D8.4 Privacy & DP 5)	
SPS-077	SHAPES shall offer option to edit personal data stored in the platform	High		X					X	X		ET5 (D8.4 Privacy & DP 5) & TS-DP-5 (D3.9)	
SPS-078	SHAPES shall offer option to request removing own personal info from the platform	High		X					X	X	Compliance with GDPR regulation - the right to be forgotten.	ET6 (D8.4 Privacy & DP 5) & TS-DP-5 (D3.9)	
SPS-079	SHAPES should permit users who left the platform to keep their data in its repository	Medium		X					X	X	Include option to define date up to which the data can be stored in SHAPES	FR-IS-11a (D3.9)	
SPS-080	SHAPES should provide training material for care providers	Medium	X							X	Integrate certification such as the European Care Certificate in the caregiver training.	FR-IS-12 (D3.9)	
SPS-081	SHAPES should provide WEB access from desktop/laptop/smartphone and tablet	Medium								X	X	FR-IS-12a (D3.9)	

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-082	SHAPES should offer programs of practical care for a person (with dementia) w.r.t. activities of daily living	Medium									X	To include feasible/uncomplicated strategies ('How-to') covering multiple activities of daily living (e.g. eating and drinking; dressing/undressing; bathing/personal care; toilet care, etc.	FR-IS-12b (D3.9)	
SPS-083	SHAPES should offer programs of managing psychological and behavioural changes for a person (with dementia)	Medium									X	To include feasible/uncomplicated strategies ('How-to') to manage the psychological and behavioural changes in the person with dementia (e.g. aggression, depression, anxiety, sleep difficulties, repetitive behaviour, wandering, etc).	FR-IS-12c (D3.9)	
SPS-084	SHAPES should support appraisal and validation of main persona's achievements/acquired knowledge and strategies.	Medium									X		FR-IS-12d (D3.9)	
SPS-085	SHAPES shall offer immediate feedback on answers to exercises	High									X	Aim to increase the main persona engagement with exercises, reduce errors and minimize ‘test anxiety’.	FR-IS-12d (D3.9)	
SPS-086	SHAPES shall offer comprehensive feedback	High									X	Feedback should be simple e.g. using colour scheme (red for wrong; green for correct) and comprehensive (justification/explanation on why an	FR-IS-12d (D3.9)	

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
													answer might be more or less appropriate).	
SPS-087	SHAPES should offer adaptation and personalisation of training intervention plans	Medium	X											FR-IS-12e (D3.9)
SPS-088	SHAPES should offer guidance on issues related to dementia and dementia care	Medium	X											FR-IS-12f (D3.9)
SPS-089	SHAPES shall offer suggestions of core lessons in the program by default to new users	High	X										Can be performed or not according to his/her final choice.	FR-IS-12f (D3.9)
SPS-090	SHAPES should offer means of keeping track of lessons completed	Medium	X										e.g. star icon marks lessons added to a plan and a check icon marks lessons already concluded	FR-IS-12g (D3.9)
SPS-091	SHAPES should offer printing personalized booklets of user sessions	Medium	X											FR-IS-12h (D3.9)
SPS-092	SHAPES may support social messaging apps	Low	X										e.g. What’s App	FR-IS-13 (D3.9)
SPS-093	SHAPES may offer a discussion forum	Low	X										e.g. for exchanging opinions about certain topics	FR-IS-14 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-094	SHAPS may support joint exercises	Low									X		FR-IS-15 (D3.9)	
SPS-095	SHAPES may support multilingual voice translation	Low									X	X	e.g. for communication with doctors	FR-IS-16 (D3.9)
SPS-096	SHAPES may allow authorised care takers to access Social Network usage of their subjects	Low									X		e.g. to check if a person has been in touch with someone during the day (friend, family, neighbour etc.)	FR-IS-17 & FR-LS-10 (D3.9)
SPS-097	SHAPES should make use of Machine Learning Matching for matching info services to user needs	Medium									X		Making use of machine learning.	FR-IS-18 (D3.9)
Health maintenance support (HMS)														
SPS-098	SHAPES should offer mental exercises for care receivers and care providers	Medium									X		Offers digital mental trainings to care receivers. To be developed by health care professionals and to be monitored by care givers	FR-HM-1 (D3.9)
SPS-099	SHAPES should offer mood self-assessment	Medium									X		Differentiate between mood and assessing clinical anxiety/depression. To be self-completed with ratings and free text assessment options. A mood graph/mood history to be displayed to represent mood progress over time.	FR-HM-1a (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-100	Mood ratings of x or lower should direct the users to relaxation and/or cognitive reframing lessons.	Medium										X		FR-HM-1b (D3.9)
SPS-101	Various different relaxation exercises should be available with options based on muscular relaxation and on imagery-based relaxation.	Medium										X		FR-HM-1c (D3.9)
SPS-102	Relaxation exercises instructions should be available in text and audio.	Medium										X		FR-HM-1d (D3.9)
SPS-103	More than one mood status should be added per day.	Medium										X		FR-HM-1e (D3.9)
SPS-104	The scale for mood assessment should discriminate the numbers from 1 to 10.	Medium										X		FR-HM-1f (D3.9)
SPS-105	Communication through an interface showing pictures & words with associated sounds, that users can use to communicate	Medium										X		FR-HM-1g (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
	needs/feelings when not able to speak / move adequately should be provided													
SPS-106	Use of customizable communication cards should be supported	Medium									X		FR-HM-1h (D3.9)	
SPS-107	Ability for the caregiver to customize mental exercise difficulty level should be provided	Medium									X		FR-HM-1i (D3.9)	
SPS-108	SHAPES should offer music/video	Medium									X		FR-HM-1j (D3.9)	
SPS-109	Music/video should be available offline if network is not available	Medium									X		FR-HM-1j (D3.9)	
SPS-110	SHAPES should support physical exercises	Medium									X	Offers digital physical trainings to care receivers. To be developed by health care professionals and monitored by care givers.	FR-HM-2 (D3.9)	
SPS-111	SHAPES should support exercises for reflect different levels of personal fitness	Medium									X		FR-HM-2a (D3.9)	

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
SPS-112	SHAPES should offer exercises support videos	Medium									X	Explanation videos on the use of the exercises.	FR-HM-3 (D3.9)
SPS-113	SHAPES should support health literacy (maintaining good healthy diet)	Medium									X	Target values and recommendations to be developed by health care professionals. It should match with dietary habits (either sensors or manual indications).	FR-HM-4 (D3.9)
Living support (LS)													
SPS-114	SHAPES should support assisted mobility at home	Medium		X		X					X	Devices to assist the care receivers should be monitored via the platform	FR-LS-1 (D3.9)
SPS-115	Assisted mobility and devices should monitor movement outside/ travel outside	Medium		X							X	Alerts to authorised care givers	FR-LS-2 (D3.9)
SPS-116	SHAPES should offer fall detection sensing and alerting	Medium	X	X		X					X	Alerts to authorised care givers. To be developed by TREE using cameras from robots and/or data from wearables.	FR-LS-3 (D3.9)
SPS-117	SHAPES should support sensor monitoring high risk situations	Medium	X	X		X					X	e.g. fire, gas, electricity with alerts to care givers to take action	FR-LS-4 (D3.9)
SPS-118	SHAPES should offer monitoring of home environment	Medium	X	X		X					X	e.g. temperature, humidity, air quality, with option to set comfort/minimum settings.	FR-LS-5 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-119	SHAPES may match support needed with available volunteer support	Low									X	Will support care receivers and unburden care givers in matching volunteers with care receivers.	FR-LS-6 (D3.9)	
SPS-120	SHAPES should offer accessibility info in public spaces	Medium									X		FR-LS-7 (D3.9)	
SPS-121	SHAPES may access to info about (public) transport	Low									X		FR-LS-8 (D3.9)	
SPS-122	SHAPES may access to click-and-pay services	Low									X		FR-LS-9 (D3.9)	
Legal and Ethical Requirements for Technology Platform (ET)														
SPS-123	SHAPES shall offer a choice of multi-cultural avatars to represent users	High	X								X	Includes avatars representing different genders and cultures	ET2 (D8.4 Rights 3.1)	
SPS-124	SHAPES shall allow on-demand switching off/on sensors and services	High	X		X		X				X		ET3 (D8.4 Rights 3.1)	
SPS-125	SHAPES shall support requesting, obtaining and storing user consent	High									X	a) technical capabilities for requesting consent, b) consent is documented properly (obligatory), c) consents is centrally stored (optional) Employ: - BPPC (Basic Patient Privacy Consent)	ET13 (D8.4 Privacy & DP 5) & TS-DP-6 (D3.9)	

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FIoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
												- Consent Document or “Patient Privacy Consent Acknowledgment Document”		
SPS-126	SHAPES shall offer traceability of personal data	High	X			X	X			X	X	Includes data mapping and data flows.	ET14 (D8.4 Privacy & DP 5)	
SPS-127	SHAPES shall support automated decision-making	High									X	Capability to re-direct the decision to a manual process.	ET15 (D8.4 Privacy & DP 5)	
SPS-128	SHAPES shall support "Privacy by design and by default"	High	X	X	X	X				X	X	Implement privacy enhancing technologies, e.g. encryption, anonymization etc.	ET16 (D8.4 Privacy & DP 5)	
SPS-129	SHAPES shall comply with trustworthy AI guidelines	High		X						X	X		ET21 (D8.4 AI Ethics 4.3)	
SPS-130	SHAPES should use AI for self-diagnosis of SHAPES cyber-security protection	Medium		X							X		ET22 (D8.4 AI Ethics 4.3)	
SPS-131	SHAPES shall comply with common minimum cybersecurity rules for mobile and online services	High		X							X	X	ET23 (D8.4 Cybersecurity 6)	
SPS-132	SHAPES shall comply with WCAG 2.1 Standards and Universal Design principles	High	X	X		X	X			X	X	Perform formative, summative, and continuous evaluations. Test throughout the project lifecycle and any time new content is added or code is updated.	ET24 (D8.4 Persons with disabilities 3.3)	

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
	in designing and implementing processes													
Technical and security requirements (TS)														
General (G)														
SPS-133	SHAPES Digital solutions shall be able to send alerts and notify the care givers	High									X		Ensure interoperability of services	TS-G-1 (D3.9)
SPS-134	SHAPES shall offer user friendly and attractive interface	High									X	X	To stimulate user acceptance.	TS-G-2 (D3.9)
Accessibility (AC)														
SPS-135	SHAPES shall offer Ease of Use interfaces for both healthy and impaired users	High	X								X	X	Ensure accessibility for disabled and impaired users (screen magnifiers, text-to-speech, color combinations with high contrast etc.), including instructions and authentication. Platform to comply with principles of total conversation, i.e. provide text, voice and sign language and their real-time transfer.	ET1 (D8.4 Rights 3.1) & ET25 (D8.4 Persons with disabilities 3.3) & TS-AC-1 (D3.9)
SPS-136	SHAPES user interface to resemble technologies used by elderly in their every-day-lives	High									X	X	Platform will not be used if not easy to use. This will make it more likely that they will use it	TS-AC2 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference	
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App			
SPS-137	SHAPES user interfaces shall minimise need for interaction for accessing the required info	High										X	X	Platform should make information accessible using "least-clicks" rule; a double-menu (top and side) to be used for navigation	TS-AC-3 (D3.9)
Data Protection (DP)															
SPS-138	SHAPES shall offer detection of personal data breach	High			X							X	X	Capabilities to identify potential personal data breaches and identification of personal data breaches.	ET18 (D8.4 Privacy & DP 5)
SPS-139	SHAPES shall support restricting data processing	High	X			X	X					X	X	Data subject rights: right to restriction	ET7 (D8.4 Privacy & DP 5)
SPS-140	SHAPES shall offer info about third parties that have gained access to own private data	High	X			X	X					X	X		ET8 (D8.4 Privacy & DP 5)
SPS-141	SHAPES shall provide support for "the right to data portability "	High	X											Capability to transmit data to data subject/third party in a structured, commonly used and machine-readable format.	ET9 (D8.4 Privacy & DP 5)
SPS-142	SHAPES shall provide support for "the right to object"	High	X				X					X	X	1) ensure that info about automated decision-making is given to user (data subject) before process starts; 2) prevent data subject’s data to be part of profiling if a data subject has objected to profiling.	ET10 (D8.4 Privacy & DP 5)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FIoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-143	SHAPES shall provide support for "storage minimisation"	High	X				X			X	X	Technical capabilities to erase or anonymise personal data after relevant data retention period and that the data would be removed from all systems. Define automated functions if this is possible.	ET11 (D8.4 Privacy & DP 5)	
SPS-144	SHAPES shall provide support for "Data protection principles: accuracy"	High	X				X			X	X	Ensure that source of data is recorded.	ET12 (D8.4 Privacy & DP 5)	
SPS-145	SHAPES shall provide necessary protection of data	High										Data is protection, recovery and attribution. Employ cyber-security protection and access logs including consent processes, compliant with GDPR	TS-DP-1 (D3.9)	
SPS-146	SHAPES shall offer capability to detect a risk of potential data breach	High	X	X	X	X				X	X	Platform needs to be GDPR conform	TS-DP-2 (D3.9)	
SPS-147	SHAPES shall support IAM (identity and access management)	High		X							X	X	Technical and organisational security measure to ensure that iam can be used for limiting access to certain categories of personal data	ET19 (D8.4 Privacy & DP 5) & TS-DP-3 (D3.9)
SPS-148	SHAPES shall implement password based authentication	High		X							X	X	Platform needs to be GDPR conform	TS-DP-4 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-149	SHAPES shall support password management	High		X							X	X	Including managing forgotten passwords/ issuing new passwords	
Scalability (SC)														
SPS-150	SHAPES shall support scaling up its services in terms of geographical coverage	High	X								X		Platform needs to stay flexible and should be able to operate at different levels	TS-SC-1 (D3.9)
SPS-151	SHAPES platform shall be scalable such that to accommodate changing user data storage requirements	High	X				X	X		X	X		Aim to meet client needs	TS-CP-1 (D3.9)
SPS-152	SHAPES shall use a modular architecture	High	X	X	X	X	X	X	X	X	X	X	At an individual level, the SHAPES platform should be able to display its services at different levels of complexity, allowing people to get gradually used to the system. Link with ProACT project. Platform needs to stay flexible and should be able to cater for different needs.	TS-SC-2 (D3.9)
SPS-153	SHAPES shall support registration of new Digital Solutions	High		X							X	X		
SPS-154	The front-end app shall support detection of registered Digital Solutions	High		X							X	X		

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
SPS-155	The front-end app should support selection of Digital Solutions associated with user accounts	Medium		X						X	X		
SPS-156	SHAPES shall support registration of devices	High						X		X		Platform needs to operate with a range of external devices and services	TS-SC-3 (D3.9)
SPS-157	SHAPES shall support recognition of connected devices	High						X		X		Platform needs to operate with a range of external devices and services	TS-SC-3 (D3.9)
Adaptability (AD)													
SPS-158	SHAPES user interface shall support automatic adaptation to visual capabilities of the access device	High	X	X						X	X	Users should be able to hide aspects they do not need. GUI needs to be adaptable in terms of relevant content to be displayed and appearance, i.e. font size, contrast, etc. by the user.	TS-AD-1 (D3.9)
SPS-159	SHAPES shall be extendable in terms of new Digital Solution options to meet changing needs of users	High	X	X						X	X	Platform needs to meet client needs	TS-AD-2 (D3.9)
SPS-160	SHAPES platform shall support adaptation to needs of care ecosystem	High	X							X	X	Platform needs to meet client needs - recommendations of new functionalities	TS-AD-3 (D3.9)
Availability (AB)													

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-161	SHAPES shall ensure service continuity and reliability at 98% on 24/7 time bases	High	X	X	X	X	X	X	X	X	X	X		TS-AB-1 (D3.9)
SPS-162	SHAPES shall be functional and operational during Internet downtimes	High		X	X			X		X	X	X	Systems shall cache essential info to be able to operate with limited functionality without network connection to data centres	TS-AB-2 (D3.9)
SPS-163	SHAPES should offer capabilities to determine service downtimes of its components	Medium	X	X	X	X	X	X	X	X	X	X	Provide activities logins, monitoring interfaces, etc. with <ul style="list-style-type: none"> • information about important events • warning about system functionality degradation • failure of one functionality, component or whole SHAPES platform SHAPES should be able to monitor the external interfaces as well as internal failures.	TS-AB-3 (D3.9)
Reliability (RB)														
SPS-164	SHAPES platform and its components should support resuming normal operation after period of network downtime	Medium	X	X	X	X	X	X	X	X	X	X	After coming back online after scheduled or unscheduled downtime, users should be able to resume from last point.	TS-RB-1 (D3.9)

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-165	SHAPES should notify users about technical problems including network downtime	Medium	X								X	X		TS-RB-2 (D3.9)
Maintainability (MN)														
SPS-166	SHAPES should support self-updates	Medium	X	X	X	X	X	X	X	X	X	X	Due to the multitude of services, their update should be easy to maintain.	TS-MN-1 (D3.9)
SPS-167	SHAPES should support collecting performance logs for improving its service quality	Medium	X	X	X	X	X	X	X	X	X	X		TS-MN-2 (D3.9)
Usability (US)														
SPS-168	SHAPES should provide tools to simplify the installation of solutions	Medium									X	X		TS-US-1 (D3.9)
SPS-169	SHAPES GUI should use self-explanatory graphical elements linked with respective services	Medium									X	X	Prevent users getting lost and/or wanting to get quickly back to the beginning.	TS-US-2 (D3.9)
SPS-170	SHAPES GUI should support key word search	Medium									X	X		TS-US-3 (D3.9)
Interoperability (IO)														

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-171	SHAPES shall support open and interoperable standards	High	X			X	X						Using FHIR® Fast Health Interoperable Resources standard in core of SHAPES: http://www.hl7.org/implement/standards/fhir , while supporting extensibility to other standards e.g. OpenMHealth.	TS-IO-1 (D3.9)
SPS-172	SHAPES shall support exchanging information among Digital Solutions	High									X			TS-IO-2 (D3.9)
SPS-173	SHAPES services and application should be accessible using Android and iOS based mobile devices	Medium									X	X		TS-IO-3 (D3.9)
SPS-174	SHAPES should facilitate (mobile) access to health documents via IHE-MHD	Medium				X					X			Ts-IO-4 (D3.9)
SPS-175	SHAPES should support linking existing patient profiles via PDQm	Medium				X					X		PDQm (Patient Demographics Query for Mobile): http://goo.gl/dSkj7x	Ts-IO-5 (D3.9)
SPS-176	SHAPES shall support interconnecting patient data across communities via XCPD	High				X					X		- XCPD (Cross-Community Patient Discovery) - Health-Care Encounter Report Document. More info at: http://goo.gl/Z83DZH	Ts-IO-5a (D3.9)

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
SPS-177	SHAPES shall support retrieval of patient data in XCA, epSOS, CCA	High			X					X		- XCA (Cross-Community Access) - epSOS Patient Summary - N206Cross-Community Access (CCA) profile: http://goo.gl/iN41Nq	Ts-IO-5b (D3.9)
SPS-178	SHAPES shall support retrieval of user identities and profiles using XUA (Cross-Enterprise User Assertion)	High			X					X		XUA (Cross-Enterprise User Assertion): http://goo.gl/4i81NX	Ts-IO-5c
SPS-179	SHAPES components should support exchange of documents in XDR, XDM and XDS and HL7-CDA and CCD formats	Medium								X		Exchange of documents should be possible. Use for example - XDR (Cross-Enterprise Document Reliable Interchange) - XDM (Cross-Enterprise Document Media Interchange). - XDS (Cross Enterprise Document Sharing). - HL7 CDA v3 R2	TS-IO-6 (D3.9)
SPS-180	SHAPES shall support compatibility with different WEB browsers	High	X									Only WEB browsers certified to be secure will be supported, excluding older versions of Internet Explorer etc.	TS-IO-7 (D3.9)
SPS-181	SHAPES accessibility via WEB browsers shall exclude	High	X							X	X	This will prevent use of Adobe Flash, Shockwave etc.	

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
	plug-ins considered as insecure												
SPS-182	Devices connected to SHAPES platform components may have CE Mark certification	Low	X			X	X				X	It is up to Digital Solutions and their users to decide if such a certification is necessary. The SHAPES platform will support all devices irrespective of having the CE Mark or not.	TS-IO-8 (D3.9)
SPS-183	SHAPES platform may support certification of connected Medical Devices	Low	X			X	X				X	It is up to Digital Solutions and their users to decide if such a certification is necessary. The SHAPES platform will support all devices irrespective of having the CE Mark, Continua Alliance certification etc., or not.	TS-IO-8 (D3.9)
SPS-184	Device Directive shall conform solutions	High										Platform needs to comply with existing directives. Conform with the relevant European Device Directive	TS-IO-8 (D3.9)
Core-Component Operational Specifications (COS)													
SPS-185	Extended core information model (CIM) approach shall supported in SHAPES platform	High	X								X	Extensions to CIM are anticipated to be registered by Digital Solutions and/or 3rd party systems and services	
SPS-186	FHIR shall be used as a common data model for Medical Data exchange	High			X							Translations from/to other standards should be supported	

ID	Specification	Priority	Applicability									Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions		
	among components of the SHAPES platform												
SPS-187	Compatibility with OpenMHealth standard shall be supported by FHIR component	High				X							eCARE (D5.2)
SPS-188	symbloTe interoperability mechanisms shall be used for exchange of IoT data among SHAPES components	High	X				X			X	X		
SPS-189	FHIR interoperability component should be handing the transmission of medical data from Digital Solutions to Data Lakehouse	Medium				X				X	X	Optional I/F may be offered by TREE to other components to transfer IoT and Medical Data to Data Lakehouse, but only as a backdoor and under the explicit responsibility of the data owners and subject to them understanding all risks of employing such an interface	
SPS-190	IoT data processed by the core Analytics Engine shall be pre-loaded onto Data Lakehouse repository	High					X			X	X	Digital Solutions pass IoT data via FINoT to Data Lakehouse.	
SPS-191	IoT data are send to Data Lakehouse from Digital	High					X			X	X		

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
	Solutions only via FINoT platform													
SPS-192	Message Broker shall be employed to manage notifications among core SHAPES components	High	X	X	X	X	X	X	X	X	X			
SPS-193	Single sign-on mechanism shall be provided by SHAPES platform	High		X							X		ASAPA component manage authenticating SHAPES users, passing it to all components registered to the SHAPES platform.	
SPS-194	Authorisation to access user data shall be managed by a components where the data has been stored and/or acquired	High		X							X		Private data is only exchanged between source and target systems by a user authorised to access such a data.	
SPS-195	No private identifiable user IoT data shall be transmitted among SHAPES core components	High	X			X				X			Exceptions are Digital Solutions being original storage of personal data and the Gateway/Front-end-App taking direct part in the acquisition of measurements by Digital Solutions	
SPS-196	No private identifiable medical data shall be transmitted among SHAPES core components	High			X					X			Exceptions are Digital Solutions being original storage of personal data and the Gateway/Front-end-App taking direct part in the acquisition of measurements by Digital Solutions	

ID	Specification	Priority	Applicability									Clarification	Reference	
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions			Front-End App
SPS-197	Data Analytics shall process data excluding private identifiable information	High									X	Data is anonymised and/or individual identity replaced by unambiguous identifier, known only to the original Digital Solution		
SPS-198	Exchange of both IoT and Medical Data between Digital Solutions shall be done by direct messaging between them	High	X			X						X	symbloTe and FHIR perform mediation only, i.e. matching between data models used by Digital Solutions, but do NOT take part in exchanging actual data	
SPS-199	Data shared as datasets on Marketplace shall exclude all private identifiable information	High	X									X	Sources of data may include Data Lakehouse and FINoT platforms, which are expected to employ anonymization and scrambling mechanisms before such a data is shared	
SPS-200	Anonymization and scrambling mechanisms shall be employed by relevant SHAPES components	High	X				X				X	X		
IoT Cybersecurity for Digital Solutions														
SPS-200	All passwords shall be unique per device and per user	High										X	See European ETSI 303645 technical standard provisions for further details	

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
SPS-201	Means to manage reports of vulnerabilities should be supported	Medium									X	See European ETSI 303645 technical standard provisions for further details		
SPS-202	All software modules should be securely updateable	Medium									X	See European ETSI 303645 technical standard provisions for further details		
SPS-203	Automatic and periodic mechanisms should be used for software updates	Medium									X	See European ETSI 303645 technical standard provisions for further details		
SPS-204	The digital solutions should verify the authenticity and integrity of software updates	Medium									X	See European ETSI 303645 technical standard provisions for further details		
SPS-205	Sensitive securely parameters in persistent storage should be stored securely by the device	Medium									X	See European ETSI 303645 technical standard provisions for further details		
SPS-206	The consumer IoT device shall use best practice cryptography to communicate securely	High									X	See European ETSI 303645 technical standard provisions for further details		
SPS-207	All unused hardware, network and logical interfaces shall be disables	High									X	See European ETSI 303645 technical standard provisions for further details		
SPS-208	Personal data stored and/or transiting between a device	High									X	See European ETSI 303645 technical standard provisions for further details		

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
	and a service shall be protected with best practice cryptography													
SPS-209	Resilience should be built in to consumer IoT devices and services, taking into account the possibility of outages of data networks and power	Medium									X	See European ETSI 303645 technical standard provisions for further details		
SPS-210	The user shall be provided with functionality such that user data can be erase from the device in a simple manner	High									X	See European ETSI 303645 technical standard provisions for further details		
SPS-211	The manufacturer shall provide users with guidance on how to check securely set up their device	High									X	See European ETSI 303645 technical standard provisions for further details		
SPS-212	The digital solutions shall validate data input via user interfaces and/or transferred via APIs	High									X	See European ETSI 303645 technical standard provisions for further details		
SPS-213	The manufacturer shall inform users with clear and transparent info about what data is processed, how it is	High									X	See European ETSI 303645 technical standard provisions for further details		

ID	Specification	Priority	Applicability										Clarification	Reference
			Marketplace	symbloTe	ASAPA + biometrics	FHIR	FINoT	Gateway	Message Broker	Data Lake & Analytics	Digital Solutions	Front-End App		
	being used and for what purposes.													
SPS-214	Telemetry data may be monitored for security anomalies	low									X	See European ETSI 303645 technical standard provisions for further details		

Table 2 System specifications

2.3 Definition of the SHAPES Missions

The SHAPES system will be developed as a Core product, integrating Digital Solutions as defined in WP5, to be enhanced with customized modules to meet the User needs expressed in WP3 in several test cases as defined by WP2 and developed in the scope of the SHAPES project according to Pilot Themes defined in WP6 as:

- **PT 1:** Smart Living Environment for Healthy Ageing at Home
- **PT 2:** Improving In-Home and Community-based Care
- **PT 3:** Medicine Control and Optimisation
- **PT 4:** Psycho-social and Cognitive Stimulation Promoting Wellbeing
- **PT 5:** Caring for Older Individuals with Neurodegenerative Diseases
- **PT 6:** Physical Rehabilitation at Home
- **PT 7:** Cross-border Health Data Exchange Supporting Mobility and Accessibility for Older Individuals

For detailed information about:

- User Requirements: refer to deliverables D3.7, D3.8 and D3.9
- Digital Solutions: refer to deliverable D5.2 and future D5.3 (M24)
- Pilot Themes (PT): refer to deliverables from WP6, e.g., D6.2
- Use Cases & Personas: refer to deliverables from WP2, e.g., D2.5, D2.6 and D2.7

3 Data Management and Privacy of Data

3.1 Generic Approach to SHAPES IoT and Medical Data Management

Within WP4 two types of data models are being considered for application in SHAPES platform, one for IoT medical data and another one for Medical Information. Both have been agreed to follow common IoT standards like FIWARE while providing extensions to medical standards commonly used in the e-Health community such as HL7-FHIR specification.

On this basis, a **super set data model has been proposed**, which combines all the different data models used in SHAPES; it unifies diverse data information models. The original data model of symbloTe is extended in respect to what is missing with respect to standards used by the Digital Solutions of SHAPES. The different data standards can for example be FHIR (for clinical data), FINoT (based on FIWARE, for IoT non-healthcare data) or Open mHealth (for IoT healthcare and wellbeing data).

SHAPES platform will provide a **federated data model approach**, as in Figure 1. Each Digital Solution retains the access control of their own platform and data.

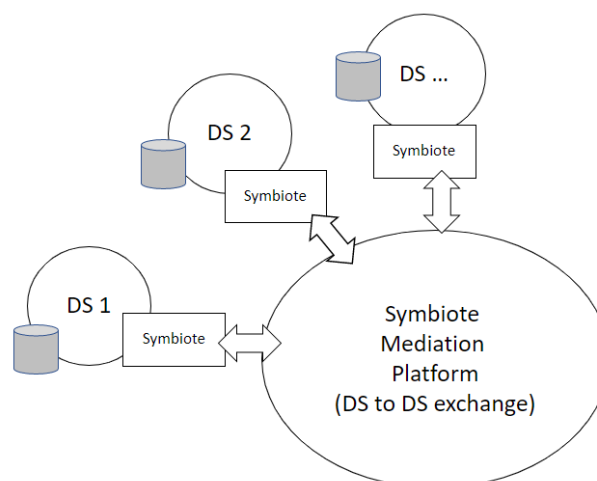


Figure 1: SHAPES federated data model approach

Hence symbloTe platform functions as a mediator between Digital Solutions for IoT type of medical; data while FHIR Interoperability component offers similar translations for exchanging Medical Information among interconnected systems. Those mechanisms define a mapping between different Data Models used on Digital Solutions aiming to exchange their data. Matching of parameters between Data Models used by each entity participating in the exchange of information thus enables them to understand the data that is being transmitted between them. This implies that in the SHAPES system, entities should use core SHAPES interoperability mechanisms in order to ensure seamless data exchange, unless both parties are known to use the same Data Model definitions. Only then they can safely exchange their IoT data and Medical Information.

The important aspect of the SHAPES interoperability mechanisms is that NO actual; data is transmitted to the core of the SHAPES platform. Only metadata needed to align

diverse data models is being provided, after that both entities are able to exchange their data and information bilaterally. This way issues of data privacy are reduced to a minimum, as there is no personal identifiable data sent to interoperability mediation components in SHAPES core. A summary of data management strategies used in SHAPES is provide below in a form of Q&A, summarising numerous discussions among WP4, WP5 and WP6 in order to align the understanding of how data is managed in the SHAPES platform between technology providers from WP5, architecture developers in WP4 and end users from WP6.

Where will the data be stored?

The **data will be stored in Digital Solutions**, i.e. where it has been originally captured, thus under direct and explicit control of patients themselves, their authorised physicians, care takers and/or Digital Solution providers.

Whether Digital Solutions can be deployed locally on premises or offered as cloud or remote service, will be decided for each use case, taking into considerations technical capabilities of individual Digital Solutions and their form of deployments.

In all use cases the **Digital Solutions are the hosts of the data**. The Digital Solutions are connected to the devices, which capture the data. The Digital Solutions exchange the data among each other via the core. The core is just the mediator, to make sure that two Digital Solutions can connect to each other and exchange information.

In the core we also have the **Data Lakehouse and the data analysis engine**. Its only purpose is the analysis of the data. This means that only the data which needs to be analysed by the AI engine, will be transferred to the Data Lakehouse, but most of the data to be captured within SHAPES stays in the Digital Solution of each use case. This data which goes to the Data Lakehouse & AI engine is anonymised. The primary mechanisms used for this purpose is obfuscation and referring to physical individual by ID used on the Digital Solution that stores the individual's profile. Only data subject to analysis is sent to Data Lakehouse.

The **Data Lakehouse is not a data repository per se**, even that it contains databases aiding in its service operation. **However**, to support generation of datasets in SHAPES that might be potentially used for research purposes, e.g. for developing models of chronic diseases, ageing health progression etc., TREE has agreed to store data acquired from Digital Solutions within Data Lakehouse for possible release to open community at the end of the SHAPES project. However, in the commercial version that is expected to spring out of the SHAPES project, such a repository is likely to be only a temporary one. After the analysis in the AI engine the results are transferred back to the Digital Solution.

Where will the non-IoT + non-medical data be stored (e.g. filled out questionnaires which we have on paper)? How do I get it to any of the storage devices?

All data captured in the pilots have to go through one of the Digital Solutions. And if this data is needed for the macro-level evaluation afterwards, it should also be sent to the Data Lakehouse to be analysed. This means that all the data collected by the researchers (also including questionnaires regarding the care giver conditions for example) will be collected by a specific Digital Solutions via an app. Either the older persons themselves will use an app to answer all the questions, or alternatively a care-giver will enter the answers (on behalf of the elderly) into the Digital Solution. The technical leader of each use case is responsible that all the data that should be transferred to the Data Lakehouse and the AI engine of SHAPES is collected digitally by the respective Digital Solution. The same refers for example to a smart

speaker. The technical leader of the respective use case should take care that the data of the smart speaker will be made available via a SHAPES Digital Solution.

In the case of the iSupport platform¹, this is a special situation, as the iSupport platform is a Digital Solution of an external (non-SHAPES) partner. It has to be analysed, how the data from the iSupport programme can be send to the Artificial Intelligence (AI) engine of SHAPES. It was originally agreed that iSupport would run separately from the SHAPES platform, but if the situation has changed, the technical partner can look into this problem and see how the data can be transferred. In all other use cases we have a Digital Solution provided by a SHAPES partner, who is responsible for the data transfer. The iSupport use case is the only exception.

Will the data in the data storage be available after the completion of the project for research purposes?

Discussions are on-going within the SHAPES consortium on the benefits and risks in maintaining data generated within SHAPES beyond the project's lifetime. SHAPES generated data could benefit future studies and research activities, despite this being outside the scope of the SHAPES project.

Technically it would be possible to maintain the data generated within SHAPES beyond the project's lifetime, thus potentially supporting future studies and research activities. However, one problem that needs to be overcome is the presence of personal data (for example, in questionnaires) – as such, a procedure for anonymising data would be required. Such anonymised and aggregated data might be **useful for a wide range of research purposes after the project**. Moreover, it might be a beneficial to collect not only the data which is going to be analysed in the AI engine, but also the data which will be stored in all the different Digital Solutions. This could be an important output of the project. While, technically, it is possible to store it in the Data Lakehouse, ethical and privacy constraints might restrict it. Ongoing discussions within SHAPES are considering dividing the platform up into two parts: one which is doing the analysis and the other one which is storing the anonymised data for further research work.

Currently analysis of data types that Digital Solutions might need to store, process and retrieve (such as results of data analysis by AI engine) from the Data Lakehouse is still on-going. If it is intended to keep the Data Lakehouse after the project's completion, the data will need to be harmonised. One reason is that the participant explicitly decides who will have access to the data (**risk of data leak**). Additionally, this will be more difficult for the commercialisation of SHAPES. In a commercial version **this repository would grow exponentially**. Additionally, maintaining data has associated costs that will have to be supported by the respective SHAPES partners.

The consortium is evaluating pros and cons in maintaining SHAPES generated anonymised data beyond the project. Moreover, SHAPES exploitation and commercialisation aspects will need to be considered, including the purpose of the data collection (this has to be agreed on with the participants), partner expectations (sustainable business models) and participant expectations (e.g., wish that their own data is still reachable to them). This type of personal data would not be stored in the big data platform.

A provision of external access to the SHAPES datasets is being considered in WP5, even if has not been referred to in the GA, which only refers to Open Data access regarding part of

¹ SHAPES deliverable D6.1 “SHAPES Pan-European Pilot Campaign Plan”

the Knowledge base, that could be publications related to internal research made in SHAPES. Since this would mean additional work for technology partners, such an option should be avoided in the course of the SHAPES project and only considered as pre-commercial feature. Additionally, a change of ethical approvals/informed consents already submitted would have to be made, which may risk future delays in the pilots.

Where and how does the anonymization take place?

A pseudo-anonymization process is defined as part of the pilot activities, conducted as part of WP6. For the moment, it was decided that only the pilot sites have the real names of the participants and that the Digital Solutions only have an alias (or ID-number as a coded identifier) related with the participant (thus, data is pseudo-anonymized). Nonetheless, to properly work, Digital Solutions might still need personal information such as gender, weight or height, but it should not be possible to be able to trace back the identification of the participant. In any case, to protect the confidentiality and privacy of their users, Digital Solutions either implement security mechanisms and/or own anonymization methodologies. In D5.2 it is explained how each Digital Solution works. D5.3 (submitted in M24) also addresses how Digital Solutions address security and privacy.

How can care-giver see which data belongs to which patient, when it is anonymised?

The Digital Solutions store the data both for the patient and for the care-giver. The patient gives an authorisation to the care-giver, that they can access their personal (health) data.

The anonymization is done to avoid the exchange of personal data, when two Digital Solutions exchange information. This means, that for example one Digital Solution contains all the personal data (e.g. gender, address, weight, etc.) and the other device is a sensor for the blood pressure and needs to connect to the Digital Solution. Then the data transfer is done via ID-number, so that no person (illegally) listening to this transfer can get any personal data and will not be able to trace it back to the real person. On the Digital Solution containing the personal data this information will be combined (e.g. the blood pressure and the rest of the information). Nevertheless, data stored in Digital Solutions can still be accessed.

This is possible for example via biometric data, which is stored on a tablet. If a person wants to access the information on two different tablets, the biometric data has to be stored on both tablets, as this information will not be transferred.

Where will the IoT data be stored? Where will the non-IoT data be stored?

Regarding the storage of IoT data, the main option is that each Digital Solution will store the data in its own backend. If in one of the use cases there is a **need to store data in a shared environment**, but not in the Data Lakehouse for GDPR related issues, then GNO offers this option as “storage as a service” Digital Solution. It seems to be the case that in most use cases there is no need for this extra shared storage service.

In case any pilot needs to store non-IoT clinical data, GNO is exploring the option of providing a storage service as a Digital Solution (in the form of a FHIR server).

This can happen if one Digital Solution wants to send **non-IoT medical data** to another Digital Solution or if one Digital Solution wants to send non-IoT data to the Data Lakehouse. In WP4 it has been discussed to adapt the FHIR standard (the interoperability standard) to allow this exchange of data. An example of such is e.g. the electronic records of the patients, like allergies, conditions or medications or demographic data. The status of that is that we will collect all these data from each use case and pilot and then GNO will create the equivalent

FHIR resources – a representation of this data into FHIR – and will make them available to the technical partners so that they can incorporate that in the Digital Solutions.

How device data be will transferred (i.e. all devices separately via the internet or via Bluetooth to a gateway and in a second step to a SHAPES Digital Solution)?

The transfer will depend on the Digital Solutions and device capabilities. Our priority is on wireless communication (**Bluetooth and/or Wi-Fi**). In special circumstances, a device might be able to directly connect (via e.g., Wi-Fi) to a backend server of the respective Digital Solution. However, for most cases, a device requires a connecting element (e.g., mobile phone or a gateway) to reach the backend server.

The **objective of the gateway is to collect the data of the “headless” devices**, this means all the devices that cannot reach (or don't have) a backend and cannot act as a Digital Solution themselves. So, in use cases involving “headless” devices, we need a gateway to connect them to the SHAPES Digital Solution. At the moment we are studying which type of interfaces and standards we can use to make this type of connection. We are considering Bluetooth and Wi-Fi. For simplicity and reliability, the gateway uses Wired internet connection to connect to the Digital Solution and to the SHAPES core system. It is noted that the Digital Solution is responsible for the anonymization process, not the gateway. The gateway is also useful to use in an environment in which Internet is not available all the time. In this circumstance, the gateway still connects and interacts with “local” devices.

Is it possible to use the cloud of a manufacturer as an intermediate between the device (e.g. blood glucose meter) and the SHAPES Digital Solution?

We cannot use data in SHAPES which is stored in a cloud in the US. It is not possible to share data with third parties outside the EU. The ethical team is analysing the conditions and requirements for possible use of third-party cloud within and outside of the EU.

A strict view in Horizon 2020 projects could point towards **forbidding the use of third party clouds**, because they are outside the control of the consortium. The preferred and recommended solution is that we should connect directly the device to the SHAPES Digital Solution, without going via a third-party manufacturer. However, if it cannot be avoided, when using third party clouds assurance must be provided that the third-party cloud provider is GDPR compliant, meeting same obligations as SHAPES partners in what concerns data privacy and confidentiality.

From GDPR perspective use of data storage technologies and repositories is permitted on condition that data is stored solely within EU member states and/or within 12 third countries considered by EC as “GDPR compliant”².

Which data is stored in SHAPES Data Lakehouse (storage service as a Digital Solution

The data that is stored in the SHAPES Data Lakehouse is only pilots' data which is need for the analytics. However, for research purposes we might decide as a consortium to store all the anonymised and aggregated data (see above). Discussions in WP5 and WP4 about Big Data Analysis applying AI techniques show possibilities of such techniques being used in Big Data Platform. TREE is willing to do that within the “Data Analytics Engine”. For that, we need to receive data for such analysis (IoT data and medical data). This means the pilot sites need to exchange additional data through the different Digital Solutions (and make sure the DS

² https://ec.europa.eu/info/law/law-topic/data-protection/international-dimension-data-protection/adequacy-decisions_en

can collect such data), so the analysis is possible. Besides, some support regarding medical knowledge could be needed, depending on the use case.

Who has access to this data?

User to whom data belongs and other users that he/she authorises. Core component administrators might also need access for maintaining system operation, though they will have to be authorised by data owners. Analysis of the aggregated data for SHAPES consortium (not only partners doing AI, but research partners, caregivers, etc.) If accepted, the Big Data Platform could release the datasets for specific SHAPES researchers/partners through their API, for their analysis – always for research purposes (we would need to manage properly the authorisations). But of course, for such analysis, the EHR, reports, etc. need to be available in the Big Data Platform. So, we need to make sure this happens in the different Use Cases.

Will the data only be stored temporarily there - or until the end of the pilot/ project?

During SHAPES pilots, TREE considers option of storing data for the pilots/ use cases until end of project within its Data Lakehouse repository. In commercial operation, it will only be stored temporarily for the period of performing analytics.

Is it correct that one Digital Solution cannot access to other Digital Solutions data via Data Lakehouse house?

The Data Lakehouse does NOT take part in data transmission among Digital Solutions. Those roles have SymbloTe, FInoT and FHIR interoperability components for IoT and medical data respectively. Digital solutions communicate among one another via symbloTe as a mediator (section 6.2.1).

If that is true how/where do to the Digital Solutions exchange data from on to another?

Those roles have SymbloTe and FHIR interoperability components for IoT and medical data respectively (see chapter 1.1).

3.2 Management of Biometric Data

Biometrics is the science of recognizing the identity of a person based on the physical, physiological or behavioural attributes of the individual [8]. There are three main types of Biometrics modalities [7]:

- **Hard Biometrics:** Also considered Classic or Traditional Biometrics, with biometric traits such as face, fingerprints, voice and iris.
- **Soft Biometrics:** Concerned with physical, behavioural, or material accessories related to an individual, with biometric traits such as age, height, weight, gender, hair color, hairstyle, eye color, skin color, facial hair, tattoos, facial expressions, glasses, hats, etc. [6].
- **Hidden Biometrics:** Called also Intrinsic Biometrics, with biometric traits based on medical data invisible to the naked eye, such as bio signals, MRI images or X-Ray images [9].

Biometric data is personal data resulting from specific technical processing related to these attributes. Biometric data allows for or confirms the unique identification of an

individual. The GDPR prohibits the processing of biometric data for the purpose of uniquely identifying natural persons. The exemptions the GDPR provides are limited and very restrictive. Individual's explicit consent is an example of an exemption.

The SHAPES platform includes Digital Solutions (DSs) which might need to manage personal data which could be considered as hard, soft, or hidden biometric data, even though only the DS for user authentication (Vicomtech's FACECOG) is designed for the unique identification of an individual using hard and soft biometric data, by means of facial recognition techniques. The personal data managed by the rest of DSs is used for other purposes which preserve the anonymity of users.

The FACECOG DS has two types of procedures:

- (1) the identification of people
- (2) the authentication (verification) of registered users.

Even though both tasks could be done using the same kind of biometric data, due to the specific characteristics of the use cases in which these procedures are applied, there are some differences in each case, which are explained next.

The identification of people:

This refers to the 1:N face matching procedure, in which the captured facial images are processed to determine whether they correspond to any of the N registered users or not. The people who are not registered users, are not uniquely identified.

This procedure is expected to be used in the cases in which a robot moves around a place trying to find a specific registered user to remind one task, or patrols to check whether the people around are expected to be in that place or not. In the first case, once the robot localizes the user, and approaches him/her to start the interaction, if the user needs to access medical data, he/she will be requested to authenticate following the 1:1 authentication procedure. In the second case, the robot will not interact with the detected people. If the detected person is not identified as a registered user, it sends an alert to the user that verifies whether response is true or not.

In these two cases, the detection and determination of the people in the robot's environment is supported by the navigation applications of the robotic systems (PAL / KOM), as well as the DS developed by TREE for the detection and classification of relevant objects (in this case people) in the scenario, as well as the determination of their distance from the mobile platform. This detection and location allows the execution of the identification solutions in a more robust way.

The Digital Solution, presented in *D5.2 SHAPES Digital Solutions*, performs the detection of people and the different performed activities in different supervised areas covered by CCTV and/or by integrated cameras in robotic platforms, as well as their paths, as long as they are visible by cameras, as well as the determination of their distance from the mobile platform. This detection and location allows the execution of the identification solutions in a more robust way. This solution is based on deep learning models. Currently, the solution is based on a hybrid paradigm of the results of the use of two high-performance and robust networks, such as YOLO [14] or SSD [15]. These initial DS models are being adapted (retrained) to fit the SHAPES project needs and enhance the support provided to VICOM's identification solutions.

The biometric data used for identification with this procedure needs to be processed at high framerate, while the robot is moving. Thus, this means that this procedure will not be as accurate as the 1:1 authentication procedure, because the biometric data will be lighter to process (hence, less accurate) than the data used in the 1:1 authentication procedure, and because of the very uncontrolled environmental conditions for the targeted tasks (the person is not posing in front of the camera, the robot is moving, etc).



Figure 2: People detection

The biometric data derives from descriptive identity vectors (i-vectors) extracted from the targeted face images by deep neural networks (DNNs) trained with data not corresponding to any of the targeted people, but which can still be usable for re identification purposes of the targeted people with high accuracy. These i-vectors are abstract representations of a combination of the appearances of some of the people used for training the DNNs, but not from the targeted people. In other words, this approach allows matching people's faces if their identifying appearance derived from i-vectors is close enough. Obviously, this approach has limitations, such as the possibility of matching people that look too similar for the trained model. This might happen if the images used for training the DNNs do not cover properly the subtle dissimilarities among those specific people and/or because the DNN is not sophisticated enough. This procedure is only used as an initial approximation for identification, which is then verified by the 1:1 authentication procedure or by a human person, depending on the case.

It is not possible to infer the true face appearance of the targeted people from these i-vectors. However, using techniques such as [16] could reveal relevant personal information from the user, due to the similarity to the "closest" people, such as "gender", "skin color", "hairstyle", etc., which needs to be protected. Thus, these i-vectors are homomorphically encrypted [5] to add a higher level of security to preserve the privacy of users and prevent information leakage from the face templates, while maintaining their utility through template matching directly in the encrypted domain.

The authentication (verification) of registered users:

This refers to the 1:1 face matching procedure, in which the biometric data is processed to determine whether it corresponds to the claimed identity or not.

In this case, the required level of certainty to confirm the unique identification of an individual is much higher than in the previous case. Thus, the trained DNNs are more sophisticated (hence, heavier to process), and the person in front of the camera is required to pose, i.e., the person actively collaborates. This procedure considers multiple data modalities (RGB and depth images, from cameras such as Intel's RealSense) and "liveness" detection techniques, to avoid spoofing attacks [10]. Besides, as in the previous case, the biometric data is homomorphically encrypted [5].

User registration

Initially, the user must undergo a face enrolment process to register his or her face – and biometric information– with the authentication data required to connect from the SHAPES platform. This process has the following characteristics.

- The system requires that the camera sees the user's face looking at the camera.
- The face should be at a distance of around 0.5 meters from the camera.
- The DS will update the biometric data in two cases:
 - When the user successfully authenticates with the DS.
 - When it detects a close match (not fully successful authentication), but a passcode is subsequently entered to access the SHAPES platform.
- This registration process will also be used for the extraction of the lighter biometric data used in the 1:N identification of people procedure.

Principal aspects of biometric data management

In summary, taking into account the considerations explained above, the biometric data in SHAPES is handled like this, to be GDPR compliant:

- FACECOG is the only DS that uses hard biometric data for 1:N identification and 1:1 authentication purposes. The rest of personal data used in other DSs, which could be considered as soft or hidden biometric data, are used for other purposes (not for identification) and preserve the anonymity of users.
- There is a direct connection between each DS and the authentication server which uses its own protocol. So, no FHIR connection, no SymbioTe connection, etc. The biometric data for identification and authentication are stored locally–in the DS local devices or its gateway-, not shared anywhere else, only managed internally by DS.
- This DS connects with "people detection" Digital Solution provided by TREE, which supports the system to identify people in the scenario.
- The biometric data used for identification and authentication requires the user consent to be used for the unique identification of an individual.
- The people who are not registered users and whose facial images are processed by the 1:N identification procedure, are not uniquely identified.

- No facial images are stored, neither during the user registration process, nor during the identification and authentication processes.
- Facial images cannot be reconstructed from biometric data.
- The biometric data is homomorphically encrypted, to preserve privacy of users and prevent information leakage from templates

4 Standards Compliance in SHAPES

This section describes the most important standards that are commonly used for developing electronic health services that have been adopted in SHAPES core architecture for ensuring their seamless interoperability. It has not been intended to provide an exhaustive list of all relevant technical standards. The standards mentioned in this section will be referred to when describing specific methodologies adopted within core architectural components described in detail in section 6.

4.1.1 Introduction to FHIR

The FHIR (Fast Healthcare Interoperability Resources)³, is a specification designed to facilitate the exchange of healthcare-related information. It is a platform specification that defines a set of data formats and elements (resources) as well as an application programming interface (API) to allow the seamless exchange of electronic health records. FHIR has been created by Health Level Seven International (HL7), an organisation founded in 1987 and dedicated to the creation of framework and related standards for the exchange, integration, sharing and retrieval of electronic health information⁴. The HL7 FHIR standard is designed using existing HL7 standard/model(s) and major technological overhaul by leveraging modern web and mobile technologies such as the lightweight HTTP-based REST protocol, JSON, RDF, etc. FHIR is structured around the concept of resources and profiles⁵. Resources represent granular clinical concepts and are the basis for the exchangeable FHIR content. They contain a standard definition and human readable description about how they can be used. Example of resources are “Patient”, “Observation”, “Organisation”, “Medication”, etc. (Figure 3). The full list is available in the online FHIR guide⁶. FHIR standard is dynamic and can be adjusted to meet tailored needs and domains.

For that reason FHIR allows the creation of profiles, which are built on the available resources and are extended with additional information to cover the particular use case. This process is called profiling and takes place with the use of facilitating rules. Such rules specify which resources elements are included or not and what are the additional element which are added. In addition, rules can specify which API features and terminologies are used⁷.

FHIR exposes certain benefits which leads to its endorsement by organisations and companies worldwide. Specifically, it is focused on implementation and enables fast and easy solution development. It has been evolved from previous HL7 standards such as HL7 v2, HL7 v3 and CDAs and provide support for all of them, enforcing this way interoperability. Furthermore, it leverages the modern web-based standards such as HTTP-based RESTful protocol and JSON, XML, RDF which makes its adoption straightforward. The specifications are easily understood by clinicians and technical

³ <http://hl7.org/fhir/2016may/protocol.html>

⁴ <http://www.hl7.org/index.cfm>

⁵ <https://wiki.hl7.org/FHIR>

⁶ <http://hl7.org/fhir/resourcelist.html>

⁷ <https://www.hl7.org/fhir/profiling.html>

audience which in turn reduces the application development time and cost. FHIR is offered under Creative Commons licence and is open and free to use⁸.



Figure 3: Example of Patient resource⁹

In the context of SHAPES, FHIR is leveraged to enable seamless exchange of non-IoT medical data among the Digital Solutions and with the SHAPES platform. In addition, it promotes interoperability with third-party solutions that will integrate with the SHAPES ecosystem. Solutions that are FHIR compliant can out of the box interact with the SHAPES ecosystem and exchange information. Further information on how FHIR is utilised within SHAPES platform is provided in 6.2.4.

⁸ [http://www.hl7.org/documentcenter/public/standards/FHIR/FHIR for the C-Suite.4Sep19.pdf](http://www.hl7.org/documentcenter/public/standards/FHIR/FHIR%20for%20the%20C-Suite.4Sep19.pdf)

⁹ <https://hl7.org/implement/standards/fhir/2015Jan/summary.html>

4.1.2 Introduction to Open mHealth Standard

Open mHealth¹⁰ has been created based on the work initiated by Deborah Estrin and Ida Sim after publishing a Policy Paper in Science calling for an open ‘mHealth’ architecture in 2010. Open mHealth coped with the increasing rate of wearables, smart devices and Apps related with health and wellbeing being commercialised and adopted by users. This plethora of new devices and components generate a broad variety of data, addressing parameters beyond the traditional health/clinical domain. As a result, Open mHealth proposes an open data standard to make the patient-generated mobile health data more interoperable and accessible. Indeed, Open mHealth provides an open source specification adopting the Apache 2.0 license¹¹, which includes data schemas based on widely adopted formats and technologies (e.g., JSON).

Concerning clinical data, Open mHealth associates each clinical measure schema with the most precise code identified from standard clinical vocabularies that are being used for EHR data (e.g., SNOMED for diagnoses, LOINC or SNOMED for lab tests, and RxNORM for medications). For example, the blood glucose schema is annotated with the SNOMED code 365812005, which is referenced within the schema to a persistent URL that is hosted by the BioPortal terminology server at Stanford. Schemas that can describe either a single or an aggregate measurement are annotated with the code for the measurement (e.g., body weight is annotated with SNOMED code 363808001, which describes body weight measure). For digital health data that are not yet represented in standard vocabularies, Open mHealth does not reference a code. Given its open nature, Open mHealth has attracted attention from numerous players worldwide, including SHAPES partner EDGENEERING, that follows the Open mHealth specifications in its eCare Digital Solution.

Given its open nature, Open mHealth has attracted attention from numerous players worldwide, including SHAPES partner EDGE, that follows the Open mHealth specifications in its eCare Digital Solution. Open mHealth can also be integrated with other standards: for example, there is a working group in FHIR building interoperability components between Open mHealth and FHIR¹²: Also, Open mHealth developed a FHIR implementation guide describing how to use the Open mHealth Shimmer and the Open mHealth to FHIR server to make health data accessible to a FHIR SMART client, either in the native OmH schema format or as FHIR resources (typically FHIR Observations)¹³.

For more info refer to <https://www.openmhealth.org/documentation/#/omh-to-fhir/>.

4.1.3 Medical Device CE Certification

A list of regulations (May 2021) identified by WP8 to be relevant to the use of medical devices in SHAPES and especially of non CE-Marked ones is provided in Table 3.

¹⁰ Open mHealth: <https://www.openmhealth.org/organization/about/>

¹¹ Apache 2.0 license: <https://www.apache.org/licenses/LICENSE-2.0>

¹² <https://healthdata1.github.io/mFHIR/index.html>

¹³ <https://www.openmhealth.org/documentation/#/omh-to-fhir/>

Document	Valid from/to	Link
Medical Device Directive (MDD) - Council Directive 93/42/EEC of 14 June 1993	Implementation date: 1994, will be replaced by Medical Device Regulation EU 2017/745	https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A31993L0042
Medical Device Regulation (MDR) - Regulation (EU) 2017/745	Come into force 25 May 2017; Implementation date: 26 May 2021	https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32017R0745
EU directives on active implantable medical devices (90/385/EEC)	will be replaced by Medical Device Regulation EU 2017/745	https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:31990L0385
Directive 98/79/EC of the European Parliament and of the Council of 27 October 1998 on in vitro diagnostic medical devices	from 7 December 1999 to 16 May 2022	https://eur-lex.europa.eu/eli/dir/1998/79/oj
Regulation (EU) 2017/746 of the European Parliament and of the Council of 5 April 2017 on in vitro diagnostic medical devices and repealing Directive 98/79/EC and Commission Decision 2010/227/EU	Come into force 25 May 2017; Implementation date: 26 May 2022	https://eur-lex.europa.eu/eli/reg/2017/746/oj
REGULATION (EU) No 536/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 16 April 2014 on clinical trials on medicinal products for human use, and repealing Directive 2001/20/EC	Come into force 2014. Regulation harmonises the assessment and supervision processes for clinical trials throughout EU, via a Clinical Trials Information System (CTIS). CTIS is planned to go live by 31 January 2022. A final go-live date will be six months after the EC confirms full functioning of CTIS through an independent audit.	https://ec.europa.eu/health/sites/health/files/files/eudralex/vol1/reg_2014_536/reg_2014_536_en.pdf

Table 3 Table of regulations relevant to use of medical device

The most important classification of medical devices is covered by the European Directive 93/42/EEC ('the Directive'). It outlines the process for classifying medical devices and explains how to seek clarification on classification of a medical device.

The Directive define a series of rules which can be used to classify a medical device. It is the responsibility of the manufacturer to classify a medical device that they intend to place on the market. The classification of a device determines the applicable route(s) for establishing conformity with the Directive.

General medical devices and related accessories are classified into one of four classes, which are based on the perceived risk of the device to the patient or user. The classification of a device determines the conformity assessment options that are applicable to the device, with higher risk devices undergoing higher levels of assessment. Classes range from I to III, corresponding to posing low to high risk.

The Directive classifies devices according to:

- **Degree of invasiveness**

A device, which in whole or in part, penetrates inside the body either through a body orifice or through the skin surface, is invasive. Invasiveness is generally categorised as invasive of a body orifice (including the surface of the eye), surgically invasive devices and implantable devices.

An implantable device is one which is intended to be totally introduced into the human body or to replace an epithelial surface or the surface of the eye by surgical intervention and which is intended to remain in place after the procedure.

Any device intended to be partially introduced into the human body through surgical intervention and intended to remain in place after the procedure for at least 30 days is also considered an implantable device.

- **Whether or not the device is active**

A medical device is considered to be active if the operation depends on a source of electrical energy or any source of power other than that directly generated by the human body or gravity and which acts by converting this energy. Medical devices intended solely to transmit energy between an active medical device and the patient where there is no significant change in the energy (e.g. nature, density, level) are not considered to be active medical devices.

The concept 'act by converting energy' includes conversion of energy in the device and/or conversion at the interface between the device and the tissues or in the tissues.

- **Part of the body affected**

The anatomy affected by the use of the device must be considered. Devices in contact with the central nervous system or the central circulatory system are automatically placed in a higher risk category.

Recommendations from WP8 regarding use of non-CE marked devices in SHAPES:

1. Review every POTENTIAL medical device in your use case, including software.
2. Confirm if the medical devices are CE marked
3. Confirm if any partners want to use information collected during the pilot to inform future certification or commercialisation of these medical devices
4. Identify the Sponsor and have a conversation about indemnity arrangements
5. Identify the competent authority and confirm what applications are required

The SHAPE architecture aims to support CE marked devices as standard, implying those compliant with common standards governing communication of medical

information and connectivity, with respect to specific medical standards that were mentioned in earlier sub-sections 4.1.1 and 4.1.2.

4.1.4 Cyber Security good practices for digital solutions

SHAPE architecture and digital solutions will be developed with security on mind since its design and following good practices on cybersecurity. In this sense, as core guideline, the consortium's digital solutions involving consumer Internet of Things will follow ETSI technical standard 303645 "Cyber Security for Consumer Internet of Things: Baseline Requirements". This standard provides up to 63 provisions extracted from risk assessment and threat modelling on IoT devices and grouped on thirteen good practices. As a summary, the good practices are:

- No universal default passwords in the initial setup.
- Implement an easy means to manage reports of vulnerabilities from users.
- Keep software updated, especially regarding security issues.
- Securely store sensitive/security parameters using state of the art ciphering storage mechanisms
- Communicate securely using best practice cryptography mechanisms
- Minimize exposed attack hardware (e.g. USB port) and software (e.g. open ports) surfaces
- Ensure software integrity using secure boot mechanisms.
- Ensure that personal data is secure and data transiting between a device and a service, especially associated services, should be protected.
- Make systems resilient to outages, especially to power and data network cuts.
- Examine system telemetry data regarding device and/or service use anomalies (e.g. unusual number of login attempts)
- Make it easy for users to delete user data
- Make installation and maintenance of devices easy
- Validate input data to detect unusual values which could indicate software vulnerabilities

Such good practices have been translated into requirements of the platform. Certainly, these provisions do not exclude any other consideration about security coming from specific domain and/or specific device considerations.

5 Semantic Interoperability for IoT & medical data

SHAPES aims to deliver a scalable, standardized and interoperable technological platform (TP), which will enable the integration of various digital solutions in the healthcare domain. As described in D5.2, digital solutions in SHAPES range from assistive robots to eHealth wearables and IoT devices. In order for these solutions to interoperate, they need to ensure a common level of understanding among each other. This is where semantics comes into play.

In the following sections, we introduce the concept of semantic interoperability, both on the Internet of Things (IoT) and the healthcare (medical) domain and explore ways in which we can ensure interoperable operation of both worlds.

5.1 Introduction to semantics and interoperability

Semantics concerns the study of meaning and is defined as an agreement on the common meanings of an information. Semantics is extremely important in our daily life and communication. For example, the single word “plant” can be interpreted in two different ways, either as a living organism or a working unit of a factory, depending on the corresponding context. Therefore, without semantics we would not be able to understand the meaning of the exchanged information, or at least we could not be sure that we are having the same understanding of it as our communication partner.

Interoperability refers to the ability of devices to successfully exchange data and perceive its intended meanings. We can divide the interoperability into three stages: technical interoperability, syntactic interoperability, and semantic interoperability. Technical interoperability deals with the actual transfer of digital data (bits transferred over wired or wireless medium). Syntactic interoperability makes sure that devices use a common language (symbols & concepts). Finally, semantic interoperability confirms that the devices have a common agreement on meanings of the syntax and symbols.

In SHAPES, there is a distinction between devices and digital solutions that collect IoT data, and solutions that deal with and exchange medical data, such as electronic health records (EHRs), medical history, questionnaires etc. Each digital solution employs different interoperability standards to ensure seamless exchange and interoperation of data, depending of course on the nature of these data.

In the following, subchapters we discuss and make the distinction between semantic interoperability in IoT and medical sectors and introduce interoperability standards that apply in SHAPES.

5.2 Semantic Interoperability in IoT

Internet of Things (IoT) is defined as a network of interconnected devices (either physical or virtual) that communicate without human intervention. Due to its rapid growth, the Internet of Things has a huge impact in our daily lives. Most of the devices we interact with are either already part of the IoT ecosystem or to join IoT. The current IoT landscape involves many vertical IoT silos that are rarely interconnected.

As IoT focuses on machine-to-machine communication without human intervention, a mandatory requirement is to have seamless interoperability between devices. Thus,

the devices need a mechanism to understand the context of the information to be able to perceive the intended meanings of it. Both technical and semantic interoperability levels are necessary to enable interoperability across multiple IoT platforms. Specifically, semantic interoperability is the key to data exchange and service creation across large vertical applications and seen as the next step of evolution of the IoT.

To promote interoperability, several standard developing organizations (SDO), such as the W3C¹⁴, oneM2M¹⁵, ETSI¹⁶ and OGC¹⁷, exist. These SDOs implement a series of standardization strategies to enable interoperability in machine-to-machine and IoT. Namely, the W3C (World Wide Web) develops open standards and ontologies, such as the Semantic Sensor Network ontology (SSN)¹⁸ to ensure the long-term growth of the Web of Things. The Open Geospatial Consortium (OGC) implements standards, like the SensorThings API¹⁹, which are based on existing SWE²⁰ standards and are specifically suitable for IoT. Also, the OGC and the W3C consortia have jointly developed the Sensor, Observation, Sample, and Actuator (SOSA) ontology that describe sensors and actuators, their observations, actuation, and sampling activities.

To ensure interoperability, SHAPES exploits the SymbloTe mediation framework [1], which in turn is based on several IoT standards that ensure seamless interoperation across diverse IoT devices and/or platforms. The symbloTe is based on the SSN and SOSA ontologies and also “borrowed” concepts from the SensorThings API standard to define its data model (information model). Hence, we can claim that SHAPES integrates these interoperability standards in retrospect. Additionally, SHAPES takes into account EU standards for IoT-based platforms and the exchange of IoT-based data, namely those deriving from the FIWARE initiative²¹. Specifically, the FINoT platform, developed by FINT Ltd., is a FIWARE-based IoT platform able to interconnect sensors, actuators and data loggers. Such an approach implies that Digital Solution need to adopt symbloTe mechanisms in order to be able to exchange their IoT data, and in particular medical IoT data.

Briefly, FIWARE is an open-source initiative defining a universal set of standards for context data management. Current FIWARE NGSI (Next Generation Service Interface)²² specifications are public and align with the NGSI-LD (Linked-Data)²³ specifications published by ETSI, and support the receiving, processing, contextualising, and publishing of IoT data. The FIWARE NGSI v2 information model has been evolved to better support linked data (entity relationships), property graphs and semantics (exploiting the capabilities offered by JSON-LD). The main constructs of NGSI-LD are: Entity, Property and Relationship. NGSI-LD Entities (instances) can be the subject of Properties or Relationships. In terms of the traditional NGSI v2 data

¹⁴ <https://www.w3.org>

¹⁵ <https://www.onem2m.org>

¹⁶ <https://www.etsi.org>

¹⁷ <https://www.ogc.org>

¹⁸ <https://www.w3.org/TR/vocab-ssn/>

¹⁹ <https://www.ogc.org/standards/sensorthings>

²⁰ SWE standard: <https://www.ogc.org/projects/groups/sensorwebdwg>

²¹ <https://www.fiware.org>

²² <https://www.fiware.org/2016/06/08/fiware-ngsi-version-2-release-candidate/>

²³ https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_howto/index.html

model, Properties can be seen as the combination of an attribute and its value. Relationships allow to establish associations between instances using linked data. In practice, they are conveyed by means of a special NGSI v2 attribute with a special value (relationship's object), which happens to be a URI which points to another entity. NGSI-LD Entities are represented using JSON-LD, a JSON-based serialization format for Linked Data.

5.3 Semantic interoperability in the medical domain

In the healthcare industry, interoperability is the ability of different information systems and software applications to communicate and exchange data, and use the information exchanged. The use of standards and data exchange models enables information to be shared between healthcare providers, professionals, patients, hospitals, etc., regardless of the application being used. The healthcare industry has developed and appropriated standards for various purposes related to messaging, terminology, documents, conceptual frameworks, application and architectures, both for syntactic interoperability, based on the structure of communication, and for semantic interoperability, which refers to the meaning of the communication.

For this reason, several organizations, such as HL7 International²⁴, Continua Health Alliance (CHA)²⁵, IHE (integrating the Health Enterprise)²⁶, are seeking to unify interoperability criteria. A plethora of standards have been developed, either to facilitate messaging in healthcare (e.g., the HL7 V2.X and HL7 V3²⁷, the IEEE 1073²⁸), or terminologies and common vocabularies, thus adding to the semantic component (e.g., the ICD-10²⁹, LOINC³⁰ or SNOMED³¹).

To ensure semantic interoperability between the e-health/medical Digital Solutions in SHAPES, the HL7 FHIR (Fast Healthcare Interoperability Resources)³² and the open MHealth standards will be considered. In the following sections, a small introduction to both standards is given. Please note that this is only an introductory description and more detailed ones will follow as the SHAPES project evolves and the data models that will ensure interoperability among solutions will be developed.

5.4 Technologies in semantic interoperability

In this section, we will discuss the corner-stone technologies for semantics.

²⁴ <https://www.hl7.org>

²⁵ <https://www.pchalliance.org/about-continua>

²⁶ <https://www.ihe.net/>

²⁷ http://www.hl7.org/implement/standards/product_brief.cfm?product_id=186

²⁸ <https://ieeexplore.ieee.org/document/713466>

²⁹ <https://en.wikipedia.org/wiki/ICD-10>

³⁰ <https://loinc.org>

³¹ <https://www.snomed.org/snomed-ct/software-tools>

³² <http://hl7.org/fhir/2016may/protocol.html>

- **RDF:** Resource Description Framework (RDF)³³ is the standard data representation model on the semantic web. RDF represents data as RDF statements. RDF statements are written as a triple in the form of: “subject – predicate – object”.

Data in RDF is often depicted as a labelled, directed graph (Figure 4) where the nodes represent *resources* (depicted as ovals) or *literals* (depicted as rectangles) and the labelled edges represent *relations* (often also called *predicates*).

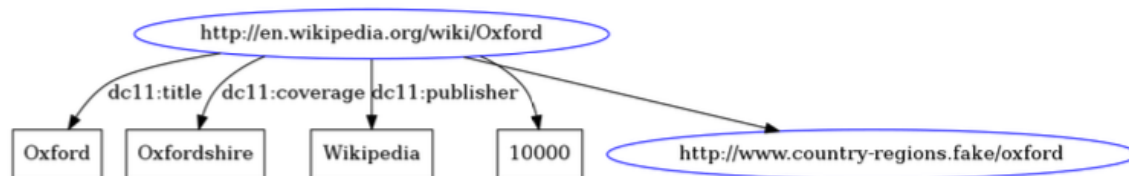


Figure 4: Example RDF data as a labelled graph

The subject in a triple denotes the resource usually specified by a URI. The object denotes another resource or entity. The predicate describes the relationship between the subject and the object e.g. title is Oxford. Datatypes of literals can be defined using XSD datatypes³⁴. The use of URIs to identify resources allows globally unique addressing even between different databases and thus, allows global interlinking of information.

RDF is an abstract (metadata) data model, which means there are multiple serialization formats that can be used to represent RDF data. The most popular are RDF/XML³⁵, N-Triples³⁶, Turtle³⁷, RDFa³⁸, Notation3 (N3)³⁹ and JSON-LD⁴⁰.

- **RDF Schema**⁴¹ is the vocabulary for RDF data. It is an extension to the RDF data model and provides semantics to RDF data. The RDF Schema expresses the relation between resources by grouping together related RDF resources. The RDF Schema⁴² defines the concept of *classes*, *property*, *domain* and *range* (Table 4 and Table 5). Resource groups in RDF Schema are called *classes*. Property is equivalent of a predicate; it defines a relation between subject and object. Domain and range are the additional constraints to provide semantic meaning to data.

³³ W. W. W. Consortium (W3C), “Rdf 1.1 concepts and abstract syntax,” W3C Recommendation, 2014, <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>

³⁴ XSD datatypes: <https://www.w3.org/TR/xmlschema-2/>

³⁵ RDF/XML format: <https://www.w3.org/TR/rdf-syntax-grammar/>

³⁶ N-Triples format: <https://www.w3.org/TR/n-triples/>

³⁷ Turtle format: <https://www.w3.org/TR/turtle/>

³⁸ RDFa format: <https://www.w3.org/TR/rdfa-primer/>

³⁹ Notation3 (N3) format: <https://www.w3.org/TeamSubmission/n3/>

⁴⁰ JSON-LD format: <https://www.w3.org/TR/json-ld/>

⁴¹ “Rdf schema 1.1,” W3C Recommendation, 2014, <https://www.w3.org/TR/rdf-schema/>

⁴² RDF Schema: <https://www.w3.org/TR/rdf-schema/>

Class Name	Description
rdfs:Resource	all things declared by RDF are resources
rdfs:Class	describes the concepts of a class
rdfs:Literal	describes the concept of a literal
rdfs:Property	the class for properties

Table 4 Most important classes of RDFS

Property Name	Description
rdfs:domain	defines to which subjects does a property applies
rdfs:range	defines the set of values a property can accept
rdf:type	used to state that a resource is an instance of a class
rdfs:subClass Of	defines one class as a subclass of another class
rdfs:label	provides a human-readable version of resource's name
rdfs:comment	provides a human-readable description of resource
rdfs:seeAlso	link to another resource that might provide additional information

Table 5 Most important properties defined in RDFS

- **Web Ontology Language (OWL)** [11] is semantic web language designed to represent complex knowledge and relationships about things. Documents of OWL are referred as Ontologies. OWL statements are written using RDF statements⁴³. The commonly used representation language for RDF is XML. RDF also support JSON as an alternative to the XML.

5.5 Information models for semantic interoperability

The best way to achieve semantic interoperability is through a common interpretation of data and information based on a shared ontology. A detailed description of how SHAPES will tackle semantic interoperability and information model that is based upon is given in the sections below. Again, we make the distinction between information/data models designed for IoT-based solutions and information models for medical-related ones. The FHIR resource-based descriptions is given in Section 6.2.4.

Note that that data models and semantic interoperability analysis is part of Task 4.2 which is an ongoing task running until Month 24 of the project. Therefore, what is presented below is a preliminary analysis of the information models and should be regarded as such.

5.5.1 SHAPES Information Model for IoT data

To achieve semantic interoperability between vertical IoT platforms, SHAPES utilizes the symbloTe⁴⁴ mediation framework [12] and extends its Core Information Model (CIM) to accommodate sensor descriptions and other devices that exist in the SHAPES Digital Solutions Ecosystem. In this section, we provide a detailed

⁴³ <https://www.w3.org/TR/owl2-overview/>

⁴⁴ H2020 symbloTe project, Available Online: <https://www.symbiote-h2020.eu>

description of the CIM and insights into how the CIM will be extended for the needs of SHAPES.

As described before, to enable interoperability between two platforms (e.g., platform A and B in Figure 5), there needs to be a mutual understanding of things, i.e., some unification of their internal information models must somehow be defined. In general, an information model is defined as the conceptual modelling of managed objects independent of any implementation or underlying protocol (that is, it describes the relationships between objects).

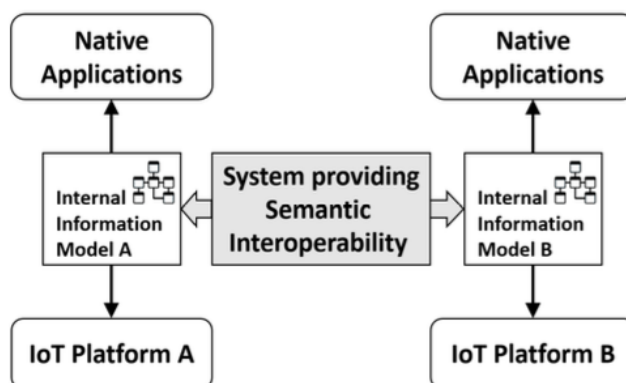


Figure 5: Schematic of semantic interoperability between different IoT platforms

The information model used in SymbloTe has been designed to be as abstract as possible but at the same time as detailed as needed. This is achieved by having a common, minimalistic Core Information Model (CIM), covering resource descriptions that are exposed by vertical platforms. Each platform, however, can have its own Platform Information Model (PIM), which should be compliant to the CIM for them to share their resources using SymbloTe.

5.5.2 The symbloTe Core Information Model

To achieve semantic interoperability, symbloTe uses a Core Information Model (CIM) approach with Extensions [13]. In symbloTe, each platform can describe the resources it offers using its own information model called Platform-Specific Information Model (PIM). However, this model cannot be chosen completely freely by the platform but is an extension of symbloTe CIM and must comply with it so that minimal out-of-the-box interoperability with other symbloTe-enabled platforms is achieved.

The CIM is designed to be as explicit as needed to ensure a minimal level of out-of-the-box interoperability, but at the same time to be as minimalistic and abstract as possible giving platforms the freedom to adapt it to their needs. For example, the CIM defines abstract classes like Sensor, Actuator, Service or Location with minimal properties like id and name. When a platform needs a more sophisticated class hierarchy or additional properties it that can define them in its PIM.

The Core Information Model (CIM) is the central information model of symbloTe (Figure 3) and serves a two-fold purpose. First, it defines all the terms (classes and their relations) that symbloTe components can understand. The second purpose is

that it serves as a shared vocabulary between all symbloTe-compliant platforms and thereby enables out-of-the-box interoperability between them.

The central class of the model is *Resource*, which is also used to link a *Resource* belonging to a *Platform*. The *Resource* is further divided into *Service* and *Device*, which is even further divided into *Sensor*, which can be a *MobileSensor* or a *StationarySensor*, and *Actuator*. For better overview, classes and relations belonging exclusively to one of these domains are highlighted using the same shade of green. Elements defined in other, already existing ontologies are highlighted in light grey. Three major domains were described in symbloTe CIM:

- The sensor domain is strongly influenced by the SSN ontology, as well as the SOSA ontology. Therefore, the basic classes and their relations of the SSN ontology can be found in the CIM: *Sensor*, *Observation*, *ObservationValue*, *Property*, *FeatureOfInterest* and *UnitOfMeasurement*. As the SSN ontology does not allow to define the capabilities of a sensor without having existing observations from that sensor, a relation between *Sensor* and *Property* was added as well as one between *StationarySensor* and *FeatureOfInterest* to support definition of properties (and feature of interest) a sensor can observe. For modelling observation times, the W3C Time Ontology was used.
- The actuator domain covers the actuating part and is based on the combination of two actuator models. The *FeatureOfInterest* and *Property* classes are used as a link between the sensing and the actuator domain.
- The third major domain is the service domain. In the CIM diagram, it only consists of a single class, the *Service* class, and four outgoing relations.

Besides the three major domains, there are even more classes and relations that belong to some crosscutting (sub-) domains, i.e., they are being used in a subset of the three major domains. They can roughly be categorised into the domains datatypes, parameters and location. The datatype domain allows definition of primitive and complex datatypes, which can be used as datatype for input and output parameters. The parameter domain allows definition of input parameters as well as fine-grained restrictions to those parameters. Both are use in the service, as well as the actuation domain.

The location domain allows definition of locations in three different types: by well-known text (*WKTLocation*), GPS position (*WGS84Location*) and as a symbolic location (*SymbolicLocation*).



Apart from the CIM, SymbloTe also defines Best Practice Information Models (BIM), which are a special kind of PIM provided together with symbloTe, and are designed to cover different domains. For the needs of SHAPES, the BIM from the symbloTe Smart Residence use case will be used as a basis for the extension of the information model towards the needs of SHAPES. Briefly, symbloTe Smart Residence BIM has been designed for the needs of the use case to support interoperability across different smart home IoT solutions through a generalised abstract model that describes interconnected objects. The symbloTe Smart Residence use case has demonstrated how concepts of collecting health information (e.g., weight, blood pressure) can be embedded into a smart home. Thus, the Smart Residence BIM fits best to the needs of the SHAPES project to accommodate not only general IoT device requirements covered by CIM, but also requirements more specific to smart medical devices which monitor various health parameters such as body weight and blood pressure.

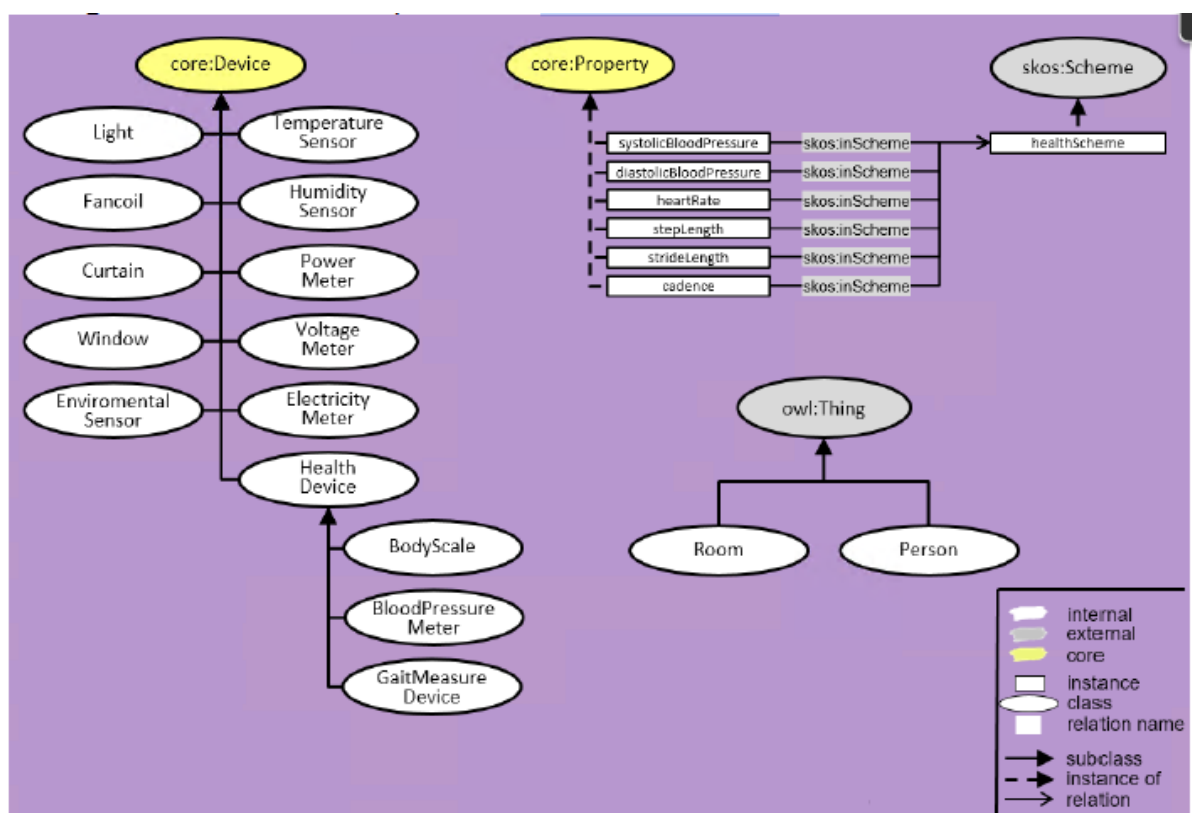


Figure 7: Devices and properties part of the Smart Residence domain model of the BIM

The Figure 7 presents the device classification used as well as properties used in the health and ambient-assisted-living subdomain. Additional devices are defined in the Smart Residence BIM, as shown in Figure 8. Following the concepts of the CIM, this BIM also defines a *Person* providing several health-related properties, namely blood pressure (*systolicBloodPressure* and *diastolicBloodPressure*), the heart rate as well as gait measurements (*stepLength*, *strideLength* and *cadence*). These properties are observed by *HealthDevices* (a *BodyScale*, *BloodPressureMeter* and a *GaitMeasureDevice*).

5.5.3 Forming the SHAPES Core Information Model

The information model used in symbloTe will be modified to deal with smart medical devices and their data by extending the devices and observations in symbloTe Smart Residence BIM to cover the needs of SHAPES project. Note that here we are only interested in including devices that deal with IoT data in SHAPES, while medical data such as questionnaires will either conform to the FHIR or translated into FHIR resources, as explained in the next section.

As an initial step towards the formation of the SHAPES core information model, we investigated similarities and differences in the different definitions of the symbloTe CIM to resource definitions found in FHIR, with the scope of exploring how to extend the current symbloTe information model to deal with medical-related data. On that note, a comparison of the main constructional elements is presented below:

- **Resource.** The *Resource* is the most generic resource description and the central class of the SymbloTe CIM (Figure 6). The properties of the Resource class include: an identifier (id), a name, description and an *interworkingServiceURL* (Figure 8). In FHIR, resource is the base class for all other definitions, both clinical and administrative. A FHIR resource is comprised of: an ID, meta-data about the resource, a base language and a reference to “Implicit Rules” (Figure 9).

```
{
  "@c": ".StationarySensor",
  "name": "Stationary 1",
  "description": [
    "This is stationary 1"
  ],
  "interworkingServiceURL": https://www.example.com/Test1Platform
}
```

Figure 8: JSON description example of a Resource in SymbloTe CIM

```
{
  "resourceType" : "[name]",
  "id" : "<id>", // Logical id of this artifact
  "meta" : { Meta }, // Metadata about the resource
  "implicitRules" : "<uri>", // A set of rules under which this content was created
  "language" : "<code>" // Language of the resource content
}
```

Figure 9: JSON description example of the resource base definition in FHIR

The two Resource definitions are matched only on the id and on the interworkingServiceURL, in the following way. The symbloTe id is a string-type property that uniquely identifies each resource, and the interworkingServiceURL is the resource’s URL address. The ‘id’ in the FHIR resource definition is either a “Logical ID” that is used as a basis for a “Location” URL (e.g., in <http://test.fhir.org/rest/Patient/123>: 123 is the Logical Id of a Patient resource) or a canonical URL that identifies the resource (like the interworkingServiceURL).

Implicit Rules are only defined in FHIR and provide a reference to a custom agreement that describes how the resource is being used⁴⁵. In general, the 'Implicit Rules' field is rarely used. Every resource has, also, an optional language element in FHIR, which is the base language for the content of the resource (e.g., English, Russian, Japanese, etc.). More information on the 'Language' field can be found here: <https://www.hl7.org/fhir/languages.html>.

Finally, the 'meta' field is comprised of a series of optional sub-fields which are presented in Figure 10. In symbloTe information model, there is no field containing similar information to the 'meta' field.

```
{
  // from Element: extension
  "versionId" : "<id>", // Version specific identifier
  "lastUpdated" : "<instant>", // When the resource version last changed
  "source" : "<uri>", // Identifies where the resource comes from
  "profile" : [{ canonical(StructureDefinition) }], // Profiles this resource claims to conform to
  "security" : [{ Coding }], // Security Labels applied to this resource
  "tag" : [{ Coding }] // Tags applied to this resource
}
```

Figure 10: JSON example for the 'meta' field of the FHIR resource definition

The symbloTe knows different types of resources, which are modelled as subclasses of Resource in the CIM. They are divided into Service and Device, which is further divided into Sensor, which can be a *MobileSensor* or a *StationarySensor*, and Actuator. We further investigated, compared, and contrasted the Device and the Sensor classes, as these will be the basis for the description of devices in SHAPES.

- **Device.** The Device class in symbloTe models IoT devices and has the Location and Services defined as properties (Figure 6). The *Location* in symbloTe is defined in three different types: by well-known text (*WKTLocation*), GPS position (*WGS84Location*) and as a symbolic location (*SymbolicLocation*). An example of a Location definition, as a *WGS84Location*, is shown in Figure 11.

```
"locatedAt": {
  "@c": ".WGS84Location",
  "longitude": 5.349014,
  "latitude": 25.864716,
  "altitude": 35,
  "name": "Somelocation",
  "description": [
    "Secret location"
  ]
},
```

Figure 11: snapshot of an example for the 'Location' field in symbloTe Device definition

⁴⁵ <https://www.hl7.org/fhir/profiling.html#glossary>

The field Services corresponds to Service class objects as depicted in Figure 3 and is, for now, not of much interest in forming the SHAPES information model. However, for the sake of completeness, we present an example (instance) of a Service in Figure 12.

```
{
  "@c": ".Service",
  "name": "Service 1",
  "description": [
    "This is service 1"
  ],
  "interworkingServiceURL": "https://www.example.com/Test1Platform",
  "name": "service1Name",
  "resultType": {
    "@c": ".RdfsDatatype",
    "array": false,
    "isArray": false,
    "datatypeName": "http://www.w3.org/2001/XMLSchema#string"
  },
  "parameters": [
    {
      "name": "inputParam1",
      "mandatory": true,
      "restrictions": [
        {
          "@c": ".RangeRestriction",
          "min": 2,
          "max": 10
        }
      ],
      "datatype": {
        "@c": ".PrimitiveDatatype",
        "isArray": false,
        "baseDatatype": "http://www.w3.org/2001/XMLSchema#string"
      }
    }
  ]
}
```

Figure 12: JSON example for the Service class in *symboloTe*

In FHIR, the Device resource can describe a medical or a non-medical device and is an administrative resource that tracks individual instances of a device and their location. The FHIR Device resource has several properties associated with it, such as the id, the status of the device (active, inactive etc.), name and device type, location reference, a URL (network address) and a patient (to whom the device is affixed). A JSON example of the FHIR Device resource is presented in Figure 13.

```
{
  "resourceType": "Device",
  "id": "2043305",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2021-04-19T05:30:48.382+00:00",
    "source": "#ZWfx7Cnv2FWElhzq"
  },
  "identifier": [ {
    "value": "00868475000235-34d8b55d-2e5d-4782-b2fc-760a35b51205"
  } ],
  "patient": {
    "reference": "Patient/2043304"
  }
}
```

Figure 13: JSON example for the Device FHIR resource

Both Device definitions have a description of location, albeit in different conceptual constructs. The definition of location in FHIR is given as a different resource construct, with its own sub-fields⁴⁶. The ones matching symbloTe definition of *Location* are the ‘name’ (i.e. the name of the location, for example ‘Evanston Hospital’), ‘address’ (i.e. the physical address of the specified location type) and ‘position’ (WGS84-type coordinates) sub-fields. An example of the FHIR Location resource is given below (Figure 11). A possible extension for the SHAPES CIM would be to include information about the organization responsible for provisioning and upkeep (i.e., the “managingOrganization” or the “part of” field of the FHIR Location) with possible instance-types relating to a hospital, special care unit, doctor’s office, home, and other sub-locations (such as room) related to the aforementioned ones.

```
{
  "resource": {
    "resourceType": "Location",
    "id": "46506c48-7815-411f-9164-2071681b20d5",
    "meta": {
      "versionId": "1",
      "lastUpdated": "2020-03-24T21:54:57.173+00:00",
      "source": "#8UXiRDk70NZenKVP",
      "profile": [ "http://hl7.org/fhir/us/core/StructureDefinition/us-core-location" ],
      "tag": [ {
        "system": "https://smarthealthit.org/tags",
        "code": "Covid19 synthetic population from Synthea"
      } ]
    },
    "status": "active",
    "name": "EVANSTON HOSPITAL",
    "telecom": [ {
      "system": "phone",
      "value": "8474328000"
    } ],
    "address": {
      "line": [ "2650 RIDGE AVE" ],
      "city": "EVANSTON",
      "state": "IL",
      "postalCode": "60201",
      "country": "US"
    },
    "position": {
      "longitude": -87.694352,
      "latitude": 42.046391
    },
    "managingOrganization": {
      "reference": "Organization/b1ad1b56-50a6-3cf0-b7b1-6a1dfb50a76c",
      "display": "EVANSTON HOSPITAL"
    }
  }
}
```

Figure 14: JSON example of the Location FHIR resource

⁴⁶ <https://www.hl7.org/fhir/location.html>

- **Sensor.** The Sensor class is one of the basic classes in symbloTe CIM and has the 'observesProperty' field as sub-type. It is further divided into the *MobileSensor*, which has no further attributes attached to it and the *StationarySensor*, which is further linked with the *FeatureOfInterest* class. The *FeatureOfInterest* class is comprised by the 'name', 'description' and 'hasProperty' subfields, and it roughly relates to a more detailed and specific description of the location the sensor (and/or the observation related to the sensor) is found.

A JSON example of a *StationarySensor* is given in Figure 15.

```
{
  "@c": ".StationarySensor",
  "name": "Stationary 1",
  "description": [
    "This is stationary 1"
  ],
  "interworkingServiceURL": "https://www.example.com/TestPlatform",
  "locatedAt": {
    "@c": ".WGS84Location",
    "longitude": 5.349014,
    "latitude": 25.864716,
    "altitude": 35,
    "name": "SomeLocation",
    "description": [
      "Secret location"
    ]
  },
  "featureOfInterest": {
    "name": "Room1",
    "description": [
      "This is room 1"
    ],
    "hasProperty": [
      "temperature"
    ]
  },
  "observesProperty": [
    "temperature",
    "humidity"
  ]
}
```

Figure 15: JSON example of a *StationarySensor* in symbloTe CIM

There is no direct correspondence of a sensor resource in FHIR; the closest one is the Device resource, described previously. Based on the descriptions provided, we could argue that all information presented in a device resource in FHIR is found in the sensor description of the symbloTe CIM, with the exception for the Location field, a possible extension of which was described above. Furthermore, the *Sensor* class is linked to the *Observation* class via the *hasObservation* property.

- **Observation.** The *Observation* class in symbloTe CIM is comprised of following properties: *id*, *Location*, *resultTime*, *samplingTime* & *observationValue* (Figure 16).

```

    "resourceId": "5ab5db974a234e717380721f",
    "location": {
      "longitude": 150.89,
      "latitude": 23.56,
      "altitude": 343.74
    },
    "resultTime": "2017-05-17T10:35:50",
    "samplingTime": "2017-05-17T10:35:50",
    "obsValues": [
      {
        "value": 21,
        "uom": {
          "symbol": "°C",
          "label": "Celsius",
          "comment": "Celsius degrees" },
        "obsProperty": {
          "label": "temperature",
          "comment": "temperature in degrees"
        }
      }
    ]
  }

```

Figure 16: JSON example of an Observation in *symbloTe*

In FHIR, the Observation resource mostly concerns simple name/value pair assertions with some metadata. Observation is intended for capturing measurements and subjective point-in-time assessments.

```

    "resourceType": "Observation",
    "id": "51670",
    "meta": {
      "versionId": "1",
      "lastUpdated": "2019-10-28T19:53:52.339+00:00",
      "source": "#AtkBjQwnx6GdpwvA"
    },
    "status": "final",
    "code": {
      "text": "averageRespiratoryRate"
    },
    "subject": {
      "reference": "Patient/51662"
    },
    "effectivePeriod": {
      "start": "2019-10-28T19:53:52+00:00",
      "end": "2019-10-28T19:54:02+00:00"
    },
    "issued": "2019-10-28T19:53:52.246+00:00",
    "valueQuantity": {
      "value": 2.0,
      "unit": "RBpm"
    }
  }

```

Figure 17: JSON example of an Observation in *symbloTe*

An example of the FHIR Observation resource is found in Figure 17. The Observation resource might include information about vital signs (e.g., body weight), laboratory measurements and/or device measurements. Observation includes several (optional) properties (subtypes). The ones that directly map into the properties found in the *symbloTe* Observation class are the id, the 'issued' property (Date/Time this version was made available, which is matched to the

‘resultTime’ property in symbloTe CIM), the effectiveDateTime (time that the observation is most relevant as an observation of the subject, which is matched to the “samplingTime” of symbloTe CIM) and the Value property (which will investigate further below). The Location property (found in the symbloTe Observation description) can be referenced by the FHIR observation description⁴⁷.

- **ObservationValue.** The *ObservationValue* class in symbloTe is comprised of the value property, the observed property (*observesProperty*), the FeatureOfInterest and the *UnitOfMeasurement* property. The *UnitOfMeasurement* has, in turn, the following properties, a symbol, name, an iri and a description. A detailed JSON example of an *ObservationValue* instance is given in Figure 18.

```
"obsValues": [
  {
    "value": 21,
    "uom": {
      "symbol": "°C",
      "label": "Celsius",
      "comment": "Celsius degrees" },
    "obsProperty": {
      "label": "temperature",
      "comment": "temperature in degrees"
    }
  }
]
```

Figure 18: JSON example of an *ObservationValue* object class in symbloTe

The corresponding amount of information is captured in the “value” property of the FHIR Observation resource and more specifically in the “valueQuantity” one, as shown in the JSON example (Figure 19)⁴⁸.

```
// value[x]: Actual result. One of these 11:
"valueQuantity" : { Quantity },
"valueCodeableConcept" : { CodeableConcept },
"valueString" : "<string>",
"valueBoolean" : <boolean>,
"valueRange" : { Range },
"valueRatio" : { Ratio },
"valueSampledData" : { SampledData },
"valueAttachment" : { Attachment },
"valueTime" : "<time>",
"valueDateTime" : "<dateTime>",
"valuePeriod" : { Period },

"valueQuantity": {
  "value": 107,
  "unit": "mmHg",
  "system": "http://unitsofmeasure.org",
  "code": "mm[Hg]"
}
```

Figure 19: The “valueQuantity” property on an Observation FHIR resource example

⁴⁷ <https://www.hl7.org/fhir/location.html>

⁴⁸ <https://www.hl7.org/fhir/observation-example-bloodpressure.json.html>

- **Person.** There is no definition for a person in symbloTe CIM but there is one in the Smart Residence BIM (Figure 7), which is defined as an OWL class. In FHIR, on the other hand, there is a Person definition that provides a set of common demographics for an individual (name, gender, address etc.). Person in FHIR is independent of a health-related context. On the other hand, Patient resource concerns demographics and information about an individual receiving care or health-related services (see Section 5.1), with an example Person resource at: <http://www.hl7.org/implementation/standards/fhir/person-example.json.html>.

Based on the comparisons presented above, we postulated that the Person class should be extended to link information between the person and an Observation about him/her, coming from a (health) device and/or sensor. Therefore, a user identification number should be incorporated in the Person class of the Smart Residence BIM. Additional additions may be required for the Observation and *ObservationValue* classes, but these will be decided as the project evolves.

Additionally, SHAPES includes a large variety of sensors and/or devices that deal with IoT data. The Table 6 below shows IoT devices, and their monitoring parameters foreseen to be used in SHAPES, along with the data types they will be recording.

Device Name	Outcome Variable	Digital Solutions/partner bringing the solution
Movement sensor	'SGID' score start_pos end_pos 'number of steps' 'target reached' 'react Time' 'goal time' 'distance' 'level' 'speed' 'average_left_step_height' 'average_right_step_height'	LLM CARE Healthcare System for Cognitive and Physical training (AUTH)
Xiaomi Band 3 (fitness tracker)	step_count heart_rate	eCare (EDGE). Follows openmhealth.org
OMRON M7 Intel	systolic_blood_pressure diastolic_blood_pressure heart_rate	eCare (EDGE). Follows openmhealth.org

OMRON VIVA	body_weight body_mass_index body_visceral_fat body_fat_percentage body_skeletal_muscle basal_metabolic_rate	eCare (EDGE). Follows openmhealth.org
blood glucose meter	blood glucose	eHealthPass (GNOMON)
weight scale	weight	eHealthPass (GNOMON)
blood pressure meter	blood pressure	ehealthPass (GNOMON)
Xiaomi smart Band 4	Steps Sleep Exercise heart rate	physical activity monitoring app
Mbientlab MetaMotionR Sensor	Fall	physical monitoring app
3D camera	Exercise routines completed number of repetitions time performance accuracy	Phyx.io (physical rehabilitation platform)
Weather Station	id' 'type' 'TimeInstant' 'airHumidity' 'airTemperature' 'averageWindSpeed' 'batteryVoltage' 'gustWindSpeed' 'location' 'rainRate' 'solarRadiation' 'todaysRain' 'windDirection'	FINoT (FINT)
Indoor Air Quality Sensor	id' 'type' 'TimeInstant' 'location' 'temperature' 'humidity' 'TVOC' 'CO2eq'	FINoT (FINT)

NOTiFY (with energymonitor)	'device_id' 'temperature' 'date_time' 'Power' 'Current' 'Voltage'	NOTiFY (OMN)
NOTiFY (Lampmonitor)	'device_id' 'date_time' 'onoffstate': 1.0	NOTiFY (OMN)

Table 6 IoT devices and data in SHAPES.

Please note that the data in the above table may be subject to change and/or additional devices or data types might be included in the future, as this work is evolving. Based on this table, we further extend the symbloTe BIM for Smart Residence to include the additional sensors that are found in SHAPES.

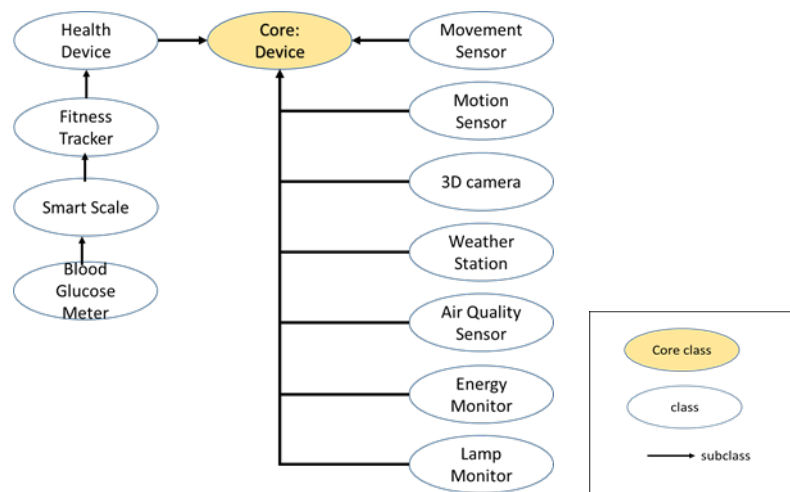


Figure 20: Devices in the SHAPES Information model

The Figure 20 presents the device classification in the new SHAPES information model. The properties classification will be added later in the process. A finalized version of the SHAPES CIM is expected to be included in the next WP4 deliverable, which is D4.3 due in M24.

6 SHAPES Architecture

The SHAPES Technological Platform (TP) brings a combination of devices, software, and accessible modes of interacting within the living environment that can adapt to the needs and priorities of older individuals, including those facing permanent or temporary reduced functionality and capabilities. Importantly, the Platform is aware of its environment and the users' physical and mental states, therefore being able to adapt to their pressing needs and concerns and use nudges to encourage users to make smart healthy and active decisions, as well as achieve improved levels of independence, active social participation and quality of life. The SHAPES TP drives interconnection and integration, physical activity, personal agency and in-person human and community relationships. In addition, the SHAPES TP brings effective and efficient mechanisms to augment social interactions between individuals and, where applicable and consented, caregivers and care and health service providers, with rich communications accessible to all (e.g., combined video, text and/or audio) and ancillary data (e.g., activity and vital signs) improving diagnosis, risk assessment and complications prediction to enable early intervention and avoid corrective actions.

As such, the SHAPES TP is an intelligent platform that processes in real-time the gathered user data and adapts itself as a suite of resources and capabilities to flexibly provide the human, digital and mechanical wherewithal to improve a wide variety of real-life environments. Powered by a big data and artificial intelligence (AI) engine, the Platform incorporates an understanding of the involving environment in which older individuals actually live and iteratively follows-up and evolves (i.e., learns) based on how the resources are being used.

6.1 Conceptual SHAPES Architecture

The generic SHAPES conceptual architecture shown in Figure 21 depicts its functions and corresponds to the end users view of its operation, thus providing intuitive view for its usability. In the background of the system lies the “Intelligent Living and Care Environment”, being a collections of services, devices, applications and other types of solutions that offer diverse e-Healthcare support to diverse classes of stakeholders for vast range of needs. As most of those are developed independently and not always compliant with common interoperability criteria, their capabilities to communicate and exchange data and information may often be handicapped.

This is where SHAPES architecture comes into play, offering interoperability mechanisms as well as additional capabilities (e.g. AI-based analytics) with advanced security mechanisms, thus enabling seamless integration of such originally dispersed and dis-connected services and applications into collections of tools for addressing similar needs and leveraging their individual advantages to mutual added benefit.

Through a SHAPES concept of a Marketplace such extended new types of solutions combining several Digital Solution can be advertised, discovered and consumed by stakeholders as well as combined together to address new and emerging e-Health needs, example being the current COVID outbreak that has identified a need for new types of e-Health solutions targeting large-scale infectious diseases.

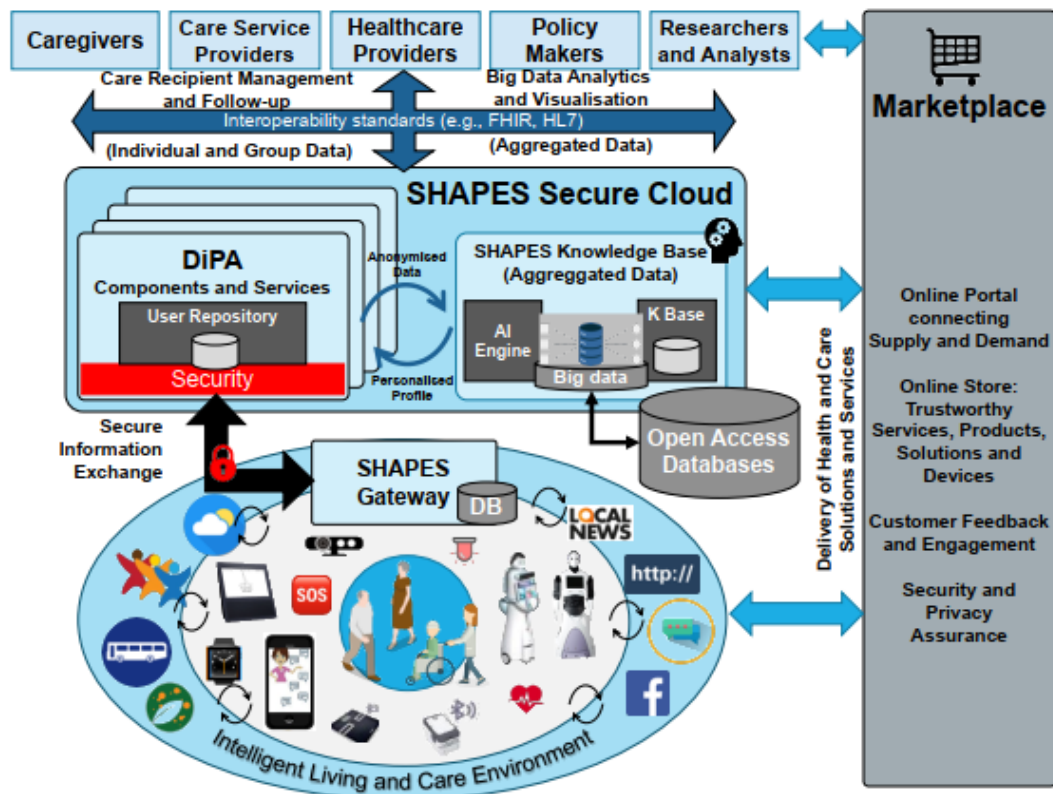


Figure 21: Generic view of the overall SHAPES Ecosystem

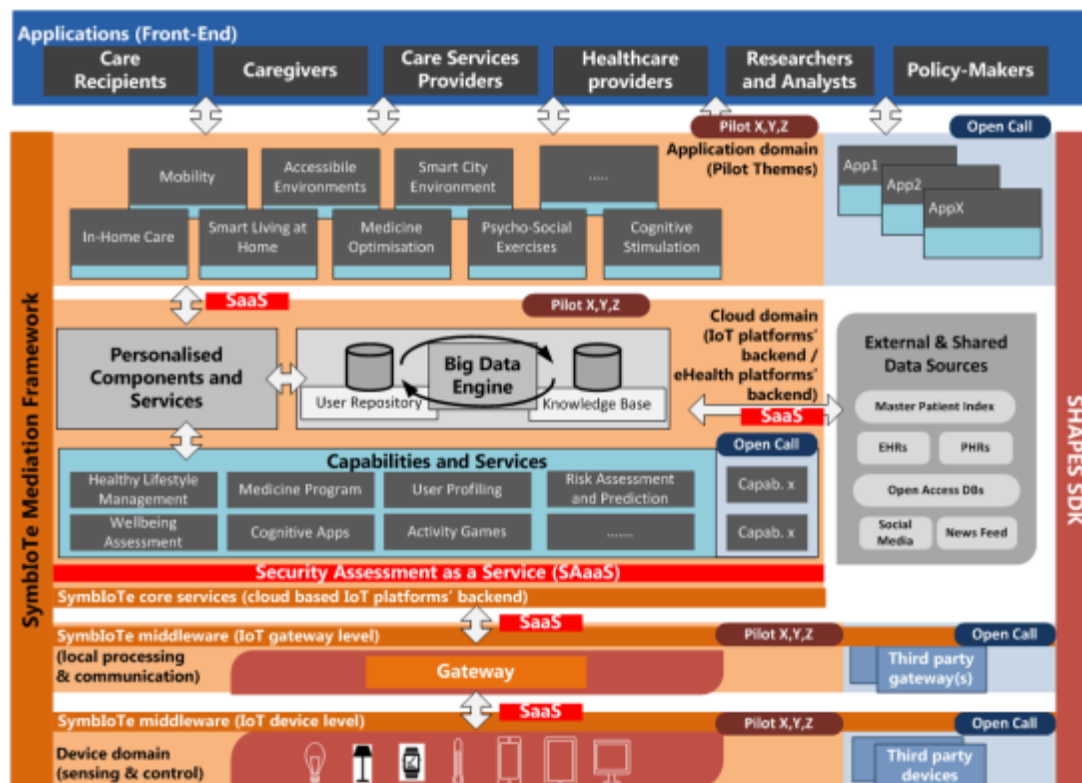


Figure 22: SHAPES system functionalities corresponding to users' needs as defined in WP2

To offer technological capabilities to address such generic user needs, a SHAPES technological platform has been proposed, as in Figure 22, based on which a practical implementation of the Technological Platform has been derived, shown in Figure 23. The concept in the former figure differentiates device, middleware and service layers at the bottom of the diagram that are all associated with symbloTe platform as the core IoT interoperability platform. Using its multiple levels of interoperability compliance, devices, platforms and services are seamlessly and flexibly register to symbloTe, match their individual data information models to produce conversion rules, allowing them to directly and bilaterally exchange their information. This way privacy of data can be achieved compliant with GDPR regulations. Such a mediation framework offered by symbloTe further permits flexible addition of new types of solutions thus supporting expandability of the SHAPES system to new solutions in the future.

Using 3rd-party expandability capabilities using concepts of “enablers”, representing approaches ranging from SDKs and APIs to explicit reusable code offered as Docker components, new solutions, services and applications can be grown on top of the SHAPES technological platform to address new and emerging e-Health domains.

6.2 Operational SHAPES Architecture

The core architecture defines the underlying system organisation and its components whose purpose is to support seamless connectivity and data exchange/sharing among interconnected devices, services, platforms and applications (as described in the concept architecture earlier), all of which being referred to as Digital Solutions. Such an architecture has been defined such that to offer the following basic functionalities:

1. Single sign-on to all Digital Solutions compatible and registered to SHAPES
2. Data authorisation managed by users and on Digital Solutions where data is stored
3. Seamless integration of devices, services, platforms and applications
4. Standard-independent interoperability w.r.t IoT data and Medical Information
5. Advanced data analytics of data shared among diverse Digital Solutions
6. Management of medical data over diverse Digital Solutions for same individual
7. GDPR-grade protection of private identifiable private data
8. Future extendibility to additional medical standards and data models
9. Optional sharing of capturing anonymised data sets for research purposes

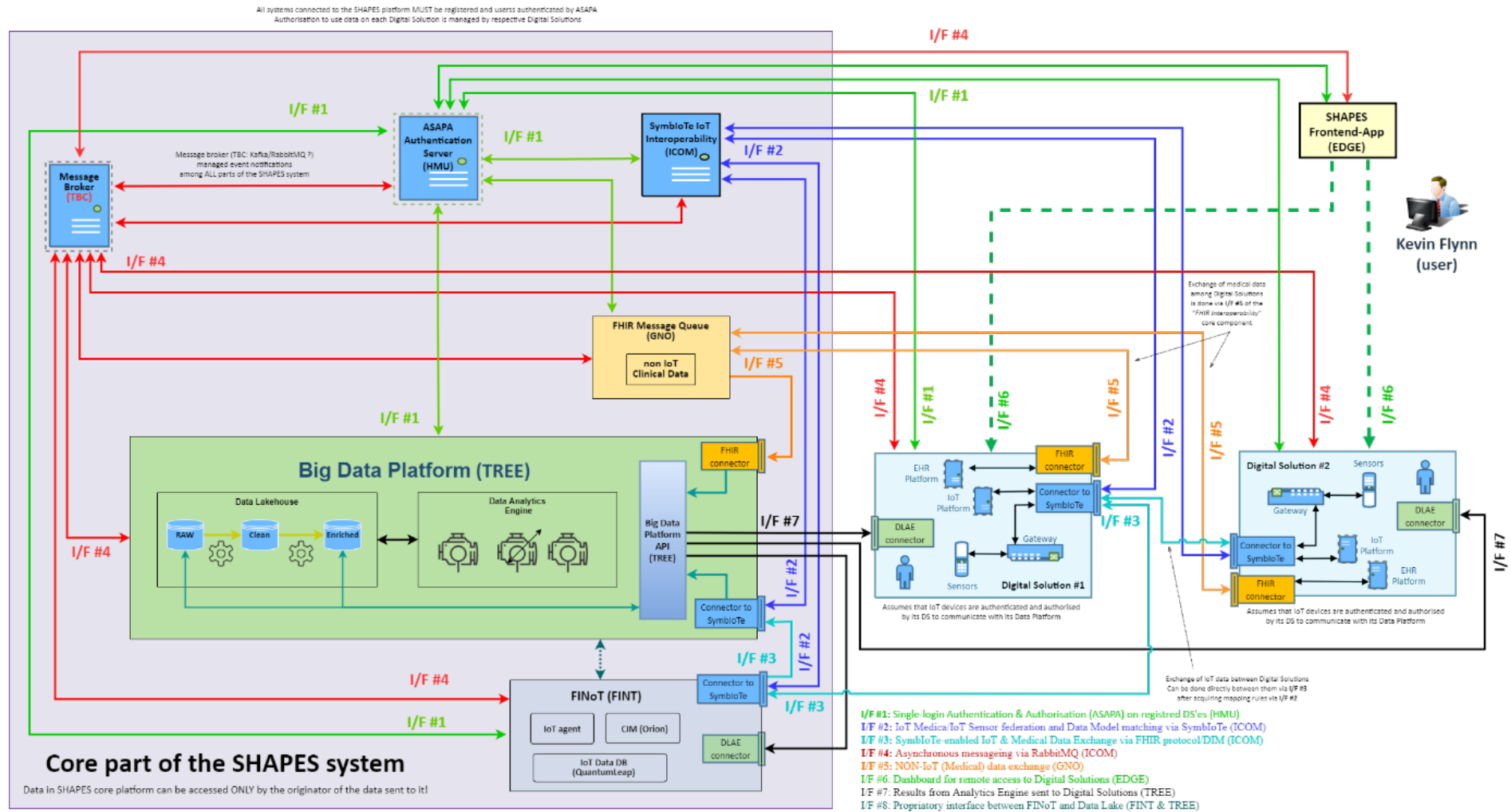


Figure 23: Operational SHAPES core architecture

In order to enable the above set of features and capabilities, the architecture has been defined as shown in Figure 23, comprising the following components that were mentioned earlier in section 2.1:

- *symbloTe IoT Interoperability Platform* from ICOM, an interoperable mediation framework that empowers the interconnection and interoperability of diverse IoT-based platforms, device, digital solutions and services (smart home, eHealth, telemedicine and smart city platforms) as well as the controlled and secure exchange of data and information. The symbloTe functions as a mediator between Digital Solutions for exchange of IoT Medial Data. It defines a mapping between two data models used on different Digital Solutions so they can exchange their data. This way the two IoT platforms are able to understand each other. The symbloTe does not store the data itself, just the metadata to support data mediating. For details refer to section 6.2.1.
- *FINoT IoT Data Management Platform* from FINT, a FIWARE-based IoT cloud management platform able to orchestrate embedded systems, to interconnect almost any kind of sensor, actuator and data logger and it is dedicated for industrial and semi-industrial usage. This platform provides data intelligence services while is capable for real-time handling of data artefacts including data fusion. Its integration SHAPES allows shifting various services to the cloud, provide required computational resources and handle massive chunks of collected data at a location-transparent centralized infrastructure and at the same moment retain historical data. It further enables smart neighbourhood and city capabilities like data acquisition and handling for weather, air quality, pollution, local public works, local transportation, local activities and other based on pilot's needs while also collect and provide information for the day to day activities within a specific community like available readings, exhibitions etc.
- *FHIR Medical Data Interoperability* with optional Medical Data repo from GNO, applies FHIR standard compliance to Medical Data exchanges among Digital Solutions w.r.t. data formats and elements and an application programming interfaces (API) for exchanging electronic health records (EHR). Thus it facilitates interoperability between legacy health care systems, to make it easy to provide health care information to health care providers and individuals on a wide variety of devices from computers to tablets to cell phones, and to allow third-party application developers to provide medical applications which can be easily integrated into existing systems. In SHAPES this information includes the electronic health record (HER), IoT medical data, ePrescriptions and laboratory results. FHIR can be used to enable exchange of non IoT clinical data between DS and DS-platform. FHIR will further facilitate the data exchange of third solutions coming in the platform.
- *Big Data Platform* combining Data Lakehouse with Analytics Engine from TREE, allows Digital Solutions to send their data to the Data Lakehouse for advance processing using AI-based Analytics Engine. Within the Data Lakehouse, different engines normalize and clean the raw data before it is stored in the Enriched layer. The raw data can be enriched (e.g. fixed spelling mistakes, added simple data sets,

filled missing pieces, used 3rd-party services like demographic databases to enrich data). Once completed, the enriched data can be consumed by the Data Analytics Engines (provided by VICOM and TREE). The data lake will store the data only temporarily for analytics. The principle is that only solution that need to consume the data for analytics can have access to such data (i.e. user requesting it from a given Digital Solution). The access rights are managed on the level of individual users and not organisations. The Data Analytics Engine (DAE) extracts the enriched information and returns the results, storing them in the “enriched layer”. Results are sent back to the Digital Solution that originated the data processing request. For more information refer to section 6.2.5.

- ASaPA Single Sign-on Authentication engine from HMU, an authentication and authorization framework. It continuously monitors the underlying network and detects existing and newly introduced network-enabled entities. It periodically assesses them against well-known vulnerabilities and certifies them against a CVSS score concerning their level of security (how vulnerable they are). These entities are assigned to the connectivity-appropriate network, according to their security level (CVSS score). After the assessment and certification of those entities, they are authenticated and authorized. This is achieved by utilizing the Authentication, Security and Privacy Assurance (ASaPA) component. To increase the security of user identification it encompasses (1) Users: using authentication mechanism based on Multimodal Biometrics, and (2) Devices and components: using IoT-friendly and Secure Stateless tokens. This incorporates user authentication mechanism from VICOM that is based on Multimodal Biometrics combining multiple physiological and/or behavioural human characteristic/modality carried out by hardware sensory devices and dedicated software modules for enrolment, verification, authentication or identification. Capabilities like face recognition, audio recognition and fingerprint (e.g., mobile sensor or touch screen), applying multi-factor authentication are integrated. For more information and details refer to section 6.2.6.
- Message Broker from ICOM, implements a RabbitMQ based common platform for applications to send and receive messages, by defining queues to which applications can connect in order to transfer a message or messages. This will enable asynchronous notification mechanisms for all core components and interconnected Digital Solutions to be able to schedule exchange of information among them, without a need for periodic checks for e.g. completion of expected operations, which might be tiresome especially in case of lengthy processing operations e.g. by Big Data Platform. In the context of the SHAPES project, an AMQP standard messaging protocol will be used with RabbitMQ as the message queue server for internal messaging between SHAPES core components. Each of them will implement a messaging solution, using the AMQP compliant framework (e.g. RabbitMQ) to be able to exchange information and events with other SHAPES core components. For more information refer to section 6.2.7.
- SHAPES Front-end Application from EDGE, brings a simple user interface providing a centralised access to the SHAPES Digital Solutions installed in the participant’s mobile phone or tablet. It also provides a mechanism for the single authentication of the user in the device. For more information refer to section 6.2.8.

- Gateway from FINT, is an accelerated access device aimed to improve acceleration capabilities in the gateway and handle local processes like speech analysis and AI analytics. It is a connection point among users, IoT devices and SHAPES online services. The SHAPES gateway is platform-independent and allows different vendors to easily incorporate or adapt it to interoperate with their own equipment and seamlessly connect to the SHAPES platform. In the first prototype of SHAPES system, the Gateway will be used to connect on premise medical devices to SHAPES system and/or selected Digital Solutions. In the final release some of such medical devices will; have a capability to register and connect to SHAPES via symbloTe Level 2 interoperability mechanisms. For more information refer to section 6.2.2.

Since most of the above components may require client-side development to be able to use their functionalities, the SHAPES architecture has introduced a concept of “enablers” being flexibly referred to either a Docker-based (refer to section 7 for description of SHAPES adopted integration approach), API or SDK interfacing. As many of the SHAPES core components are already in advanced development stage with interfaces being currently implemented and tested, descriptions of some of such “enablers” can be found in sections dedicated to individual core components.

6.2.1 The symbloTe orchestration middleware

In this section, the symbloTe architecture is described and the modules that comprise it are introduced. Also, their extension to cover the SHAPES needs is reported. Although symbloTe information model is a crucial part of it, its description and extension for the SHAPES needs is omitted here, as it has already been presented earlier in the document in section 5.

6.2.1.1 Introduction to symbloTe

The symbloTe is an IoT interoperability framework that enables the discovery and sharing of IoT devices and services across different IoT platforms and applications towards the creation of cross-domain and cross-platform applications. The abovementioned orchestration middleware is the outcome of the symbloTe project (symbiosis of smart objects across IoT environments) in ICT 30-2015 call "Internet of Things and Platforms for Connected Smart Objects" of the Horizon 2020 framework program of the European Community.

The symbloTe is not yet another IoT platform but rather a mediator that enables unified and secure access to IoT devices and services for third party applications and IoT platforms allowing the exchange of resource meta-information required to describe the IoT devices and services to be exposed.

6.2.1.2 The symbloTe architecture

The symbloTe architecture allows incremental deployment of the symbloTe functionality covering from syntactic and semantic interoperability (which are essential mechanisms for providing a unified view on the IoT platforms and their resources) to novel flavours of organizational interoperability (platform federations with bartering and trading mechanisms, dynamic Smart Spaces and roaming of IoT devices). In that

way, platform providers can achieve a desired level of collaboration within the symbloTe-enabled ecosystem choosing an adequate interoperability model for their business needs.

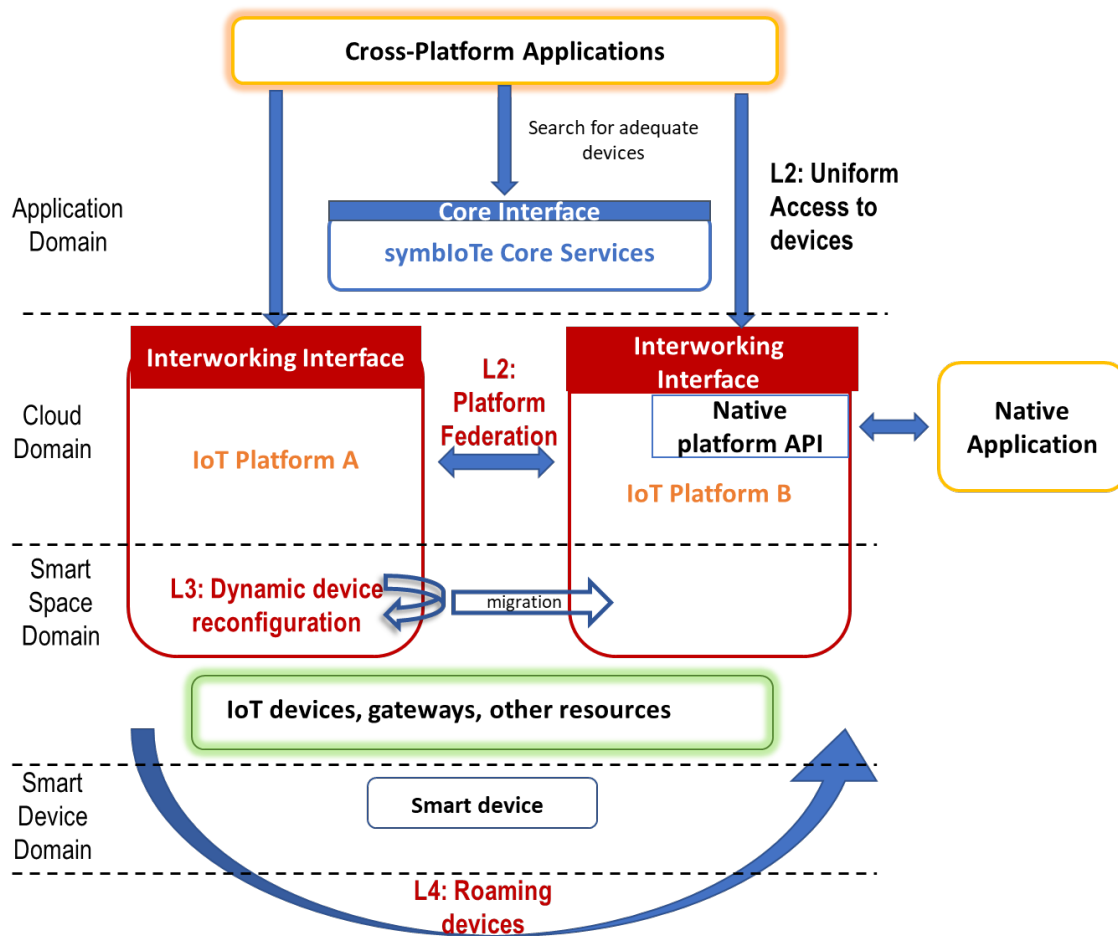


Figure 24: symbloTe Domains and Compliance Levels

More specifically, the architecture comprises of four layered domains, as depicted in Figure 24 above.

- **The Application Domain (APP)** enables platforms to register IoT resources, which they want to advertise and make accessible to third parties. It needs to rely on a common semantic representation of IoT resources, and hosts domain-specific enablers that facilitate the development of cross-platform and domain-specific application development (mobile and web applications).
- The **Cloud Domain (CLD)** provides a uniform and authorized access to virtualized resources, exposed by platforms to third parties through an open API (Interworking Interface). In addition, it offers services for IoT Platform Federations (associations between two or more platforms which are willing to share access to their IoT resources), enabling close platform collaboration in accordance with platform-specific business rules.

- The **Smart Space Domain (SSP)** delivers services for the discovery and registration of new IoT devices in dynamic local smart spaces, enabling the dynamic configuration of devices in accordance with predefined policies in those environments, and uniform interfaces for devices available in smart spaces.
- The **Device Domain (SDEV)** relates to smart devices and their roaming capabilities. It is assumed that devices have the capability to blend with a surrounding smart space while they are on the move. In other words, smart devices can interact with devices in a visited smart space, which is managed by a visited platform, in accordance with predefined access policies.

As mentioned, symbloTe offers flexible interoperability mechanisms enabled by an incremental deployment of symbloTe functionality across the listed architectural domains (APP, CLD, SSP and SDEV). The different compliance levels reflect different interoperability modes that a platform can choose to support, thus affecting the functionality and the corresponding symbloTe components, which have to be integrated on the platform side, within different domains.

- **Level 1 (L1) Compliant platform:** is a "lightweight" compliance level, where the platform may only choose to open its Interworking Interface to third parties to advertise and offer its virtualised resources through the symbloTe Core Services. L1 compliance supports the syntactic and semantic interoperability of IoT platforms in a symbloTe ecosystem, and affects only APP and CLD.
- **Level 2 (L2) Compliant Platform:** A platform may opt for a closer collaboration with another platform, by forming a platform federation. This compliance level enables platforms to federate and requires functionality needed for close organizational interoperability (e.g., for resource bartering/trading and trust management), thus requiring additional symbloTe components designed for CLD to be integrated within a platform space.
- **Level 3 (L3) Compliant Platform:** This compliance level assumes that platforms integrate symbloTe components within their smart spaces to simplify the integration and dynamic reconfiguration of devices within local spaces.
- **Level 4 (L4) Compliant Platform:** This level offers support for device roaming and can enable the interaction of smart devices with visited smart spaces. A prerequisite is that a platform is already marked as L1, L2 & L3-compliant, so that smart spaces can discover new visiting devices and integrate them dynamically (e.g., grant access to certain local resources) in accordance with Service Level Agreements (SLAs) between a visited platform and one managing visiting devices.

For the needs of the SHAPES project, L1 and L2 compliance levels will be at least required to make IoT platforms symbloTe-enabled. The essential symbloTe building blocks for achieving L1/L2 compliance is shown in Figure 25. The two main building blocks of symbloTe to be used are:

- **symbloTe Core**, which is the central point where third-party applications can search for cross-platform resources to be deployed on a central server
- **symbloTe Cloud**, which is the symbloTe connector on the side of the IoT platform to be deployed to become symbloTe-enabled

Finally, symbloTe libraries and clients are available for application developers and users to be able to interact in a symbloTe-enabled ecosystem. These parts are explained in more detail in the next sections.

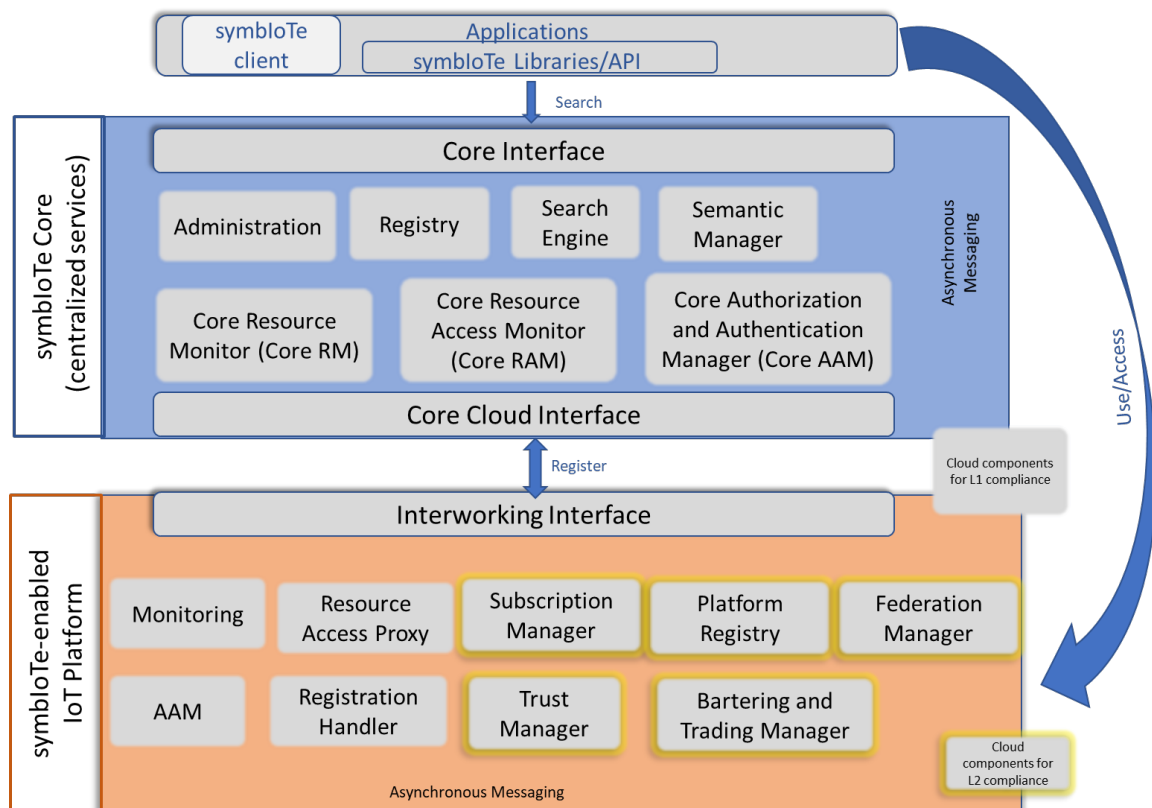


Figure 25: Interconnection of symbloTe-enabled entities for the SHAPES needs.

6.2.1.2.1 The symbloTe Core

It is a centralised part (Application domain components) that acts as an IoT search engine, where IoT platforms can register their resources while users, such as third-party applications, can search for these resources. More specifically, the symbloTe core provides search, security, administration and monitoring features, storing metadata about IoT platforms and their resources, as well as their domain models (the information models used) in order to enable device and service discovery and access. Note that IoT platforms register only resource metadata to the symbloTe Core services, while resource access requests need to be directed to the corresponding IoT platform. This is done so that platforms can maintain complete control over their data.

The symbloTe core components are the following:

- **Administration** - provides a web-based GUI as well as an API for platform administrators. The provided functionality will include: the creation and management of the platform and its properties and the generation of platform credentials for authentication.
- **Registry** – it is a repository for storing all platform-related metadata. It provides an API for registering and updating platform's devices/services, which are described

using the symbloTe Information Model. During the registration process, symbloTe-specific unique IDs are generated for each device/service.

- **Search Engine:** enables applications to find relevant registered devices/services registered within the Registry.
- **Semantic Manager:** stores information models used within symbloTe and verifies if the resources conform to the information model they claim to be using.
- **Core Resource Monitor:** tracks the availability of registered devices to ensure their availability.
- **Core Resource Access Monitor:** tracks information about resource popularity.
- **Core Authentication and Authorization Manager (Core AAM):** ensures that trusted platforms register resources with symbloTe, while mapping resource access rights to proper credentials.

There are two kinds of interworking interfaces in the symbloTe Core: the core Interface which serves northbound traffic coming from third parties (e.g., applications searching for resources) and the cloud Core Interface which serves southbound traffic coming from IoT platforms (e.g., applications).

The symbloTe Core is deployed for testing purposes at: <https://intracom-core.symbloTe-h2020.eu/>. An additional server that will be running the latest stable version will also be deployed at ICOM's premises.

6.2.1.2.2 The symbloTe Cloud

The symbloTe cloud is an IoT platform part (Cloud Domain components) that acts as a connector between symbloTe and the IoT platform to allow unified and secured remote access to virtualized IoT resources through an open API (Interworking Interface). Thus, the Interworking Interface defines a uniform API for resource access, authentication and authorization across platforms. An IoT platform must integrate the symbloTe Interworking Interface to open up its northbound interface and provide access to IoT services.

The symbloTe Cloud services enable IoT Platform providers (platform owners) to register desired resources to symbloTe Core services. Also, they provide the means for applications/enablers to access those resources, found through the symbloTe Core Services, in a uniform and secure way. These are functionalities required for achieving L1 compliance. The cloud services also implement specific functionality for the management of platform federations required for L2 compliance, i.e. the exchange of information between collaborating IoT platforms as well as bartering and trading mechanisms.

The symbloTe cloud components are the following:

- **Registration Handler (RH):** enables the IoT Platform Provider to register resources to the symbloTe Core Services.
- **Resource Access Proxy (RAP):** receives requests for resource access from an application/enabler, which found resource metadata by using symbloTe Core Services.

- **Monitoring:** monitors the status of IoT Devices and records when applications/enablers access IoT Devices/Composite IoT services.
- **Authentication and Authorization Manager (AAM):** enables a common authentication and authorization mechanism for all L2 Platforms and applications.
- **Federation Manager:** manages all federation affiliations and signed SLAs per platform, handles SLA violation notifications and triggers optimization requests and trust calculation updates.
- **Bartering and Trading Manager (BTM):** manages bartering and trading actions prior to establishing federations.
- **Platform Registry:** stores resource metadata for L2 resources shared within a platform federation.
- **Subscription Manager:** is used for publish/subscribe interactions between platforms within a federation, and to update records stored within the Platform Registry.
- **Trust Manager:** computes trust levels for other platforms within a federation.

The symbloTe cloud components are essentially the part of the symbloTe software that needs to be downloaded and integrated by an IoT platform in order to become symbloTe-enabled. Apart from the aforementioned symbloTe cloud components (the set required for either L1 or L2 compliance), platform owners need to deploy a platform specific plugin, the RAP platform plugin, which is a part of the RAP component in charge of accessing the resources exposed by the platform. Note that RAP plugin is platform-specific and requires development from the platform owner who may choose to register their own platform specific information model. Thus, platform providers are required to implement the code which can acquire the data from the platform (the RAP plugin). After obtaining the data, the RAP plugin forwards it to the generic part of the RAP, which then handles all communication with the upper layers, i.e., the users of the exposed data. The RAP plugin supports:

- acquiring the current value from resource,
- reading history values from resource,
- subscribing to resource values and creating notifications,
- writing values into resources when dealing with actuating devices

Corresponding methods are defined within the provided platform plugin that need to be implemented by each IoT platform. After the implementation, the platform plugin will be integrated with the generic part of the RAP during the build process.

Once the RAP plugin implementation is ready, the rest of the steps to make an IoT platform symbloTe L1- or L2-enabled needs to be followed. The process of making IoT platforms symbloTe L1-enabled consists of downloading and setting up the symbloTe Cloud for L1 compliance by integrating the cloud components with the platform, and then registering the platform and resources to symbloTe Core Services. The platform owners, who are interested in making IoT platforms symbloTe L2-enabled, in order to create IoT Federations, will need to deploy the symbloTe Cloud with additional cloud components for L2 compliance (symbloTe Cloud for L2 compliance) within their platform's space and interconnect them with their platforms,

register the platform and create the federation with other ones. Afterwards, they may choose to share their resources freely within the federations, by guaranteeing predefined Quality of Service levels related to resource access, or by bartering them for future access to another platform's resources. In the latter case, registration of federated resources happens without a central registry and access is granted directly from platforms to applications, without any kind of intermediary or centralised registry at the federation level.

Each IoT Platform owner, such as FINT, will need to setup and maintain these components on the platform side, as well as integrate them with their platform for either L1 or L2 compliance. Detailed documentation on how to make a platform L1 or L2 compliant can be found in <https://github.com/symbloTe-h2020/symbloTeCloud/wiki> and Instructions for symbloTe compliance are provided in “Annex 2 Instructions for symbloTe compliance”.

6.2.1.2.3 The symbloTe API for applications

An application that wants to interact within a symbloTe-enabled ecosystem (i.e., become symbloTe-enabled) needs to download and use available symbloTe libraries. Security rules must be strictly applied by application developers when communicating with symbloTe components. In particular, the application must gain explicit access rights to interact both with the Core or any symbloTe-enabled IoT platform based on the policies defined for exposing its resources to third parties. Example symbloTe clients implemented in Java are also available for application developers to interact with symbloTe. However, in order to facilitate the use of symbloTe from SHAPES partners, a symbloTe API component (Spring Boot application) is under development that offers a set of REST services designed to cover basic client requirements. These are included in Table 7:

Interface	Method	Description	Provider	Caller	Payload
Login	REST POST	User login	symbloTe	Any SHAPES component/ DS	Parameters: 'username', 'password' Response: 'token'
getListOfL1	REST POST	Get list of L1 resources	symbloTe	Any SHAPES component/ DS	Parameters: 'token', 'platformid' Response: JSON array
getListOfL2	REST POST	Get list of L2 resources	symbloTe	Any SHAPES component/ DS	Parameters: 'token', 'fedid', 'platformid' Response: JSON array

getL2Resource	REST POST	Get a L2 resource	symbloTe	Any SHAPES component/DS	Parameters: 'token', 'fedid', 'resourceName', 'platformid' Response: JSON object
getL1Resource	REST POST	Get a L1 resource	symbloTe	Any SHAPES component/DS	Parameters: 'token', 'resourceid', 'platformid'. Response: JSON object
registerL1Resource	REST POST	Register L1 resource	symbloTe	Any SHAPES component/DS	Parameters: 'resourceName', 'pluginId', 'resourcetype', 'token'
shareL1Resource	REST POST	Share L1 resource to a federation	symbloTe	Any SHAPES component/DS	Parameters: 'token', 'fedid', 'resourceName', 'platformid'
registerUser	REST POST	Register a user to symbloTe	symbloTe	Any SHAPES component/DS	Parameters: 'username', 'password', 'email', 'platformid'

Table 7 *symbloTe API*

Finally, symbloTe has also introduced the concept of symbloTe enablers to simplify the development of applications within the symbloTe ecosystem. Enablers are cloud-based, back-end services that add value on top of platforms and their resources within the symbloTe ecosystem, by taking over complex tasks, such as some sort of data analytics or real-time processing services, to offer simple APIs to the applications. Thus, enablers may simplify the development of applications by concealing the complexity of the underlying symbloTe ecosystem hosting many IoT platforms. Each Enabler instance integrates generic symbloTe-defined components, which are independent of its domain-specific features. They represent the basic Enabler structure that is easy to use and deploy so that a developer only needs to focus on the value-added and domain-specific features when building a new Enabler. The generic architecture of an Enabler is composed of two types of components, components in common with symbloTeCloud which implement the platform role and Enabler-specific components. The use and implementation of Enablers will also be investigated in the next period, as the SHAPES project evolves.

6.2.1.3 Integration of symbloTe with ASAPA

As described earlier, symbloTe contains its own Authentication and Authorization Manager Module to protect resources from unauthorised access, by providing distributed and decoupled mechanisms for authentication and authorization, namely the Attribute Based Access Control (ABAC) with token-based authorization. It authenticates components belonging to different symbloTe-enabled IoT platforms to be able to use services offered by symbloTe core services or other symbloTe-enabled IoT platforms such as resource registration and update, but also applications (users) registered in the core layer or in the symbloTe-enabled platform to search and access resources.

For the needs of the SHAPES project, symbloTe is extended to add communication with the ASAPA component for authentication and authorization purposes. More specifically, the workflow used in symbloTe for the registration of a user is modified so that a user account can be created or updated to SHAPES through symbloTe, assigning the required groups/roles/permissions to the users for its organization. The user login procedure in symbloTe is also modified so that a registered user can login to SHAPES through it. The symbloTe will communicate with ASAPA to validate user's credentials and allow user login. When login is successful the user token is retrieved, otherwise user login fails. Finally, when access to symbloTe services is requested, information of the user and ABAC information will be retrieved from the ASAPA component to protect them from unauthorized access and control their use.

6.2.1.4 Dependencies on other tools

The symbloTe software makes use of the following external open-source tools:

- **RabbitMQ**, which is the message queue server for internal messaging between platform components.
- **Mongo DB**, which is the database used by symbloTe components.
- **Nginx**, which is the proxy server serving as an Interworking Interface for receiving all the requests directed to symbloTe APP/CLD and redirects the requests to corresponding application/platform components. Nginx will need to be configured so that it redirects all the requests correctly to the corresponding components.
- **Zipkin service**, which collects logs from various services
- **Eureka service**, which allows discovery of application/cloud components.
- **Cloud Config service**, which is responsible for configuration of user credentials, ports and interfaces used in communication between symbloTe Cloud and symbloTe Core Services.

6.2.1.5 Concluding Remarks

In this section, the symbloTe architecture and main building blocks have been described. Also, the extension of symbloTe modules for the SHAPES needs has been presented. Note that the extension of symbloTe information model towards the SHAPES needs has been described in the previous section. During the next period of

the project the design of the interfaces required in the context of the SHAPES project will be finalised, and the use of enablers and smart spaces will be also investigated. The finalised solution will be documented in the D.4.3 deliverable due in M24.

6.2.2 Gateway

SHAPES Gateway (GW) facilitates the interconnection of the edge IoT devices with the SHAPES Core cloud platform enabling as such the accommodation of the IoT collected data to the FINoT IoT platform (part of the SHAPES core); further to that it enables, via the symbloTe connector, for the registration of edge IoT resources in the symbloTe ecosystem. In this direction, and as Figure 26 depicts, the SHAPES GW will utilise several hardware and software communication and processing modules.

More specifically, and as far as it concerns the hardware, the GW will be built on a hybrid hardware platform utilising ARM and Acceleration modules (e.g., FPGA, GPU) with the former destined to support light-weight computing tasks, management operations, data pre-processing and communication functionalities and the latter intended to provide the necessary computing resources for handling potentially more sophisticated processing tasks (e.g., Machine Learning analysis) at the edge level. Regarding the communication interfaces, the GW is envisaged to support 6LoWPAN, Bluetooth, Wi-Fi and Ethernet interfaces for interacting with the edge IoT sensors whereas for the communication with the SHAPES cloud Ethernet, Wi-Fi and optionally 4G are to be available.

As far as it concerns the SW modules, the GW employs a System Monitor (SM) module that will provide information about the device's operational status, a Sensor Data Handler (SDH) that will have the responsibility to accept the data stemming from the sensors, adapt them where needed, towards meeting any upper (or switching) layer protocols requirements, and after that forward them to the FINoT connector. There the data will be forwarded to the FINoT IoT platform where they will further be processed and stored; as an alternative SDH can also store locally the IoT data and interact with the Data Processing Manager (DPM) for arranging for the stored IoT data to be analysed from an accelerated algorithm (APA in Figure 26) loaded to the GW's acceleration module; the processed data could then in turn stored locally in the GW or forwarded to the FINoT platform. Further to that, SDH has the option to register the IoT (processed and/or raw) data that are locally stored to the GW as resources available to any third symbloTe registered IoT platform via the symbloTe connector. The Gateway Manager (GWM) will coordinate and handle the operational status of the deployed SW modules (e.g. (re)start, stop) whereas the dashboard will provide an easy way to manage the modules operation and also visualise the system monitoring information. Finally, the Proxy will accept incoming questions and will forward them to the specific micro service which usually will be the SDH, the symbloTe connector or the Dashboard whereas the Authentication and Authorisation Manager (AAM) will handle the interactions with the ASAPA module as far as it concerns the authentication of the gateway in the SHAPES ecosystem.

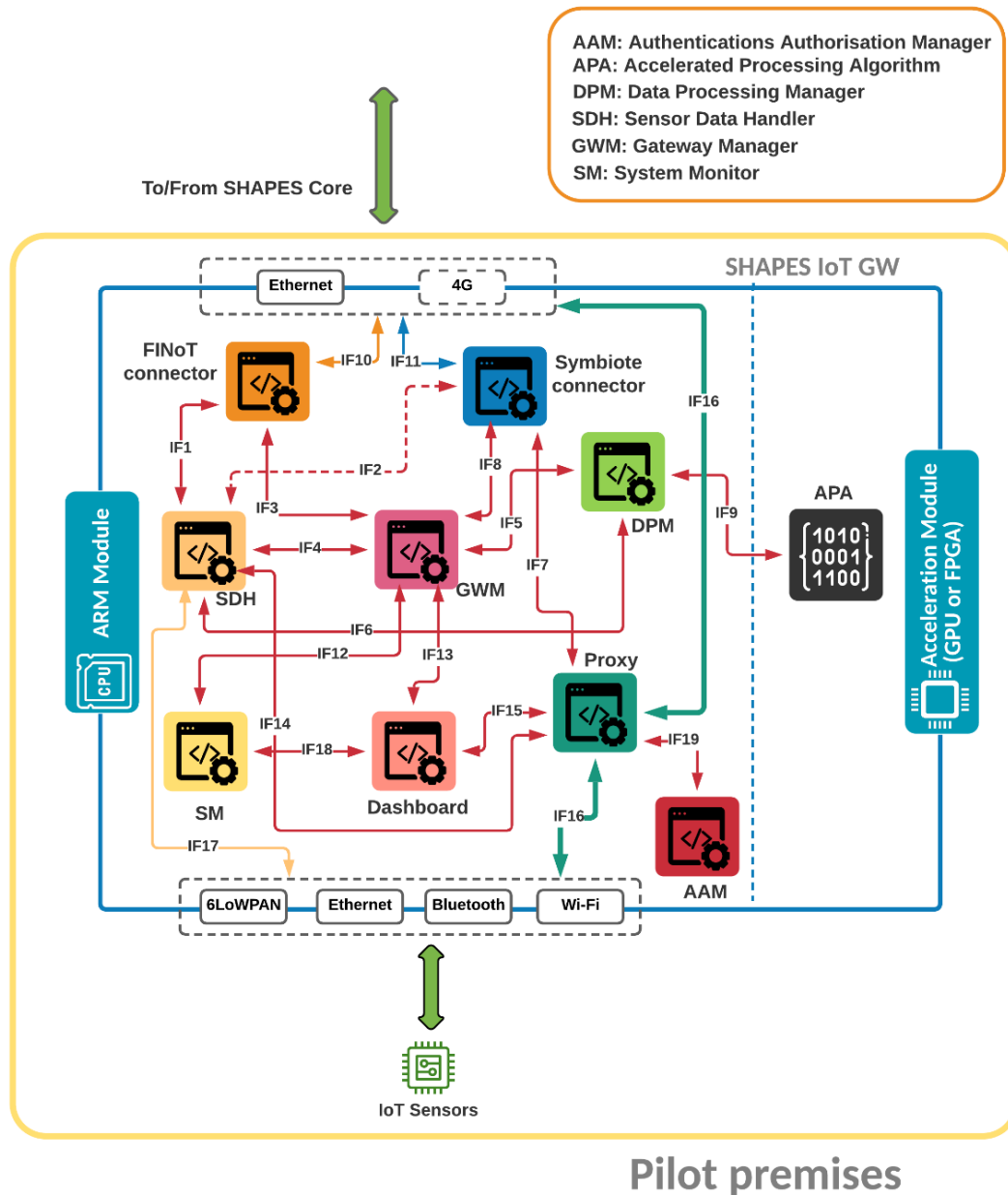


Figure 26 SHAPES GW

The SHAPES GW will utilise Docker framework to allow for deployment of additional connectors as micro services (e.g., EDGE’s connector for pushing relevant data to the EDGE platform) or SDHs that may be used to connect to 3rd-party platforms or handle sensor data that are not complying to the specifications of native SDH modules (compatible with FINoT supported sensors) respectively. Nevertheless, for the newly introduced connectors or SDHs to be able to utilise services of the native GW modules they should implement the required interfaces and adhere to the agreed project deployment, operational and security rules as dictated and enforced by the ASAPA. Using as reference Figure 26, Table 8 describes further the SHAPES GW interfaces.

Interface	Description	Type	Peers
-----------	-------------	------	-------

IF1	SDH uses this interface to get the forward the data to the connector module that will push the IoT data to the IoT cloud platform. The default SDH will forward the data to the FInoT connector	Internal, RESTful	SDH, FInoT connector
IF2	This interface can be used from the SDH when in need to register locally (i.e., in the GW) stored IoT resources in the symbloTe enabled federated SHAPES IoT ecosystem.	Internal, RESTful	SDH, symbloTe connector
IF3	This interface is used from the GWM to be able to manage and monitor the operational status of the FInoT connector.	Internal, RESTful	GWM, FInoT connector
IF4	This interface is used from the GWM to be able to manage and monitor the operational status of the SDH.	Internal, RESTful	GWM, SDH
IF5	This interface is used from the GWM to be able to manage and monitor the operational status of the DPM.	Internal, RESTful	GWM, DPM
IF6	This interface will be used from the SDH to notify the DPM when incoming or already locally stored IoT data need to be analyzed locally from algorithms running in the acceleration subsystem of the GW.	Internal, RESTful	SDH, DPM
IF7	This interface is used from the Proxy to forward to the symbloTe connector any incoming requests destined to it.	Internal, RESTful	Proxy, symbloTe connector
IF8	This interface is used from the GWM to be able to manage and monitor the operational status of the symbloTe connector.	Internal, RESTful	GWM, symbloTe connector
IF9	This interface is used from the DPM to feed data to the APA and get back the analysis results.	IPC (e.g., named pipes)	DPM, APA
IF10	FInoT connector uses this interface to forward the data to the FInoT IoT platform.	External, MQTT (RESTful is also available)	FInoT connector, FInoT IoT platform
IF11	The symbloTe connector uses this interface to communicate and interact with symbloTe core platform.	External, RESTful	symbloTe connector, symbloTe core platform

IF12	This interface is used from the GWM to be able to manage and monitor the operational status of the SM.	Internal, RESTful	GWM, SM
IF13	This interface is used from the GWM to be able to manage and monitor the operational status of the Dashboard. Further to that it is used to feed the Dashboard with information about the operational status (e.g., started, stopped, etc.) of the other deployed modules for being visualized.	Internal, RESTful	GWM, Dashboard.
IF14	This interface is used from the Proxy to forward to the SDH any incoming requests destined to it.	Internal, RESTful	Proxy, SDH
IF15	This interface is used from the Proxy to forward to the Dashboard any incoming requests destined to it.	Internal, RESTful	Proxy, Dashboard
IF16	This interface is used from the Proxy to receive, handle and control any incoming requests from other SHAPES entities to the GW modules	Internal, RESTful	Proxy, External entities
IF17	This interface is to be used for sending IoT data directly to the SDH when the used IoT data transfer protocol is not supported from the Proxy.	External, TBD based on the IoT sensor capabilities.	SDH, IoT sensors
IF18	This interface is used from the SM to forward to the Dashboard the monitored system information towards being visualized.	Internal, RESTful	SM, Dashboard
IF19	This interface is used from the AAM to interact via the Proxy with the ASAPA module towards complying with the SHAPES AA scheme.	Internal, RESTful	AAM, Proxy

Table 8 SHAPES GW interfaces

6.2.3 Component: FInoT IoT platform

FInoT (see Figure 27) is a FIWARE-based IoT cloud management platform able to orchestrate and interconnect almost any kind of sensor, actuator and data logger and it is dedicated for industrial and semi-industrial usage.

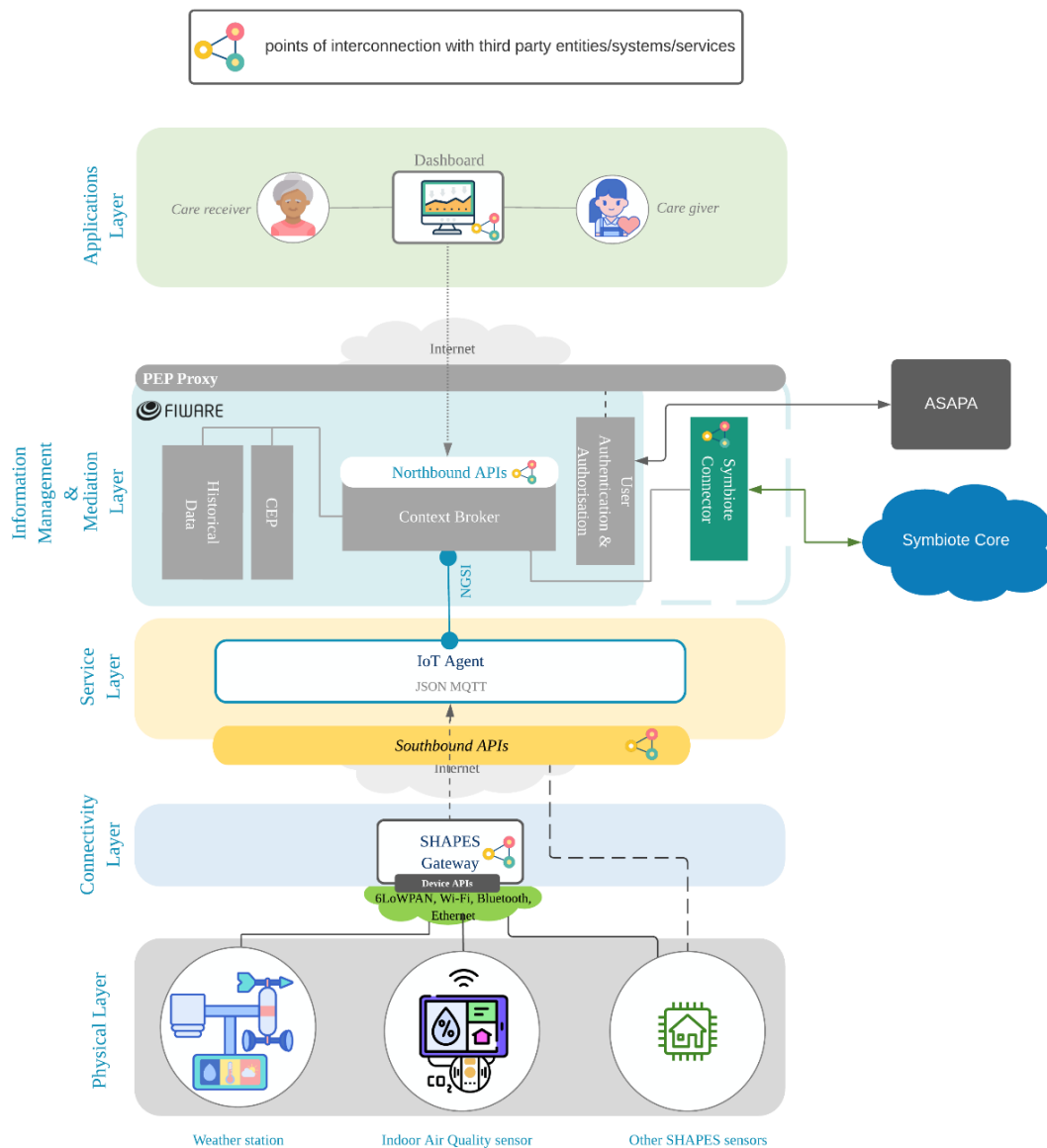


Figure 27 FINoT platform architecture

The integration of FINoT Platform into SHAPES TP will enable for the accommodation, via the SHAPES GW or directly via the southbound API, of IoT data (non-medical) generated at the edge in a platform tailored by design to carry out IoT data management/fusion and to provide services based on the utilisation of these accommodated data. The introduction of symbloTe connector in the FINoT ecosystem will allow to the other SHAPES IoT platforms to locate and consume these IoT data whereas in cases that the symbloTe powered communication is not available or applicable there will be also the option of using directly the FINoT RESTful API for consuming the hosted IoT resources. FINoT supported data type is based on NGSiv2 protocol while interconnection to the platform is supported through HTTP REST, JSON and MQTT for both inputs and outputs. As part of the SHAPES Digital Solutions FINoT will provide to SHAPES' users relevant information on climate, air quality, pollution,

urban noise levels, energy, local public works, parking, local transportation and local activities where this information will be available.

The following paragraphs discuss the purpose of each FInoT architectural layer as depicted in Figure 27.

Physical Layer: The physical layer is the lowest layer of the architecture of the IoT platform, and its main responsibility is to gather all the available data from the environment (such as humidity, temperature etc.) and transform them in a digital format. The main purpose of the objects/devices is to collect all this information and forward it to the upper layers using communication technologies such as Wi-Fi, Bluetooth, 6LoWPAN. In addition, this layer is responsible to translate high-level command from the upper layers to actuators, if applicable.

Connectivity Layer: This layer consists of IoT gateways (i.e., SHAPES GW in the project's context). The main purpose of this layer is to provide a secure and reliable communication channel between the IoT devices and the cloud (e.g., FInoT). A gateway is able to communicate with the IoT devices using short-range communication technologies such as Bluetooth, NFC, 6LoWPAN but also provides connectivity to the upper's layers through the core network (4G, Wi-Fi, LAN etc.) Alternatively, if the sensor devices are not compatible with the SHAPES GW, they can use directly the provided FInoT API for forwarding the IoT data to the FInoT.

Service Layer (Object Abstraction Layer): The service layer acts as an intermediate layer between mediation and the lower layers. The responsibility of this layer is to translate the raw data coming from the lower layers to a context aware entity (NGSI) and vice versa. In this layer there are 2 main components:

- **MQTT Broker:** The MQTT broker provides the transportation layer between the gateways and the cloud. The broker exposes the southbound API to the gateways.
- **IoT Agent:** The IoT Agent is a service responsible to translate the raw data from the gateway to an NGSIv2 representation and notify the context broker for the updated values. Additionally, the IoT Agent is responsible to translate commands from the upper layers to raw data and forward this data to the gateway.

Mediation layer: The mediation layer is the “brain” of the platform. Can store, process and aggregate data. Additionally, it can execute relevant tasks using an advance CEP mechanism. This layer has the following components:

- **Context Broker (Orion):** Orion Context Broker is the core component of the platform. Collects, manages, and provides access to data from different sources. Technically, the context broker consists of a MongoDB database and an NGSIv2 REST API on top of it.
- **Historical Data Storage (Quantum Leap):** This module is responsible of managing (storing and retrieving) historical raw and aggregated time series data registered in an Orion Context Broker.
- **CEP (Complex Event Processing):** The CEP module uses Kafka stream processor as a communication channel and is responsible to execute complex event driven tasks over real-time streams of sensor data. Additionally, it can run asynchronous tasks/jobs in the platform.

Information Management Layer: This layer consists of 2 main components.

- **User Management, Authentication & Authorization:** This service is responsible for managing the various users in the system. Provides all the necessary tools to authenticate and authorize the users to access an application, API, service, or any other data resource. In the context of SHAPES, this module will interact with ASAPA towards conforming to the SHAPES AA scheme.
- **FINoT Core:** This service provides all the necessary REST APIs endpoints for the application layer to communicate with the mediation layer.

Application Layer: The top-most level of the architecture is the application layer. The application layers consist of the actual applications (interfaces) that are responsible to represent the actual data to the user.

6.2.3.1 FINoT API

FINoT provides a RESTful APIs that gives programmatic access to FINoT middleware API. All API access is over HTTPS, and accessed from <https://api.shapes.fintot.cloud> whereas all data are sent and received as JSON; in addition all timestamps are returned in ISO 8601 format: YYYY-MM-DDTHH:MM:SSZ (UTC Zulu Time). Furthermore, all API requests must be written as HTTP requests, and include the following components:

- HTTP Method: Describes the type of HTTP action (POST, GET, PUT or DELETE).
- URL: Describes the resource you are creating or accessing, along with any optional arguments.
- HTTP Headers: Specifies attributes of the request, including authentication, encoding and request format.
- Request Body: Describes resources or specifies a call-control script.

Moreover, the URL entries must adhere to the following format:

- `<protocol>://<host>:<port>/<module>/<version>/<ResourceName>`

The API provides numerous methods spanning from authentication to data read and write operations. A detailed list of the supported operations along with the necessary information for using them is provided in the “*Annex 1 FINoT Middleware API for SHAPES*”.

6.2.4 FHIR Medical Interoperability

The FHIR Medical interoperability component facilitates the transmission of FHIR messages among the SHAPES parties. As mentioned in section 4.1.1, FHIR is used as a healthcare interoperability standard, in order to establish interoperability among SHAPES Digital Solutions and external solutions that will need to interact with the SHAPES ecosystem. The FHIR messages are leveraged for the exchange of non-IoT medical data. The Table 9 shows representative examples of non-IoT medical data.

Patients demographics	Medical Record	Medication
Name	Conditions	Ingredient
Gender	Allergies	Dosage
Birth Date	Medication	Frequency
Address	Procedures	Commercial Name
Marital Status	Vaccinations	
Contact		

Table 9 non-IoT Medical data

The following three networking scenarios are foreseen within SHAPES ecosystem:

- 1) Exchange of non-IoT medical data between Digital Solutions (DS). For example, a DS #1 exchanges medical record information with DS #2.
- 2) Exchange of non-IoT medical data between a DS and the Big Data Platform. For example, the DS exchanges patient demographics information that is required by the big data platform to perform relevant data analysis.
- 3) Exchange of non IoT medical data between third party DS that are integrated within the SHAPES ecosystem.

From a technical point of view, FHIR messages are composed of a set of FHIR resources which represent the non-IoT medical data that will need to be exchanged. Below, there are examples of FHIR resources which represent non-IoT medical data.

Patient

```

<Patient xmlns="http://hl7.org/fhir">
  <!-- from Resource: id, meta, implicitRules, and language -->
  <!-- from DomainResource: text, contained, extension, and modifierExtension -->
  <identifier><!-- 0..* Identifier An identifier for this patient --></identifier>
  <active value="[boolean]"/><!-- 0..1 Whether this patient's record is in active use -->
  <name><!-- 0..* HumanName A name associated with the patient --></name>
  <telecom><!-- 0..* ContactPoint A contact detail for the individual --></telecom>
  <gender value="[code]"/><!-- 0..1 male | female | other | unknown -->
  <birthDate value="[date]"/><!-- 0..1 The date of birth for the individual -->
  <deceased[x]><!-- 0..1 boolean|dateTime Indicates if the individual is deceased or not --></deceased[x]>
  <address><!-- 0..* Address Addresses for the individual --></address>
  <maritalStatus><!-- 0..1 CodeableConcept Marital (civil) status of a patient --></maritalStatus>
  <multipleBirth[x]><!-- 0..1 boolean|integer Whether patient is part of a multiple birth --></multipleBirth[x]>
  <photo><!-- 0..* Attachment Image of the patient --></photo>

```

```

<contact> <!-- 0..* A contact party (e.g. guardian, partner, friend) for the patient -->
<relationship><!-- 0..* CodeableConcept The kind of relationship --></relationship>
<name><!-- 0..1 HumanName A name associated with the contact person --></name>
<telecom><!-- 0..* ContactPoint A contact detail for the person --></telecom>
<address><!-- 0..1 Address Address for the contact person --></address>
<gender value="[code]"/><!-- 0..1 male | female | other | unknown -->
<organization><!-- ?? 0..1 Reference(Organization) Organization that is associated with the contact --></organization>
<period><!-- 0..1 Period The period during which this contact person or organization is valid to be contacted relating to this patient --></period>
</contact>

<animal> <!-- 0..1 This patient is known to be an animal (non-human) -->
<species><!-- 1..1 CodeableConcept E.g. Dog, Cow --></species>
<breed><!-- 0..1 CodeableConcept E.g. Poodle, Angus --></breed>
<genderStatus><!-- 0..1 CodeableConcept E.g. Neutered, Intact --></genderStatus>
</animal>

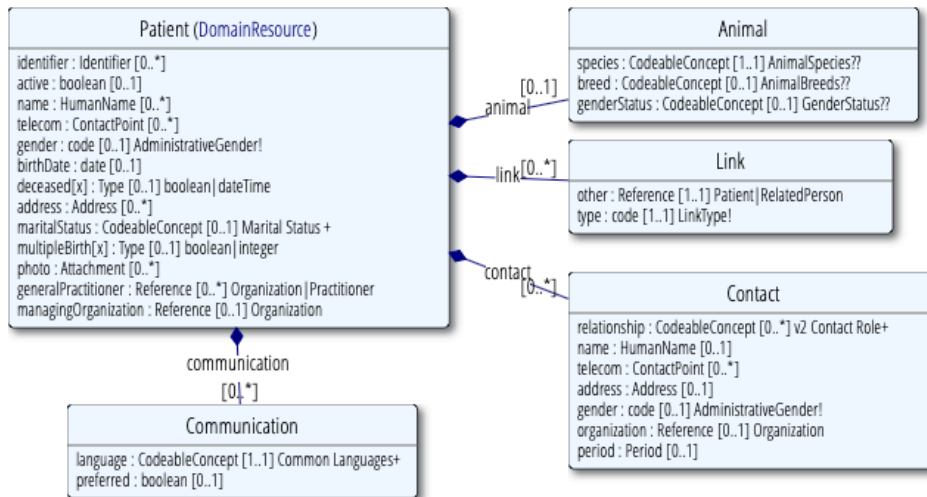
<communication> <!-- 0..* A list of Languages which may be used to communicate with the patient about his or her health -
->
<language><!-- 1..1 CodeableConcept The language which can be used to communicate with the patient about his or her h
ealth --></language>
<preferred value="[boolean]"/><!-- 0..1 Language preference indicator -->
</communication>

<generalPractitioner><!-- 0..* Reference(Organization|Practitioner) Patient's nominated primary care provider --></general
Practitioner>

<managingOrganization><!-- 0..1 Reference(Organization) Organization that is the custodian of the patient record --></man
agingOrganization>

<link> <!-- 0..* Link to another patient resource that concerns the same actual person -->
<other><!-- 1..1 Reference(Patient|RelatedPerson) The other patient or related person resource that the link refers to --></
other>
<type value="[code]"/><!-- 1..1 replaced-by | replaces | refer | seealso - type of link -->
</link>
</Patient>

```



Condition

```
<Condition xmlns="http://hl7.org/fhir">
  <!-- from Resource: id, meta, implicitRules, and language -->
  <!-- from DomainResource: text, contained, extension, and modifierExtension -->
  <identifier><!-- 0..* Identifier External Ids for this condition --></identifier>
  <clinicalStatus value="[code]"/><!-- ?? 0..1 active | recurrence | inactive | remission | resolved -->
  <verificationStatus value="[code]"/><!-- ?? 0..1 provisional | differential | confirmed | refuted | entered-in-error | unknown -->
  <category><!-- 0..* CodeableConcept problem-list-item | encounter-diagnosis --></category>
  <severity><!-- 0..1 CodeableConcept Subjective severity of condition --></severity>
  <code><!-- 0..1 CodeableConcept Identification of the condition, problem or diagnosis --></code>
  <bodySite><!-- 0..* CodeableConcept Anatomical location, if relevant --></bodySite>
  <subject><!-- 1..1 Reference(Patient| Group) Who has the condition? --></subject>
  <context><!-- 0..1 Reference(Encounter| EpisodeOfCare) Encounter or episode when condition first asserted --></context>
  <onset[x]><!-- 0..1 dateTime|Age|Period|Range|string Estimated or actual date, date-time, or age --></onset[x]>
  <abatement[x]><!-- ?? 0..1 dateTime|Age|boolean|Period|Range|string If/when in resolution/remission --></abatement[x]>
  <assertedDate value="[dateTime]"/><!-- 0..1 Date record was believed accurate -->
  <asserter><!-- 0..1 Reference(Practitioner| Patient| RelatedPerson) Person who asserts this condition --></asserter>
  <stage> <!-- 0..1 Stage/grade, usually assessed formally -->
  <summary><!-- ?? 0..1 CodeableConcept Simple summary (disease specific) --></summary>
  <assessment><!-- ?? 0..* Reference(ClinicalImpression| DiagnosticReport| Observation) Formal record of assessment --></assessment>
</stage>
<evidence> <!-- 0..* Supporting evidence -->
<code><!-- ?? 0..* CodeableConcept Manifestation/symptom --></code>
```

```

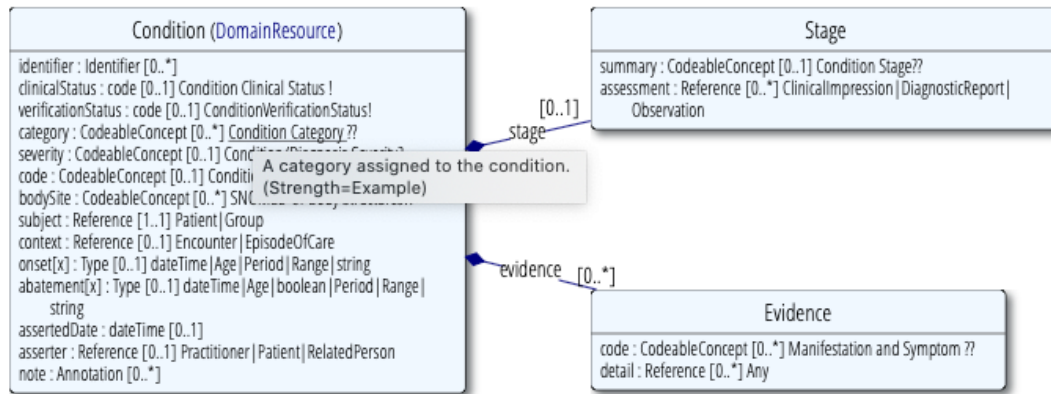
<detail><!-- ??? 0..* Reference(Any) Supporting information found elsewhere --></detail>

</evidence>

<note><!-- 0..* Annotation Additional information about the Condition --></note>

</Condition>

```



Medication

```

<Medication xmlns="http://hl7.org/fhir">

  <!-- from Resource: id, meta, implicitRules, and language -->

  <!-- from DomainResource: text, contained, extension, and modifierExtension -->

  <code><!-- 0..1 CodeableConcept Codes that identify this medication --></code>

  <status value="[code]"/><!-- 0..1 active | inactive | entered-in-error -->

  <isBrand value="[boolean]"/><!-- 0..1 True if a brand -->

  <isOverTheCounter value="[boolean]"/><!-- 0..1 True if medication does not require a prescription -->

  <manufacturer><!-- 0..1 Reference(Organization) Manufacturer of the item --></manufacturer>

  <form><!-- 0..1 CodeableConcept powder | tablets | capsule + --></form>

  <ingredient> <!-- 0..* Active or inactive ingredient -->

    <item[x]><!-- 1..1 CodeableConcept | Reference(Substance | Medication) The product contained --></item[x]>

    <isActive value="[boolean]"/><!-- 0..1 Active ingredient indicator -->

    <amount><!-- 0..1 Ratio Quantity of ingredient present --></amount>

  </ingredient>

  <package> <!-- 0..1 Details about packaged medications -->

    <container><!-- 0..1 CodeableConcept E.g. box, vial, blister-pack --></container>

    <content> <!-- 0..* What is in the package -->

      <item[x]><!-- 1..1 CodeableConcept | Reference(Medication) The item in the package --></item[x]>

      <amount><!-- 0..1 Quantity(SimpleQuantity) Quantity present in the package --></amount>

    </content>

```

```

<batch> <!-- 0..* Identifies a single production run -->

<lotNumber value="[string]"/><!-- 0..1 Identifier assigned to batch -->

<expirationDate value="[dateTime]"/><!-- 0..1 When batch will expire -->

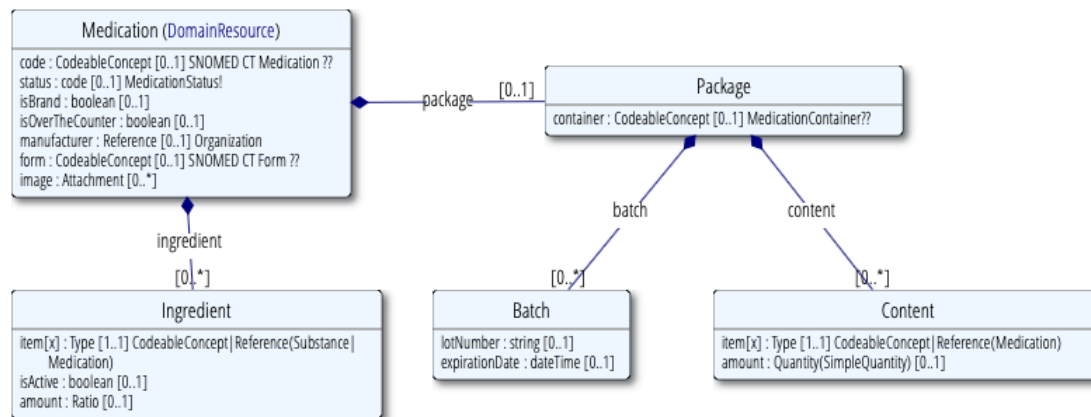
</batch>

</package>

<image><!-- 0..* Attachment Picture of the medication --></image>

</Medication>

```



Regarding the architecture, the main component of the FHIR interoperability is the Message Queue (MQ). The DS that needs to exchange non-IoT medical data with another DS or the Big Data platform, transmits the FHIR message to the MQ. Subsequently, the recipients who subscribe to the specific queue receive the message. In the current version of the SHAPES architecture and in order to utilise the components involved, the MQ deployed in the SymbloTe platform will be reused for the exchange of the FHIR messages.

6.2.5 Big Data Platform: Data Lakehouse & Analytics Engine

The Big Data Platform is composed of two main components: The Data Lakehouse and the Analytics Engine.

The Big Data Platform will process IoT data and medical (non-IoT) data coming from the different Digital Solutions (DS) via FInoT and FHIR connectors, respectively. Data exchanged with the Big Data Platform within the SHAPES project will be stored in the Data Lakehouse for the whole duration of the project, enabling Big Data Analysis both during and after the Pilots execution. Authorised Digital Solutions providers will have access to the output of the analysis made by the Analytics Engine, so these outputs be showed to the end-user through their DS interface. For commercial operation, the data will be stored in the Data Lakehouse only for the period of performing analytics.

The Big Data Platform will be a robust and scalable data processing infrastructure, working both in batch and streaming modes, and will provide a suitable place for data loading, transformation, and exploitation. This information will be used by the different Analytics Engines, to obtain enriched data and insights that will be later used by the

Digital Solutions (and presented to the end-users) or other authorised SHAPES partners (researchers) to further analyse the enriched data.

A **Data Lakehouse** is an emerging system that addresses the limitations of Data Lakehouses. A data Lakehouse combines the best elements of Data Lakehouses and data warehouses. Data Lakehouses are enabled by an innovative system design: they implement similar data structures and data management features to those in a data warehouse, directly on the kind of low-cost storage used for Data Lakehouses.

The main functionalities of the Data Lakehouse to be deployed in SHAPES are:

- ACID (Atomicity, Consistency, Isolation, Durability)⁴⁹ transaction support,
- robust governance and auditing mechanisms,
- storage is decoupled from computing,
- storage formats are open and standardized, such as Parquet⁵⁰,
- support unstructured and structured data,
- support for diverse workload, including SQL and analytics,
- facilitates GDPR (General Data Protection Regulation)⁵¹ compliance,
- and is possible to use real-time reports.

Therefore, the new Data Lakehouse paradigm is ideal to meet the requirements of SHAPES project. Regarding interoperability, the platform will include components that make it interoperable with the other SHAPES core components. In **Figure 28**, the workflow within the Big Data Platform is detailed. Regarding the architecture design in Figure 23 the connectors and the Message Broker are kept as a reference of the main SHAPES architecture, and the operation of the Big Data Platform is detailed.

Currently Tree Technology (TREE), partner leading the design and development of this Core component, deployed a first version of the Big Data Platform using Amazon Web Services (AWS). The Big Data Platform is deployed in Ireland and the hosting provider does not have access to the data available in the Platform. AWS provides security protection and encryption mechanisms. The data stored in the Big Data Platform will not have any identifiable/personal element, as the data is anonymised in each Digital Solution.

⁴⁹ <https://en.wikipedia.org/wiki/ACID>

⁵⁰ <https://parquet.apache.org/>

⁵¹ <https://gdpr.eu/>

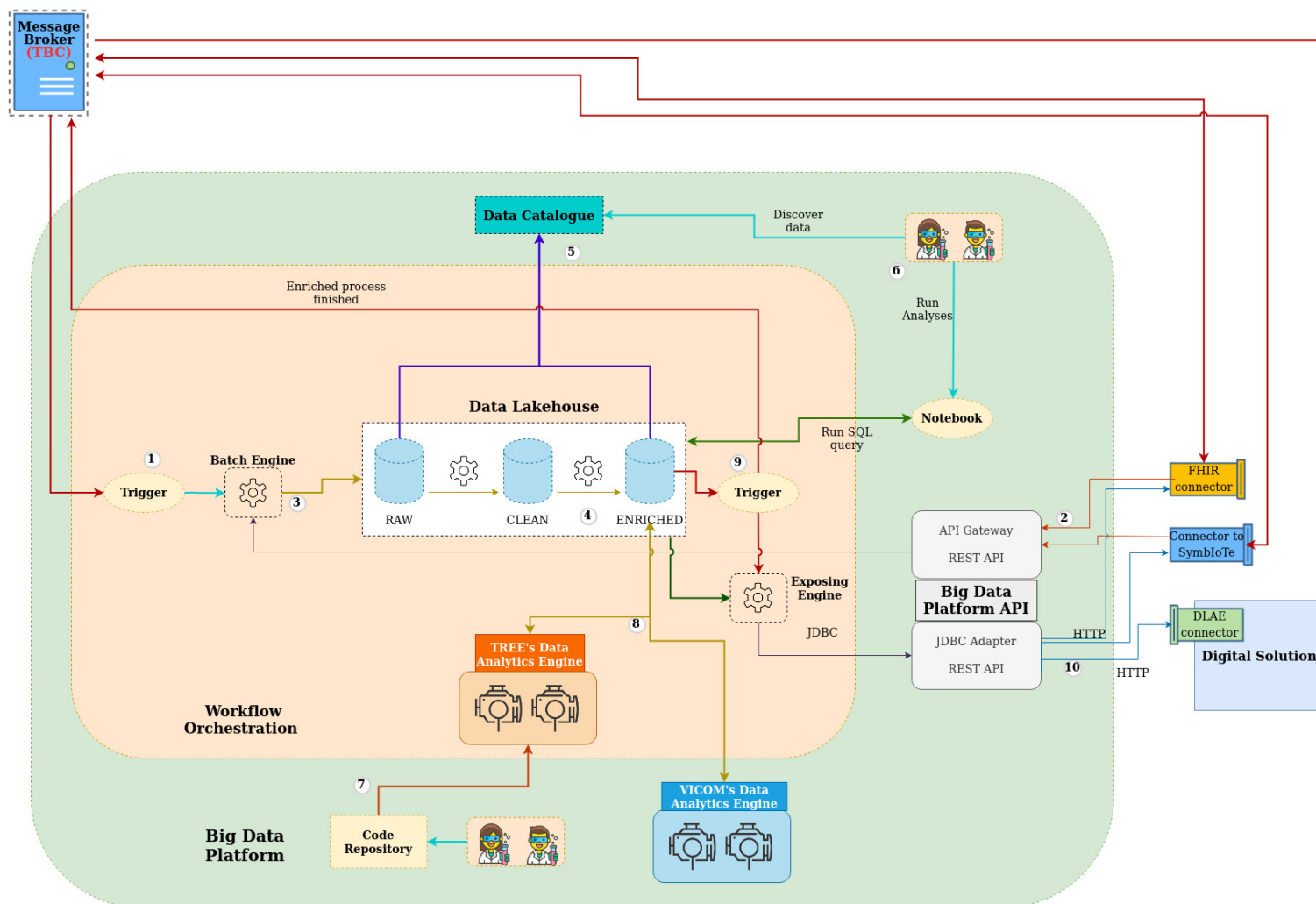


Figure 28: Internal structure of the Big Data Platform.

Within the Data Lakehouse, it is important to define an approach for data government. For this purpose, all stages of the data cycle in the Data Lakehouse are mapped, and more particularly, focussed on the raw and enriched data tiers (clean data tier is erased once it is used in later stages) – See point 4 in Figure 28. It is supposed that big data analytics will be done after the end of the SHAPES pilots, after gathering all the data and having it aggregated. Because of that, the raw and enriched data tiers need to be stored until the end of the SHAPES project, and so it is defined into the Data Lakehouse configuration.

In addition to the Data Lakehouse, there is a plan to include a tool to help cataloguing the data – Data Catalogue (marked with the number 5 in the diagram). This Data Catalogue will be of great help for the Data Scientists when developing the different data enrichment and data analytics algorithms. It makes it possible to exactly know what data is available and their description (through metadata management) and where it comes from, either from its origin or if was processed previously (and how many times), keeping track of data lineage. The Notebook complements this tool by facilitating access to information through data queries.

Another fundamental piece is the Workflow Orchestrator (coloured with light orange background in the diagram). This block is in charge of executing the different engines and actions within the Big Data platform, listening to the messages coming from the global Message Broker, and launching the adequate Data Pipeline, when necessary. It will also be responsible for sending a message back to the Message Broker when requested analytics results are ready.

In the following lines, a description of the different processes the data undergoes from its ingestion in the Big Data platform, to its consumption by the connectors is presented. The data ingestion starts with the notification by the SHAPES Core components through the Message Broker of the availability of data to be consumed. Trigger element (marked with the number 1 in the diagram) listens for messages coming from the general Message Broker. If a connector sends a message about the existence of new data, the trigger is launched, and the Orchestrator activates the process that extracts the data from either the FINoT or FHIR connector through the REST API (marked with the number 2 in the diagram) through the Batch Engine. This Batch Engine helps to store the data inside the Data Lakehouse (marked with number 3 in the diagram). Within the Data Lakehouse, different engines will normalize and clean the raw data before storing it in the Enriched layer (marked with the number 4 in the diagram). Once there, the data can be consumed by the Data Analytics Engines. On the other side, the raw and enriched data will be inventoried for proper organization and later querying. This is achieved by using the data cataloguing tool (marked with the number 5 in the diagram and introduced above). The Data Scientist can consult the data catalogue and use a web-based interactive code development notebook will be able to build models or develop statistics about the ingested and transformed information (marked with the number 6 in the diagram). This is also the starting point for building Data Analytics Engines and the code generated from the analytics is stored in a code repository. From this repository, analytics engines are deployed for their use in production (marked with the number 7 in the diagram).

One of the most important parts of the SHAPES project is that it can generate value from the collected data. The Data Analytics Engine extracts the enriched information and returns the results of advanced analytics, also storing them in the Enriched layer (marked with the number 8 in the diagram). After the results are stored and are available, another trigger is used to send a message to the Message broker (marked with number 9 in the diagram). When a Digital Solution detects the analysis is ready, they request this information via the Big Data Platform's REST API. Therefore, an Exposing Engine is invoked and using a JDBC connector, extracts results from the Data Lakehouse and they are sent to the connector via REST API via HTTP protocol (marked with the number 10 in the diagram). The analytical results and enriched data can be accessed by Digital Solutions as well as by FHIR and FInoT connectors.

It should be noted that, due to the needs of the project, TREE is developing a Big Data Platform API to facilitate communication between the Data Lakehouse and the different Digital Solutions, FHIR and FInoT platforms. All information exchanged is preceded by a message managed by the Message broker. This message highlights actions related to data consumption or the availability of results from Analytics Engines. In a first approach, the platform provides two types of messages: the first one indicates what data is consumed and from what source, and a second message indicates that a particular analytic is available to be consumed by one or more Digital Solutions. Besides, the triggers send an internal message to the orchestrator to start the processing and analytical engines.

Point	Keys Description
1	Trigger Trigger listens for messages coming from the Message Broker. If a connector sends a message about existence of data, it will be launched.
2	FInoT/FHIR connector The connector data is consumed via REST API through the "Batch Engine".
3	Batch Engine Store the data inside the Data Lakehouse (DLH) using the Batch Engine.
4	Data Lakehouse (DLH) Processes Within the DLH, different engines will normalize and clean the raw data before it is stored in the Enriched layer. Once there, the data can be consumed by the Data Analytics Engines.
5	Data Catalogue The raw and enriched data will be inventoried for proper organization and later querying by the Data Scientist.
6	Data Discover & Analyses

	Data Scientist consults the data catalogue and using a notebook will be able to build models or develop statistics about the ingested and transformed information. This is also the starting point for building DAEs.
7	Code Repository Code generated from the analytics is stored in a code repository. From this repository, analytics engines are built.
8	Data Analytics Engine (DAE) DAE extract the enriched information and returns the results, storing them in the "enriched layer".
9	Result Trigger Using a Trigger, once results from analytics are available, a message is sent to the Message broker. At the same time, an "Exposing Engine" is invoked.
10	Result to be consumed "Exposing Engine" (using a JDBC connector) extracts results from the DLH and they are sent to the connector via REST API (HTTP protocol).

Table 10 Key descriptions for Big Data Platform

6.2.5.1 Communication between the Big Data Platform and the other components

The FHIR Connector (I/F#5) enables the setup of a service to ingest non-IoT medical data into Big Data Platform API in a scalable, secure, and compliant manner. FHIR Connector accepts JSON-based messages sent out by an IoMT device and this data is persisted through the Big Data Platform API for FHIR. The data transformation logic is defined through an HTTP protocol that is configurable based on the message schema and FHIR requirements.

On the other hand, IoT data comes through the symbloTe connector from FINoT platform (I/F #3) or as temporary solution, directly via I/F #8. This data has also a JSON-based message format and comes from the different Digital Solutions available in SHAPES.

Integration work is currently ongoing by ICOM and FINT to deploy and integrate the symbloTe connector between symbloTe and FINoT (I/F#2). Once completed, the other core components will be able to integrate the same connector more easily. This approach has been considered in the various WP4 meetings as best practices for interoperability between IoT solutions and the core components, including the Big Data Platform.

In this case, and temporarily, the idea is to connect FINoT and Big Data Platform directly via I/F #8. Here, the Big Data Platform consumes data directly from the FINoT API, using an actor with HTTP protocol.

This is possible by building one or more data pipelines that consume information from FINoT via its API. As described above, these processes are executed through a trigger

that receives a message via the Message Broker. This trigger is waked up a process that will execute a data loading engine that, through proper authentication with the FInoT API, and it is extracted the data batches in a time frame.

Regarding the results from the Analytics Engines, Digital Solutions, FHIR and FInoT will consume these results through the Big Data Platform API using the HTTP protocol (I/F #7). This will be possible thanks to an HTTP client that we can see in Figure 28 as DLAE connector, which makes results accessible in JSON format.

6.2.5.2 Big Data Platform API details

As previously mentioned, the Big Data Platform API will facilitate the communication between the Big Data Platform and the Digital Solutions, FHIR and FInoT connectors. For its correct operation, the provisioned computing resources should be described in its configuration. To be able to access the API endpoint provided by the Big Data Platform API, the connection to the SHAPES Core Authentication system is needed. A simple procedure will be set up through which authorised consortium members can request access. In this case, the API acts as a bridge between the Big Data Platform running on Amazon Web Services and the different data providers (FInoT and FHIR) and analytical consumers (Digital Solutions, FHIR and FInoT).

Below, a description of the already defined resources and verbs provided by the Big Data Platform API are described. As the project will progress and other analytics are defined and/or more detailed, the API methods will evolve, and more details should be provided in the next deliverables.

Store:

- GET **/store**: Return the ids and name of all the stored source data
- POST **/store**: Save an object in the assigned store. Cannot overwrite existing objects
- PUT **/store**: Save or update an object in the assigned store
- GET **/store/{id}**: Return the document with the provided id
- DELETE **/store/{id}/{other_keys}**: Delete the document with the provided id; this verb will have a special permission

Data Lakehouse Query:

- GET **/query/meta/stores**: List of all available stores by Raw or Enriched tier
- GET **/query/meta/stores/{tier}/{digitalSolution}**: Return all the available stores by Digital Solution
- GET **/query/{tier}/data/{id}/{digitalSolution}**: Return the document with provided id

Data Analytics Query:

- GET **/listAnalytics**: List of the different analytics

- GET **/read/{dataAnalytics}**: Query the default store using time and other parameter by Data Analytics Solutions

Authentication APIs:

- POST **/auth/login**: A simple login which simplify starting a session
- GET **/auth/logout**: A facility method to logout
- GET **/auth/status**: Check the authentication status

Maintenance APIs:

- GET **/admin/logs**: Returns all the logs collected in the server
- GET **/admin/permissions**: Return the current permissions file
- GET **/admin/status**: Return current status information

6.2.5.3 Access to SHAPES datasets

For research purposes, some of the datasets (still to be defined by Pilot Themes) available in the Data Lakehouse will be available for the SHAPES partners, under authorisation. This access will be audited, to know who interacts with the information, when and by what means. This will ensure that data governance is enforced.

The access to the datasets will be done through the Big Data Platform API. Partner authentication will be ensured by ASAPA component and the Big Data Platform API will have a set of methods (to be defined) to allow partners' interaction with the data.

SHAPES consortium is considering making some of the results of the project available for external parties. When using a marketplace to make datasets available to third parties for research purposes, it must be considered that such data must be easy to handle and available in a standard format. Currently, the idea is to make the datasets available via the Big Data Platform API. In case the consortium goes ahead with this decision, TREE considers exposing the data from the Enriched layer of the Data Lakehouse in this marketplace. The information will come in CSV format with a description of the data that composes it, its type, and the Digital Solution of origin.

6.2.5.4 Big Data Platform Stack

For the deployment of the different components of the Big Data Platform, TREE will use Amazon Web Services' Big Data stack components⁵², which is based on open-source technologies. The aim is to take advantage of public cloud services for resource provision and management, while benefiting from the use of the big data stacks and tools offered by the open-source community. In Figure 29, the technologies proposed for the Big Data Platform deployment are depicted.

⁵² <https://aws.amazon.com/big-data/datalakes-and-analytics/>

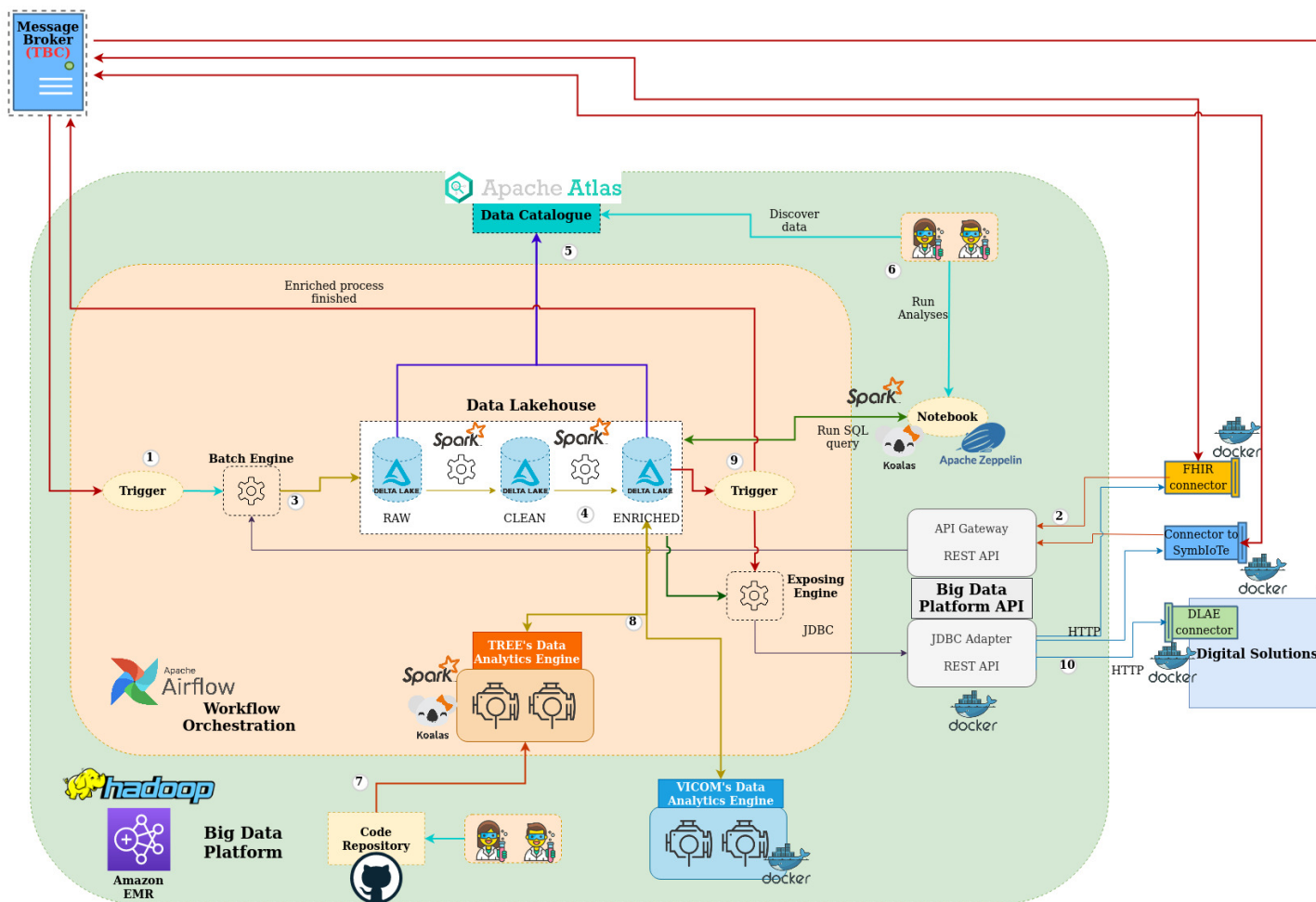


Figure 29: The Big Data Open-Source Stack.



Below, there is a brief description of the main technologies proposed:

- **Delta Lake** is an open-source storage layer that provides ACID transactions through optimal concurrency control between writes and snapshot isolation for consistent reads during writes. Delta also provides data versioning to facilitate rollbacks.
- **Amazon EMR** (Amazon Elastic MapReduce) is a management platform that allows the creation and management of Big Data clusters with the **Hadoop** and **Apache Spark** open-source ecosystem, warrant the distributed processing of large datasets across different instances. Amazon EMR has HDFS as the storage layer for the cluster.
- **Apache Atlas** is an open-source data governance tool that enables integration with the entire enterprise data ecosystem.
- **Apache Airflow** is a workflow manager, i.e., a tool that manages, monitors, and plans workflows, used as a service orchestrator.
- **Apache Zeppelin** is an implementation of the web notebook concept, focused on interactive data analytics using languages and technologies such as Apache Spark, among others.
- **Docker** is a software platform that allows the quick creation, testing and deployment applications, by packaging software into standardized units called containers.
- **Koalas** is an open-source Python package that implements the pandas API on top of Apache Spark, to make the pandas API scalable to big data.

6.2.6 Component: ASAPA Authentication and Authorisation

6.2.6.1 Purpose and Scope

The Authentication, Security and Privacy Assurance (ASAPA) component is a framework that provides a Single Sign-On (SSO) mechanism for centralized user-management and allows for the secure communication between internal components by providing a Public Key Infrastructure (PKI) for internal distribution of SSL certificates.

Moreover, the ASAPA component stores information concerning user-authorization, for Digital Solutions (DS) to utilize and enforce their authorization policies internally.

Finally, through a Security Assessment as a Service (SAaaS) module, existing and newly introduced entities are periodically assessed to define the overall risk they impose to the SHAPES ecosystem, in terms of vulnerabilities. Each assessed entity is flagged with the assessment score, based on the CVSS v3.1 standard.

6.2.6.2 Basic Use Case

Being the centralised authentication entity within the SHAPES ecosystem, ASAPA is a key component for the proper, safe, and secure functionality and interaction of internal services.

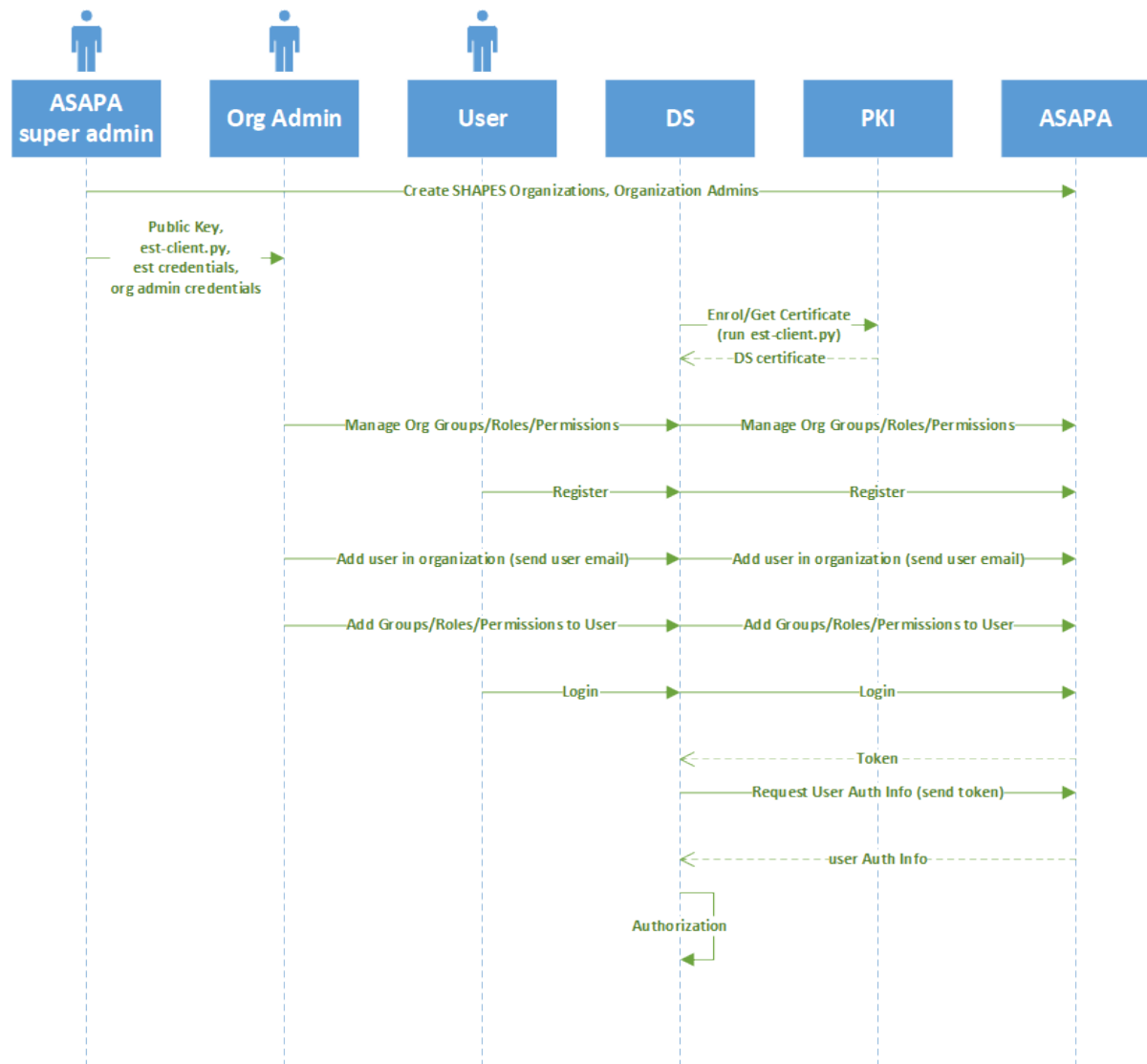


Figure 30 basic use case sequence diagram

The sequence in which actions take place, for the proper use of the ASAPA component is shown in Figure 30. Actions that each entity must take are as follows:

1. The ASAPA super admin, creates all the SHAPES organizations, and creates one organization admin for each organization.
2. The ASAPA administrator, distributes to all organization admins, the SHAPES public key, the est-client.py script, which will be used by Digital Solutions to enrol in the PKI and get their certificates, and the organization admin credentials.
3. Each DS enrolls in the PKI and retrieves its certificate, by running the est-client.py.

4. Organization admins, through their DS, manage their organization's groups/roles/permissions.
5. A SHAPES user, through a DS, registers an account.
6. Organization admins, through their DS, add a registered (existing) user in their organization, using the target user's email.
7. Organization admins, through their DS, assign groups/roles/permissions to a user belonging to their organization.
8. A registered user, through a DS, logs in.
9. The DS retrieves the user token, as a result of the user's login.
10. The DS requests and acquires the logged-in user's authorization information (groups/roles/permissions).
11. The DS enforces authorization policies on the user internally, based on the user's authorization info.

6.2.6.3 Component Architecture

The Figure 31 below illustrates the internal structure of the ASAPA component on a component level. There are three separate services the ASAPA component provides, namely the authentication mechanism, the PKI, and the SAaaS. The PKI will be utilized by all SHAPES entities, to allow secure inter-component communication within the SHAPES ecosystem. The authentication mechanism will be utilized as a Single Sign On mechanism, thus centralizing SHAPES user management. The functionality of the SAaaS will be detailed in the following subsections. For the SSO, several communication drivers will be developed, such as REST, COAP, etc. At the time this document is being authored, only the RESTful driver has been developed. Moreover, the Platform-Agnostic SEcurity Tokens⁵³ (PASETO) token have been utilized to provide secure, token-based authentication. Figure 32 below illustrates the high-level database schema, and the relationships between database entities. It is obvious that a user can belong in several realms and organizations, within which, different groups, roles and permissions exist. The framework's functionalities are separately detailed in the following subsections.

It must be pointed out that the ASAPA framework will only store user information concerning authentication, for the SSO functionality to utilise, and authorization for Digital Solutions and devices to store and manage, to enforce their authorization policies locally. All other information concerning the user is locally stored and maintained in each responsible entity (Digital Solution or device).

⁵³ <https://github.com/paragonie/paseto/>

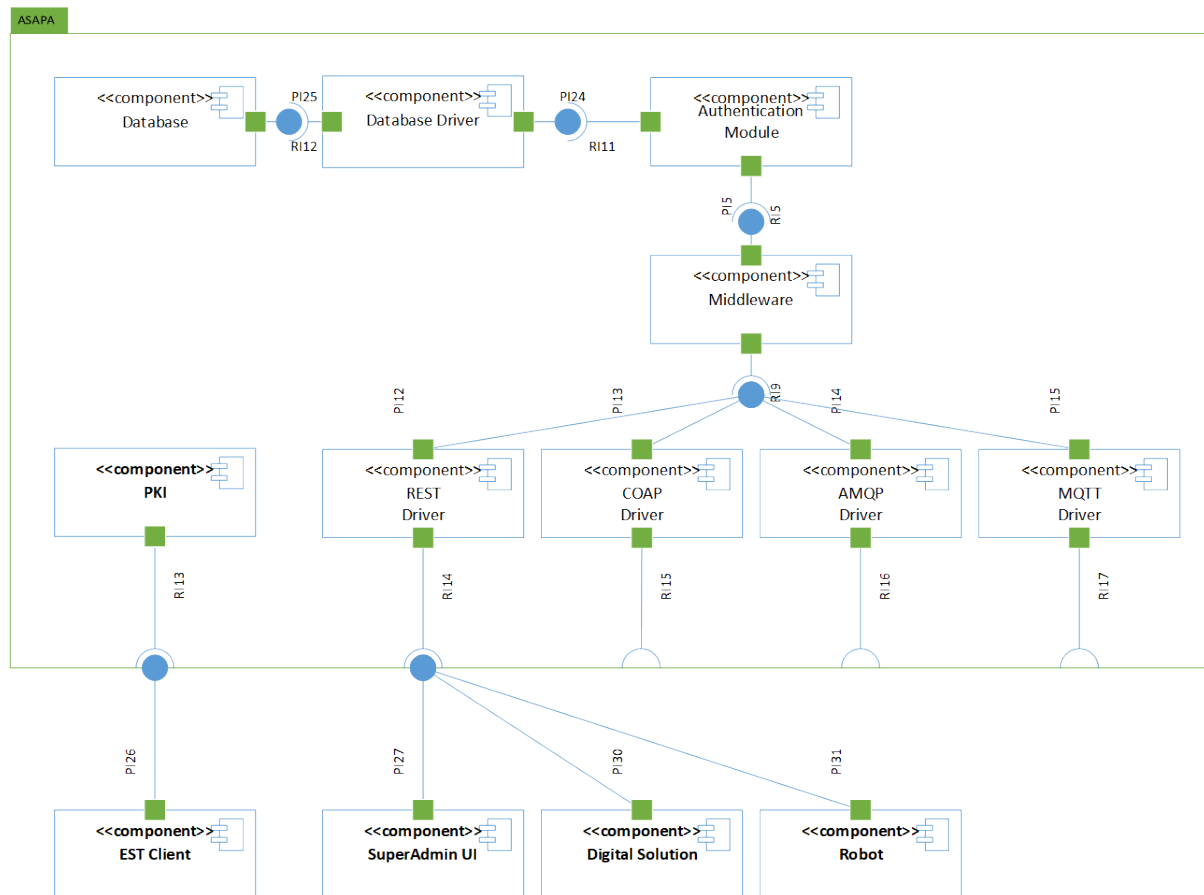


Figure 31 ASAPA component diagram

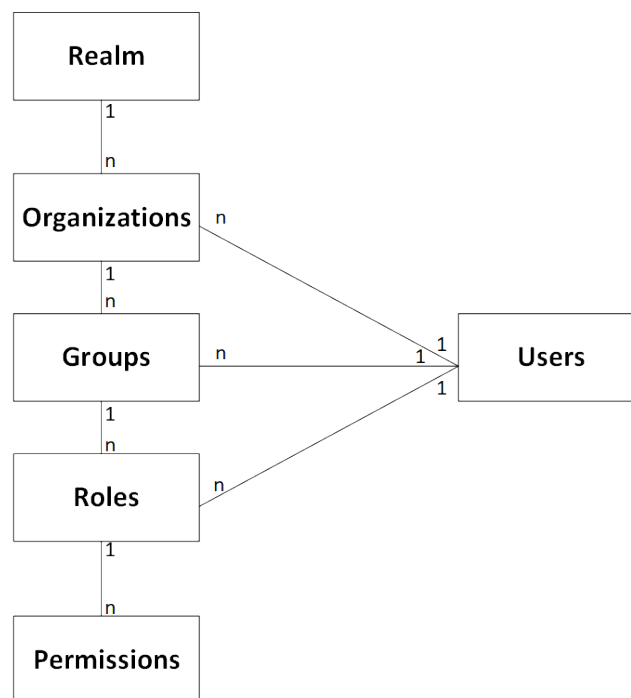


Figure 32 ASAPA database schema

6.2.6.4 Authentication

The authentication procedure is a rather straightforward workflow (Figure 33), where users utilize the available interfaces to perform all provided functions (e.g., login, logout, register). The ASAPA component currently exposes a RESTful API, through which every entity requiring user authentication can utilize and perform all the required tasks (e.g., login, register, update user, etc.). The relevant endpoints are detailed in the table below (Table 11).

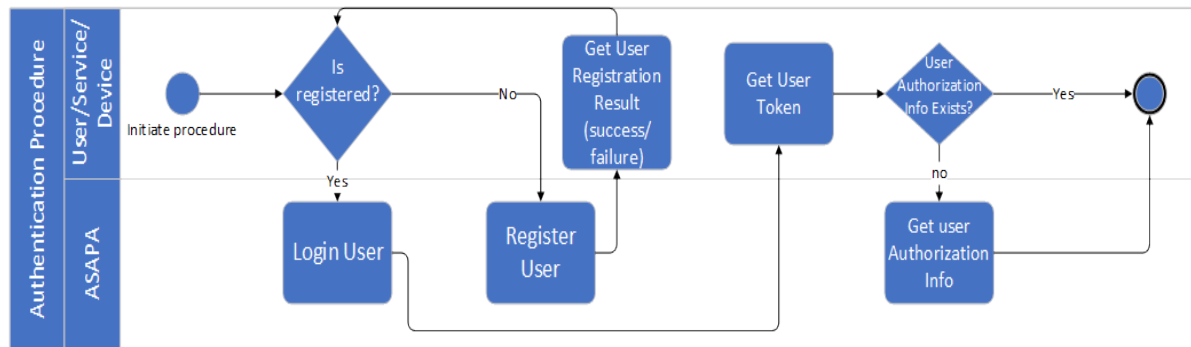


Figure 33 Authentication procedure

PATH	METHODS	HTTP	EXPECTED RESPONSE	COMMENTS
/auth/login	POST	BODY: {“email”: string, “password”: string}	A JSON response containing in the items field the token of the user and the expiration seconds	
/auth/logout	POST	HEADERS: X-Pasiphae-Auth: token	A JSON response containing a corresponding message and code	Expired tokens will return 403 Unauthorized Access
/auth/register	POST	BODY: {“email”: string, “password”: string}	A JSON response containing a corresponding message and code	*Passwords must be hashed
/auth/refreshToken	GET	HEADERS: X-Pasiphae-Auth: token	A JSON response containing a corresponding message and code with the new token in items field	

/users/me	GET	HEADERS: X-Pasiphae-Auth: token	A JSON response containing the information of the user and RBAC information in hierarchical view	
-----------	-----	---	--	--

Table 11 Authentication REST API endpoints

6.2.6.5 Security

The security functionality is provided by allowing all internal components to communicate over secure, encrypted channels, by utilizing SSL certificates, distributed through a Public Key Infrastructure (PKI). The procedure for entities (users, services, etc.) to acquire a certificate is illustrated in the following workflow diagram (Figure 34). A dedicated client software (est-client.py) will be distributed to partners to enrol in the PKI, and retrieve their certificates, over the EST protocol (RFC7030⁵⁴).

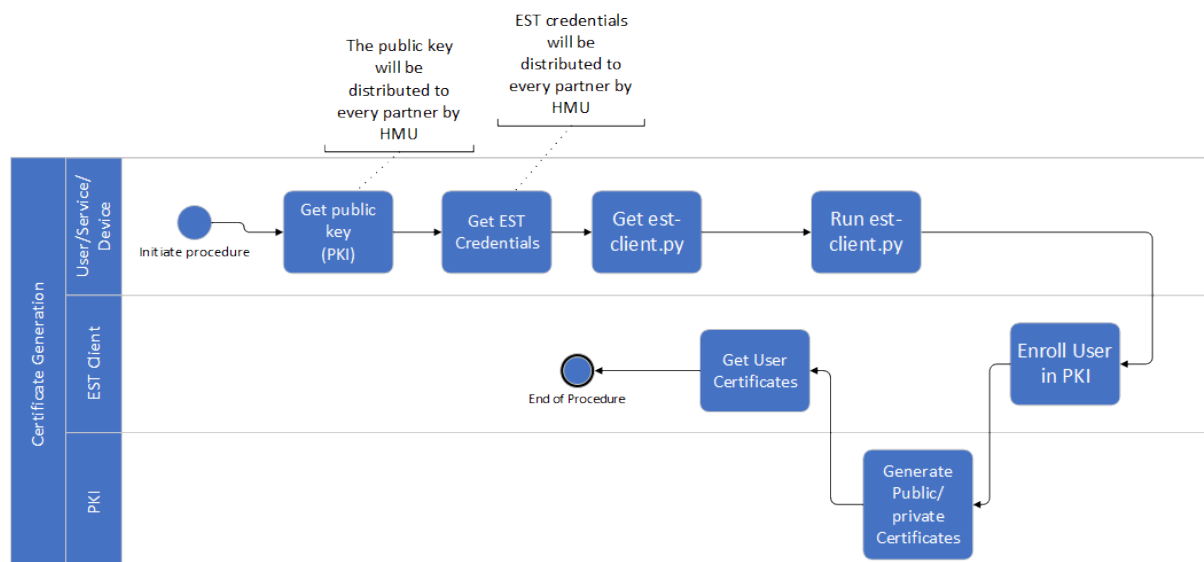


Figure 34 Certificate generation

Furthermore, to fortify the security aspect, the ASAPA framework provides a Security Assessment as a Service (SAaaS) component. The SAaaS component periodically scans the underlying network, to discover newly introduced network-enabled entities (IP/MAC address). Consequently, it assesses them to discover, if any, inherent vulnerabilities they might impose to the SHAPES ecosystem, due to ill-maintained services (older versions of software, carry well-known vulnerabilities). Administrators can manage discovered entities, assessment tasks and their assessment reports through a simple web-based user interface (dashboard). The Figure 35 illustrates the

⁵⁴ <https://tools.ietf.org/html/rfc7030>

internal architecture of the SAaaS component, on a component level. Finally, the SAaaS component exposes a RESTful API, for fine-grained programmatic manipulation. The Table 12 details the RESTful API's exposed endpoints for the SAaaS component.

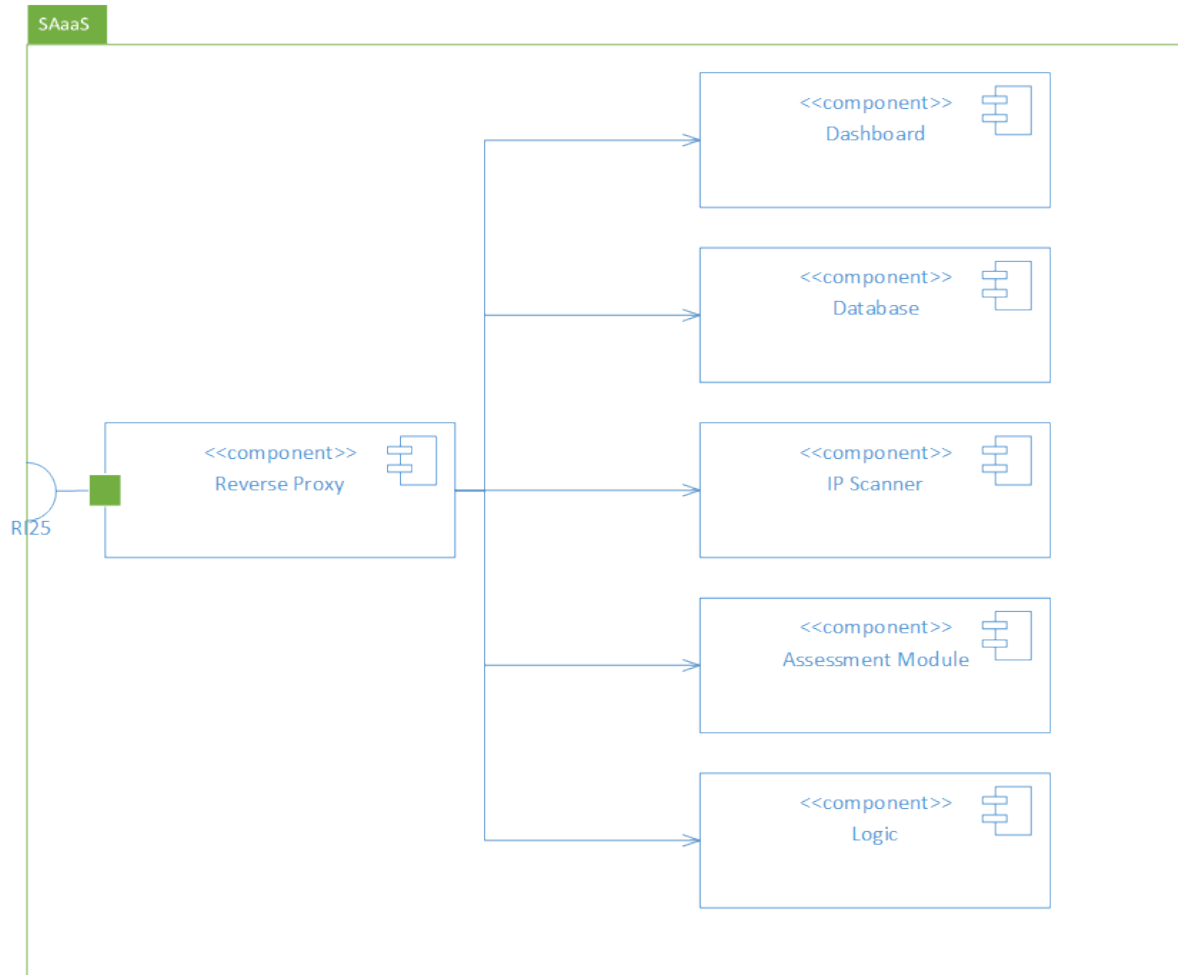


Figure 35 SAaaS component diagram

PATH	METHODS	HTTP	EXPECTED RESPONSE	COMMENTS
/health	GET		{“result”:“ This is the vulnerability assessment as a service component”}	API health check
/tasks	GET		{“result”:“ GET_ALL_TASKS_SUCCESS”, “items”: {“tasks”: <list_of_tasks>}}	Returns the list of all assessment tasks
/tasks	POST	BODY: {“name”: string (e.g., “192.168.1.1”}	{“result”:“ GET_TASK_SUCCESS”, “items”: {“task”: <task>}}	Returns a specific task

/tasks/start	POST	BODY:{"target":string (e.g., "192.168.1.1", "name":string (e.g., "test"), "speed":number [1-5] (e.g., 3)}	{"result": "SCAN_NETWORK_STARTED", "more": "Started assessment for <target>"}	Starts a new assessment task for a specified IP address, in a specified speed (1-5)
/tasks/progress	POST	BODY:{"name":string (e.g., "192.168.1.1")}	{"result": "GET_TASK_PROGRESS_SUCCESS", "items": {"processes": <task_processes>}}	Returns the progress for a specific task
/tasks/<task_name:string>	DELETE		{"result": "DELETE_TASK_SUCCESS", "more": "Task with task name: <task_name>, was successfully deleted"}	Deletes a specific task
/reports	GET		{"result": "GET_ALL_REPORTS_SUCCESS", "items": {"reports": <list_of_reports>}}	Returns all existing reports
/reports	POST	BODY:{"name":string (e.g., "192.168.1.1")}	{"result": "GET_TASK_REPORTS_SUCCESS", "items": {"reports": <list_of_reports>}}	Returns all existing reports for a specific task

Table 12 SAaaS REST API

6.2.6.6 Privacy Assurance

Finally, the ASAPA component will store users' authorization information, for system-wide authorization policy enforcement by Digital Solutions and devices. Entities can add, retrieve, and update users' authorization information by utilizing the provided interfaces (e.g., RESTful API). The available endpoints of the provided API are detailed in the table below (Table 13) and authentication process in Figure 36.

PATH	METHODS	HTTP	EXPECTED RESPONSE	COMMENTS
/users/<id>/organizations	GET, PUT	PARAMETERS: User id BODY: {"id": string, "organizations": [string]}	A JSON response containing information regarding the status of the update or in GET method, the organizations of the user	In case of updating (PUT), the request must include the complete list of organizations

/users/<id>/groups	GET, PUT	PARAMETERS: User id BODY: {"id": string, "groups": [string]}	A JSON response containing information regarding the status of the update or in GET method, the groups of the user	In case of updating (PUT), the request must include the complete list of groups
/users/<id>/roles	GET, PUT	PARAMETERS: User id BODY: {"id": string, "roles": [string]}	A JSON response containing information regarding the status of the update or in GET method, the roles of the user	In case of updating (PUT), the request must include the complete list of roles
/roles/<id>/permissions	GET, PUT	PARAMETERS: Role id BODY: {"id": string, [{"permission": string, "action": string, "resource": string}]}	A JSON response containing information regarding the status of the update or in GET method, the permissions of the role	In case of updating (PUT), the request must include the complete list of permissions
/groups/<id>/roles	GET, PUT	PARAMETERS: Group id BODY: {"id": string, [{"role": string}]}	A JSON response containing information regarding the status of the update or in GET method, the roles of the group	In case of updating (PUT), the request must include the complete list of roles

Table 13 User authorization information API endpoints

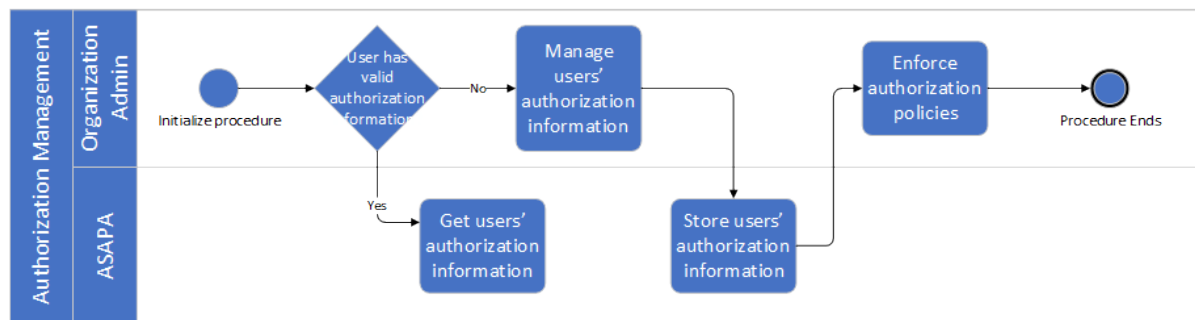


Figure 36 Authorization management

6.2.6.7 Component Deployment and Prerequisites

The ASAPA framework will be deployed publicly and will expose its API endpoints. For SHAPES entities to utilise the framework, several prerequisites must be initially met, namely:

1. Firstly, entities must obtain the SHAPES public key (same for all entities), and the est-client.py script's credentials (same for all entities).
2. Acquire the est-client.py, with which entities will enrol with the PKI and retrieve their personal certificates.
3. Integrate the SSO API to utilise the provided authentication functionalities.
4. Integrate the user authentication information API, to centrally store and manage users' authorization info.

6.2.7 Component: Message Broker

RabbitMQ⁵⁵ is a general-purpose message broker that provides a common platform to applications to send and receive messages, by defining queues to which applications can connect in order to transfer a message or messages. RabbitMQ offers rich routing capabilities, employing point to point, request/reply and publish/subscribe communication styles patterns. It is lightweight and easy to deploy, either on premises or in the cloud, while different distributed and federated configurations can be used to meet large-scale and high-availability requirements. RabbitMQ is supported on a number of operating systems and provides official client libraries, with the RabbitMQ community having created numerous clients, adaptors and tools. The RabbitMQ management plugin provides an HTTP-based API for the management and monitoring of RabbitMQ nodes and clusters and can, also, be configured to use different setups and mechanism like HTTPS, OAuth 2, cross-origin resource sharing and others.

A message broker receives messages from publishers (i.e., applications that publish the messages), also known as producers, and routes them to consumers (i.e., applications that process the messages). Message queues consist the backbone of the system and are essentially buffers, where messages are stored. The message broker (i.e., RabbitMQ) ensures that the messages from a publisher go to the right

⁵⁵ RabbitMQ message broker: <https://www.rabbitmq.com>

consumers. A messaging protocol is required to deal with publishers and consumers and enable conforming client applications to communicate with the conforming messaging middleware brokers. RabbitMQ supports multiple messaging protocols, directly or through the use of plugins. Among them, AMQP (Advanced Message Queuing Protocol)⁵⁶ is commonly used as the standard to control the entire message passing process.

In the RabbitMQ architecture, messages are not published directly to a queue, but to an exchange. Thus, a producer can only push messages to exchanges. An exchange is basically a message routing agent that on one side receives messages from producers and on the other side pushes them to different queues based on routing rules (bindings). The routing algorithm used depends on the exchange type and the routing rules. Different types of exchanges require different bindings. Then, the broker either delivers messages to consumers subscribed to queues, or the consumers fetch/pull messages from queues on demand.

In the context of the SHAPES project, the AMQP standard messaging protocol will be used with RabbitMQ as the message queue server for internal messaging between the SHAPES core components. Each component will be implementing a messaging solution, using the AMQP compliant framework (e.g. RabbitMQ) to be able to exchange information and events with other SHAPES core components.

The details of the asynchronous communication for the messaging between the SHAPES core components will be finalized in the next period of the project and the relevant interfaces (RabbitMQ topics) and their description (such as exchange, routing keys, payload) will be finalized and presented in detail in the next deliverable.

6.2.8 Component: Front-end App

SHAPES provides a plethora of Digital Solutions developed by different partners with the common goal of improving the quality of life of older individuals, by maintaining and extending their independence and autonomy. To enable the largest possible number of Digital Solutions working together in support of the piloting activities, the SHAPES Consortium devised the SHAPES Front-end App as an archetype, to be used by all pilot use cases, as a unique access point to the SHAPES Digital Solutions ecosystem. Also, because of the intrinsic idiosyncrasies of the main SHAPES user type (older individuals) and of the likelihood that most SHAPES users would have low technological skills and could exhibit difficulty interacting with novel technologies, it became clear that a simple user interface providing a centralised access to the SHAPES Digital Solutions would be the right architectural approach to sustain higher levels of user experience and acceptance.

As a result, the SHAPES Front-end App became a new SHAPES development, proposed by EDGENEERING, with the clear purpose of facilitating the users' interaction with the different SHAPES Digital Solutions, namely those running on Apps either on smartphones or tablets (devices). Overall, the SHAPES Front-end App aims:

⁵⁶ <https://www.amqp.org>

- To represent the main entry point to the SHAPES Digital Solutions running on smartphones or tablets.
- To provide a mechanism for the single authentication of the user in the SHAPES Platform (as opposed to requiring the user to authenticate when accessing each Digital Solution).
- To deliver a single point of access for the various SHAPES Digital Solutions relevant for specific pilots and deployments. Where possible, a simplified access (single-click) to specific Digital Solutions features is envisaged, to minimise the users' challenges navigating the Digital Solutions.

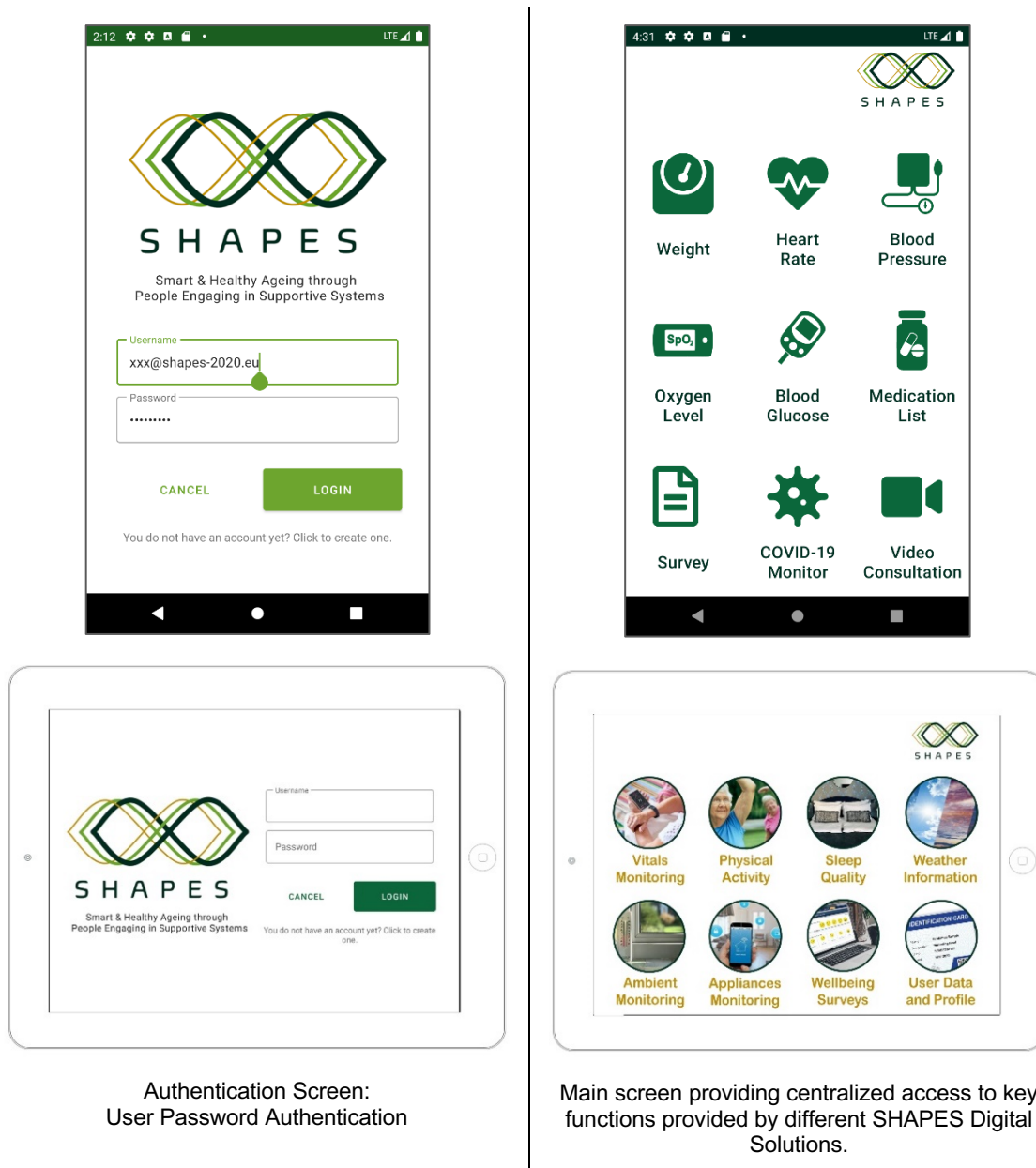


Figure 37 - SHAPES Front-end App: Snapshots

The SHAPES Front-end App makes use of the SHAPES Single Sign-On (SSO) mechanism (see section 5.2.6) to authenticate the user in the SHAPES Platform. The

authentication process may be based on user password or adopt other user-friendly mechanisms, including biometrics (face recognition and fingerprint). Following the user's successful authentication, the SHAPES Front-end App grants access to the different SHAPES Digital Solutions installed in the device and the user can benefit from their many functionalities.

The design of SHAPES Front-end App (shown in Figure 37) follows the visual identity and branding of the SHAPES project, namely its logo and colour theme. Below, it is provided the snapshots of the two screens that comprise the SHAPES Front-end App. It should be noted that the second screen will be adapted to the specific requirements of each pilot use case (these adaptations and the SHAPES Front-end App as a Digital Solution per se will be documented in deliverable D5.3). The SHAPES Front-end App provides an easy to use interface allowing the user to identify and authenticate in the SHAPES Platform (i.e., username and authentication token) to access all SHAPES Digital Solutions installed in the device. Upon user request (click the icon), the SHAPES Front-end App launches a functionality provided by a specific SHAPES Digital Solution. The below diagram illustrates the interaction of the SHAPES Front-end App with other SHAPES components.

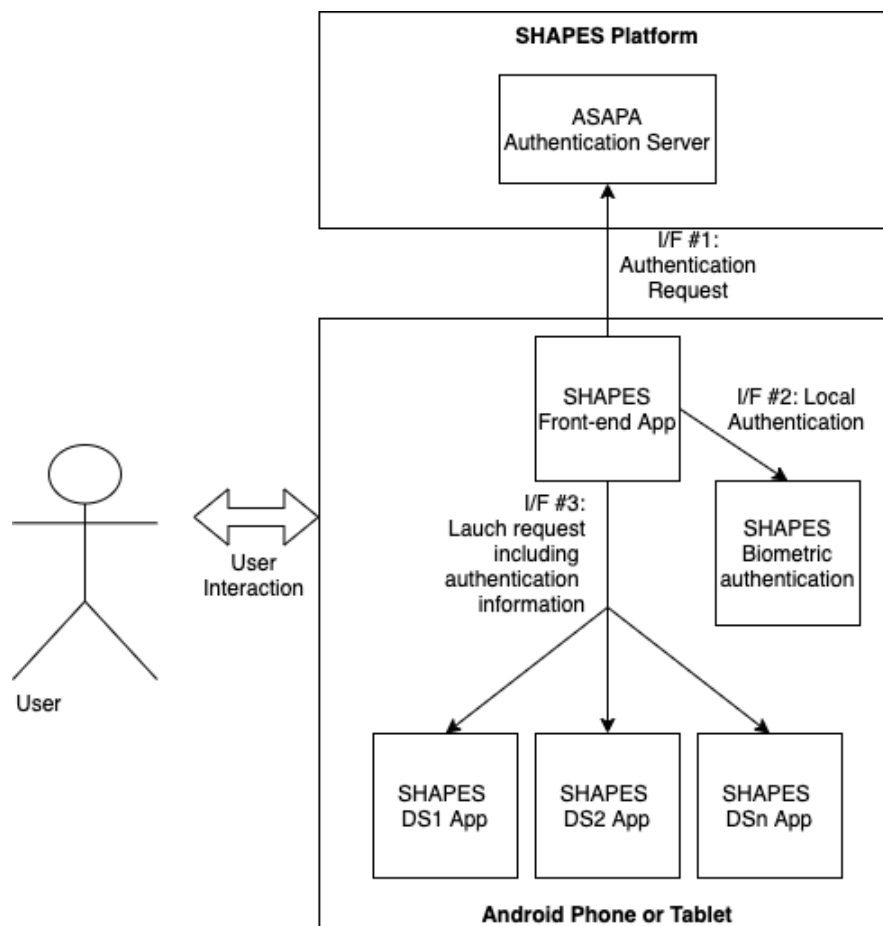


Figure 38 SHAPES Front-end App: Interfaces

The SHAPES Front-end App requires user interaction to initiate specific actions.

- **I/F #1: Interface between the SHAPES Front-end App and the ASAPA.** It is used to authenticate a user in the SHAPES Platform and SHAPES Digital Solutions. The SHAPES Front-end App sends an authentication request to ASAPA and receives back an authentication *token* (see 6.2.6.4 Authentication).
- **I/F #2: Interface between the SHAPES Front-end App and the SHAPES Biometrics Authentication.** It is used to authenticate the user applying biometrics (e.g., face or fingerprint recognition). Via the SHAPES Front-end App, the user can initiate the biometric authentication process and receives a positive or a negative confirmation. Biometric authentication is based on FACECOG (see section 3.2).
- **I/F #3: Interface between the SHAPES Front-end App and the SHAPES Digital Solutions.** It is used by the SHAPES Front-end App to launch a SHAPES Digital Solution, as selected by the user. As part of the launch process, information concerning the username and the authentication token is passed to the Digital Solution.

The SHAPES Front-end App runs on Android devices above Android 5.0 (API 21).

The open source code of the SHAPES Front-end App is available in GitHub (<https://github.com/SHAPES-H2020/Front-end-App>) allowing a collaborative development by technical SHAPES partners and ensuring the right connection to Digital Solutions and specific functionalities, as they support SHAPES piloting activities.

NOTE: The SHAPES Front-end App described in this section focuses on the integration aspects for other SHAPES DS running as Android Apps. The Front-end App as a unique access point to the plethora of SHAPES DS installed in a mobile phone and its adaptation to each pilot use case will be further described in D5.3.

6.2.9 Add-on Service: Adilib Chatbot Building Platform

Adilib provide a platform to build, train, deploy and serve Chatbots. Adilib has a web that allows developers to perform these operations.

Functionalities:

Adilib has the following main functionalities:

- **Train Natural Language Understanding models:** Adilib can be used to create & tag examples using intents and entities to train NLU models. These models will be used by the Chatbot to extract the semantic meaning of the users' queries.
- **Manage Dialogue Memory:** Adilib offers functionalities to define the dialog memory, transitive elements that are stored turn by turn to give more flexibility to the dialog. The dialog memory is modified using rules that can be entered from a web interface.
- **Create Interaction Rules:** Adilib allows you to create interaction rules to determine the next response of the Chatbot with different priority levels.
- **Train and manage several Chatbots:** Adilib's web platform allows training AI models and maintaining a versioning of the Chatbot models.
- **Deploy Chatbots:** Adilib allows to deploy Chatbot models so that they can interact with users through WS channels. Being prepared to manage multiple Chatbots, Adilib allows serving N Chatbots from a single deployment.

- **Monitorization:** All the data collected from the conversations are stored in an ElasticSearch database, on which Kibana is deployed. Kibana enables the monitoring of the conversations being had with the Chatbots, as well as defining graphs and visualizations with various KPIs.

Interacting with the Chatbots:

The only exposed endpoint for the chatbots of Adilib is accessed via WS, which receives the following JSON:

```

1  {
2    "dialogid": string, // Identifier of the dialogue
3    "message": string, // Message from the user
4    "language": string, // Language identifier (optional)
5    "userid": string, // User identifier (optional)
6    "additionalinfo": {}, // Additional info (optional, interface)
7  }

```

Figure 39 JSON schema of the input message

Adilib will send the following response to an input message:

```

1  {
2    "dialogid": string, // Dialogue ID
3    "message": [], // Response Messages
4    "language": "eu",
5    "nluinformation": {}, // NLU Response
6    "dminformation": {
7      "semantic-response-list": [], // System response actions
8      "dm-in-out": {
9        "given-output": [], // list of response actions
10       "received-input": {} // Structured NLU input
11     },
12     "dialogue-info": {
13       "dialogue-state-json": {}, // Dialogue State
14       "language": "",
15       "user-id": "" // User ID
16     }
17   },
18   "starttime": "2021-03-03T07:41:09.642452022Z",
19   "endtime": "2021-03-03T07:41:09.731832382Z"
20 }

```

Figure 40 Adilib's response scheme

With this JSON, one can get the results of the Natural Language Understanding module, the current Dialogue State and so on. Additional information about this WS channel will be provided in the technical documentation.

Maintaining a dialogue

The key to maintaining a conversation with the assistant is to keep the "dialogid" throughout the interactions.

1. In the first interaction send an empty string ("dialogid": "")
2. Adilib will generate a unique identifier for the current dialog/session.
3. In the following calls, send this identifier in the field "dialogid".

Keeping websocket open

To keep the websocket open, a simple system of calls must be implemented.

Every 30 seconds (approximately) send the string "ping" to the WS. The WS will reply with a "pong" on return, a message to be filtered.

Building Custom Connectors

If an external system wants to deploy the Chatbots made with Adilib, an appropriate front-end that implements this communication mechanism needs to be implemented.

Where it is deployed

Adilib will be served as a Software as a Service on an OVH cloud, based in France. Each Chatbot has a channel accessible via WS communication using a randomized hash as an endpoint.

6.3 Marketplace

The SHAPES marketplace will strongly contribute to the foreseen “market shaping”, by seamlessly connecting supply with demand. It will host a detailed portfolio of Digital Solutions, provided by the SHAPES consortium, open to third parties, for the active and healthy ageing, independent living and integrated care markets. Moreover, the SHAPES marketplace will offer informational resources, best practices, tutorials, and educational material, thus helping the health and care community to benefit from the project’s outcomes and solutions. The marketplace aims to become an additional revenue stream for SHAPES, as well as to transform SHAPES into a future-proven platform, as it will allow for the continuous addition of services and solutions, throughout the project’s lifecycle and thereafter.

As part of the Marketplace, SciFY will develop a collection of digital interventions (games and cognitive training apps) that will be used in context of several piloting use cases of SHAPES. Additionally, a set of easy-to-use resources and tutorials will be created and offered as part of the SHAPES Marketplace (game resources and tutorials, non-pharmaceutical activities for the elderly, etc). Among the Digital Solutions offered by SciFY, two will also be integrated into other technologies offered by other technical partners:

- **“Memor-i” game:** is a digital adaptation of the classic memory card game. SciFY will translate it into 3 languages, and offer it as an integrated solution of the ARI robot, by PAL. “Memor-i” will then make use of the text-to-speech REST API of the robot, in order to communicate with the users.
- **“DiAnoia” Android app:** is an Android app that offers a wide collection of non-pharmaceutical interventions, for people with mild dementia. This Digital Solution will also be offered in 3 languages, and new content will be created. In the context of the SHAPES piloting activities. The application will run on an Android tablet that

will be held in the back pocket of the ARI robot (PAL). Once running, the application will make use of the text-to-speech REST API of the robot, in order to communicate with the users.

NOTE: since the respective “Task 7.4: SHAPES Marketplace” will start on M19 i.e. after the submission of this deliverable, this section will be populated with relevant content and will include additional solutions once this work has started.

7 Integration Strategy

The micro services architecture will be used for the design and development of the SHAPES platform to enable independent development, maintenance and deployment of each individual component of its core architecture (i.e., the SHAPES core components). The micro services architecture allows to structure an application as a collection of autonomous but loosely coupled services which are able to work together. This section describes the integration approach that will be used for the development and deployment of the SHAPES technical platform.

7.1 Components integration

Asynchronous communication between SHAPES core components will be supported using the message broker, described in the previous chapter, section 6.2.7, while RESTful APIs will be used to expose services provided by the SHAPES core components.

Swagger⁵⁷ is an open-source tool that provides a formal format to develop and document RESTful APIs and can be used in the context of the SHAPES project to define the RESTful APIs that SHAPES core components will expose. The OpenAPI Specification (currently in version 3.0.3⁵⁸) defines a standard, language-agnostic interface to RESTful APIs in a way that both humans and computers can understand the RESTful services and their functionalities, allowing to describe the entire API including available endpoints and operations on each endpoint (GET, POST etc.)

Finally, there are different serialization formats to be used for the transmission of data between applications. Among them, JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications, as it is human readable and lightweight and can be used for the SHAPES project needs.

7.2 Tool chain overview

As depicted in the following table, the GitHub and Docker tools are required to facilitate the implementation workflow and support incremental feature deployment, maintenance and scalability of the highly distributed SHAPES ecosystem.

	Version control	Code repo	Container MGT	Image repo
Tool	Git	GitHub	Docker/Docker Swarm	DockerHub

Table 14: Tool chain overview

⁵⁷ Swagger: <https://swagger.io/>

⁵⁸ Open API spec v3.0.3: <https://swagger.io/specification>

7.2.1 Version control system

Git is an open-source distributed version control system that allows developers to collaborate, manage and track changes of the source code. It allows developers to work in parallel, by using a local copy of the repository at their working environment and committing their changes to it. When ready, local changes can be pushed onto a remote repository (the git server), where other developers can see the changes and pull them to update their local repository. Also, multiple local branches are supported that can enable the parallel development of features, as well as the management of product releases. Git will be used in the SHAPES project to enable its developers to work in parallel and collaborate on the source code for the development and maintenance of SHAPES core components. Two main branches will be maintained to facilitate the integration and release process; a master branch pointing to the latest source code in production-ready state (i.e., latest release version) and a develop branch pointing to the latest development changes to be delivered for a next release.

7.2.2 Code repository

GitHub is a cloud-based Git repository hosting service for open-source projects. It will be used to host and make available the source code of the SHAPES project. GitHub public repositories and free accounts are used. More specifically, a remote GitHub repository has been set at: <https://github.com/SHAPES-H2020>, shown in Figure 41.

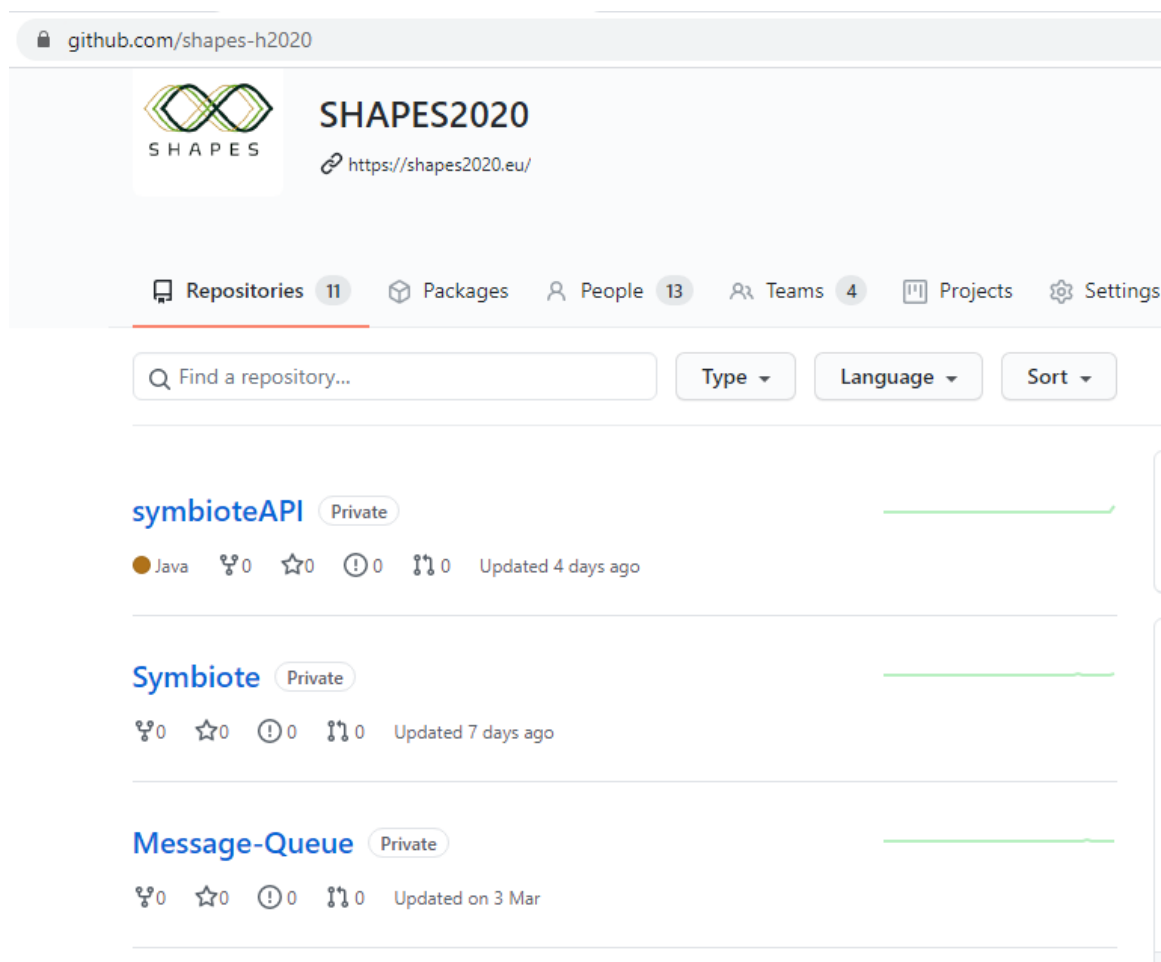


Figure 41 SHAPES GitHub repository

To reflect the micro service architecture described earlier, a multi-repository setup was used with the SHAPES core components being bundled into GitHub super-repositories (Figure 41). In that way, independent development of the software components can be achieved with clear ownership. Also, the build is faster due to the smaller code base, as developers need to download and build only the repositories they use. Project members will be able to download the latest versions, fetch changes from other members, implement their features locally and commit their changes back to the shared repository. For each repository, a main contact is assigned administrator rights in order to be able to manage their corresponding repository, create projects related to their component, add team developers, and organise their code base, as needed. Finally, GitHub provides access control and collaboration features such as bug tracking, feature requests and task management, which can be utilised during the development phase in the project.

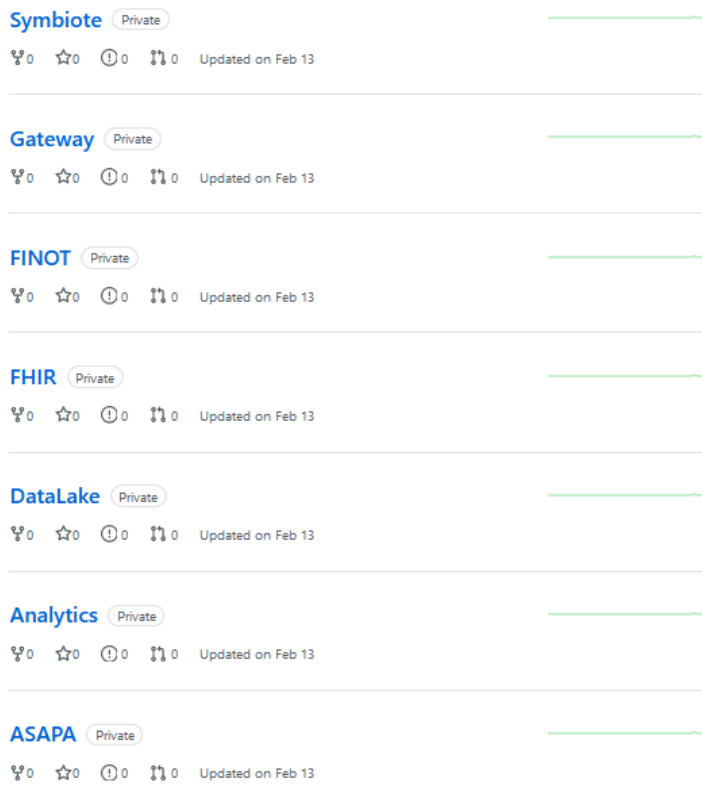


Figure 42 Snapshot of the SHAPES GitHub repositories

7.2.3 Container management

Containerisation is a form of operating system virtualisation that allows to package applications and their dependencies in containers (i.e., loosely isolated environments) to support their secure and isolated execution on a same shared operating system and hardware. It allows to run applications quickly and reliably in different computing environments, independently of the underlying hardware and operating system. Applications are encapsulated in Docker container images, executable packages of

software that includes everything required to run the application, such as code, runtime, libraries and settings. Container images become containers at runtime.

Each SHAPES core component will be encapsulated in a container image to be deployed independently in the form of a Docker container. The consortium partners that own SHAPES core components will provide the configuration files required to build the Docker images for the deployment of their components, such as Dockerfiles or Docker Compose files for more complex components.

7.2.4 Image repository

DockerHub⁵⁹ is a service provided by Docker for discovering and sharing container images within a team. It comprises the largest repository of container images from container community developers, open-source projects and independent software vendors building and distributing their code in containers. Users can get access to free public repositories for storing and sharing their container images. DockerHub will be used to store and share container images for the SHAPES project needs, and can be found at <https://hub.docker.com/u/shapes2020>.

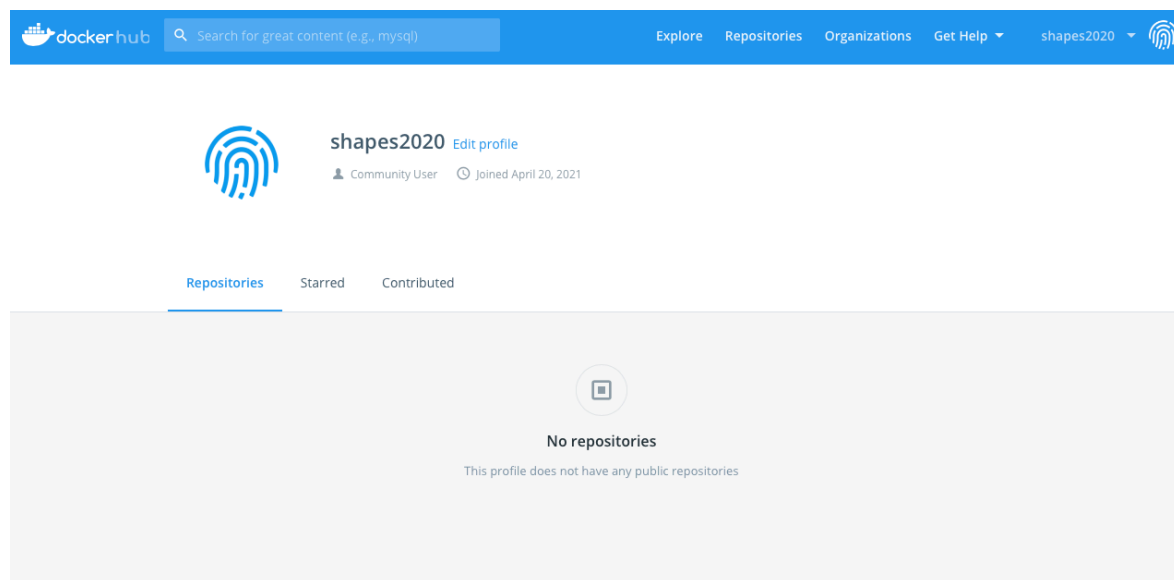


Figure 43 A snapshot of the SHAPES Docker Hub account.

7.3 Integration testing

Integration testing is the phase in software testing, where one or two individual software components are combined and tested as a group. Thus, it is used to test a service and its functionalities to ensure that subsystems work fine together, for example the database and the application, mocking only other services. All the software components of the system must pass the integration tests defined, in order to release a new version of the software. With the help of the owners of the SHAPES core components test cases for testing the components' interfaces and their

⁵⁹ Docker HUB: <https://hub.docker.com>

functionalities will be defined. The tests defined will be then executed by the integrator to ensure the correct functioning of the interfaces and smooth interaction between the integrated components. In case of test failures, respective developers will be notified to fix identified bugs, ensuring a correct operation of the system to be released.

8 Summary of Outcomes and Conclusions

This report aimed at summarising and detailing the whole of the work performed in WP4 by all partners until M18, conducted as part of the following ongoing tasks:

- Task 4.1 “SHAPES TP Requirements & Mapping a Reference Architecture”
- Task 4.2 “Data Models Analysis and Guidelines for Semantic Interoperability”
- Task 4.3 “Implementation of Mediation Framework & Interoperability Services”
- Task 4.4 “Implementation & Deployment of Secure Cloud & Big Data Platform”
- Task 4.5 “Human Interaction & Visual Mapping”
- Task 4.6 “SHAPES Authentication, Security & Privacy Assurance”
- Task 4.7 “SHAPES Gateway Reference Implementation”

Through collaboration with WP5, WP6 and WP8 technical work has been performed with due consideration of end user needs from all Pilot Themes of SHAPES in joint teleconferencing with WP2 and WP6 as well as on technical level with providers of Digital Solutions in joint telco meetings with WP5. As such this report contains some elements of technical considerations for integration of Digital Solutions, while refraining from including specific details related to design and interfacing of Digital Solutions, which are expected to be included in the upcoming D5.3 deliverable from WP5 on M24. As such only Annex 3 outlining types of Digital Solutions deployments, as most important for work in WP4, has been included in D4.1.

The work in WP4 has performed user needs translation into specifications, although they might still change as the development work in both WP4 and WP5 progresses. In doing the translation inputs from WP2, WP3, WP5, WP6 and WP8 have been incorporated as detailed in Section 1. Based on this and though numerous discussions with suppliers of all relevant technologies and Digital Solutions, the “core architecture” has been defined, i.e. part of the SHAPES system responsible for offering expected interoperability functionalities to Digital Solutions and functional operations to users.

It is important to mention here that the core SHAPES system has been designed such that to provide all required functionalities, while avoiding transmission of private identifiable data through the SHAPES core, putting strong effort on ensuring that only Digital Solutions that need to have access to such data, can have it and under strict authorisation from owners of such data.

Each of the core components has been described w.r.t. to its functionalities, operation, design and interfacing with other core components and Digital Solutions (where applicable). Note also that many of the core components have already progressed with their development and many of the interfacing are in place and being tested.

NOTE: since deliverable D4.1 is the main and only document produced in WP4 that describes technical SHAPES solution and no other such document is expected to be produced in WP4, the D4.1 will continue being updated throughout the project to include always up to date information regarding design, implementation and interfaces among core SHAPES architectural components. This will happen until M30 when WP4 will conduct formal technical tests of the integrated SHAPES Technological Platform.

9 Ethical Requirements Check

The focus of this compliance check is on the ethical requirements defined in D8.4 and having impact on the SHAPES solution (technology and related digital services, user processes and support, governance-, business- and ecosystem models).

Comments: The relevant ethical aspects, as defined by D8.4, have been taken into consideration both within the table of specifications (spinning out of user needs) as presented in section 2.2 as well as detailed in relevant sections of D4.1, as indicated in the ethics compliance table above.

Ethical issue (corresponding subsection of D8.4 in brackets)	How it has been taken into account in this deliverable (if relevant)
Fundamental Rights (3.1)	N/A
Biomedical Ethics and Ethics of Care (3.2)	Refer to section 3.2 for details
CRPD and supported decision-making (3.3)	Accounted for in specs (2.2) Anticipated in front-end app (6.2.8)
Capabilities approach (3.4)	N/A
Sustainable Development and CSR (4.1)	N/A
Customer logic approach (4.2)	N/A
Artificial intelligence (4.3)	Limited applicability to processing of data only (refer to section 6.2.5 for details)
Digital transformation (4.4)	N/A
Privacy and data protection (5)	Architecture will not store, process and transfer private data (refer to section 3)
Cyber security and resilience (6)	Cyber-security anticipated in specs w.r.t best practices (section 4.1.4)
Digital inclusion (7.1)	N/A
The moral division of labour (7.2)	N/A
Care givers and welfare technology (7.3)	N/A
Movement of caregivers across Europe (7.4)	N/A

References

- [1] Bolognesi, T., Brinksma, Ed., Introduction to the ISO Specification Language LOTOS, in the Formal Description Technique LOTOS, Elsevier Sci. Pub., pp. 23–73, 1989.
- [2] ISO., Information Processing Systems - Open System Interconnection - LOTOS - A Formal Description Technique based on the Temporal Ordering of Observational Behavior, IS 8807, 1989. Google Scholar
- [3] Emerson, E.A., Temporal and Modal Logic, Handbook of Theoretical Computer Science, Elsevier Science Publishers B.V., pp. 995–1072, 1990.
- [4] Milner R., Communication and Concurrency, Prentice-Hall, 1989.
- [5] Vishnu N. Boddeti. Secure face matching using fully homomorphic encryption. IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS), Redondo Beach, CA, USA, pp. 1-10, 2018.
- [6] Antitza Dantcheva, Petros Elia, Arun Ross. What else does your biometric data reveal? A survey on soft biometrics. IEEE Transactions on Information Forensics and Security, Institute of Electrical and Electronics Engineers, 11 (3), pp.441-467, 2015.
- [7] Asma E. K. Ghaleb, Najoua E. B. Amara. Soft and hard biometrics for the authentication of remote people in front and side views. Int J Appl Eng Res. 11(14), pp. 8120-7, 2016.
- [8] Anil K. Jain, Patrick Flynn, Arun A. Ross (Eds.). Handbook of Biometrics. Springer US, 2008.
- [9] Amine Nait-Ali (Ed.) Hidden Biometrics. Series in BioEngineering, Springer Singapore, 2020.
- [10] Yuanhan Zhang, Zhenfei Yin, Yidong Li, Guojun Yin, Junjie Yan, Jing Shao, Ziwei Liu CelebA-Spoof: Large-scale face anti-spoofing dataset with rich annotations. Computer Vision – ECCV. Lecture Notes in Computer Science, vol 12357, 2020.
- [11] Antoniou, Grigoris, and Frank van Harmelen. 2009. “Web Ontology Language: OWL.” In Handbook on Ontologies, edited by Steffen Staab and Rudi Studer, 91–110. Berlin, Heidelberg: Springer.
- [12] Soursos, S., Jarko, I.P., Zwickl, P., Gojmerac, I., Bianchi, G., Carrozzo, G.: Towards the cross-domain interoperability of iot platforms. In: 2016 European Conference on Networks and Communications (2016).
- [13] M. Jacoby, A. Antonic, K. Kreiner, R. Lapacz and J. Pielorz, “Semantic Interoperability as Key to IoT Platform Federation,” Interoperability and Open-Source Solutions for the Internet of Things, Forthcoming 2017.
- [14] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, CVPR 2020.
- [15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, SSD: Single Shot MultiBox Detector, ECCV 2016.
- [16] Guangcan Mai, Kai Cao, Pong C. Yuen, Anil K. Jain. Face image reconstruction from deep templates. CoRR abs/1703.00832, 2017.

Annex 1 FINoT Middleware API for SHAPES

Overview

The REST APIs provide programmatic access to FINoT middleware API.

Conventions

We use the following conventions in this document:

- Responses are listed under 'Responses' for each method.
- Responses are in JSON format.
- Request parameters are mandatory unless explicitly marked as Optional.
- The type of values accepted for a request parameter are shown the values column.

Schema

All API access is over HTTPS, and accessed from the <https://api.shapes.finet.cloud>

All data is sent and received as JSON.

Timestamps are in ISO 8601 format: YYYY-MM-DDTHH:MM:SSZ (UTC Zulu Time)

HTTP Requests

API requests must be written as HTTP requests, and include following components:

- **HTTP Method:** Describes the type of HTTP action (POST, GET, PUT or DELETE)
- **URL:** Describes a resource to be created or accessed, with any optional arguments
- **HTTP Headers:** Specifies request attributes, including authentication, encoding and request format
- **Request Body:** Describes resources or specifies a call-control script

URL Format

Describe the format of the URL.

`<protocol>://<host>:<port>/<module>/<version>/<ResourceName>`

Authentication

Username and password required.

Request

Method	URL
POST	https://api.shapes.fiot.cloud/auth/jwt/login

Parameters

Name	Type	Description	Required
loginId	String	User email address	Yes
password	String	User password	Yes

Response

Name	Type	Description
token	String	JWT token (Valid for 60 Minutes)
refreshToken	String	Refresh token (Valid for 30 Days)
user.email	String	User email
user.id	String	User Id
user.tenant	String	Tenant name
user.roles	List	List of user roles assigned

Examples

Request

Method	URL
POST	https://api.shapes.fiot.cloud/auth/jwt/login
REQUEST BODY	
<pre>{ "loginId": "demouser@shapes.com", "password": "demodemo" }</pre>	

Response

Status	Response
200	<pre>{ "status": 200, "result": { "token": "5cCl6lkpXVCIsCl6IjYzM...", "user": { "email": "demouser@shapes.com", "tenant": "shapes", "id": "ce363781-7ba0-4772-a0d8-04d231a2945f", "roles": ["role1", "role2"] }, "refreshToken": "IBnmvKadPxxkletcJzkpg1ctw..." } }</pre>

JWT Token Refresh

Request

Method	URL
POST	https://api.shapes.fiot.cloud/auth/jwt/refresh

Parameters

Name	Type	Description	Required
refreshToken	String	Refresh token	Yes

Response

Name	Type	Description
token	String	JWT token (Valid for 60 Minutes)

Example

Request

Method	URL
POST	https://api.shapes.fiot.cloud/auth/jwt/refresh
REQUEST BODY	
<pre>{</pre>	


```
{
  "refreshToken": "ZfN7kxT9HwdkV_BIJQyYOaA..."
}
```

Response

Status	Response
200	{ "status": 200, "result": { "token": "eyJhbGciOiJIUzI1NiUxQ..." } }

Object Management

List Objects

Retrieves a list of objects which match criteria defined by the following parameters: id, type, idPattern, q, geometry and coords attribute (see below for a detailed description of these parameters). A given entity have to match all the criteria to be retrieved (i.e. criteria are combined in a logical AND way).

Request

Method	URL
GET	https://api.shapes.finot.cloud/inventory/v1/objects

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name “ shapes ”	Yes
Authorization	String	JWT <Token>	Yes

URL Parameters

All the URL parameters are optional

Name	Type	Description
id	String	A comma separated list of elements. Retrieve entities which ID match one of the elements in the list.
type	String	Comma-separated list of elements. Retrieve entities which type match one of the elements in the list.

q	String	A query expression, composed of a list of statements separated by ;, i.e. q=statement;statements;statement
geometry	String	Defines a geographical area so only the entities located in that area matches the query. It is composed of a tokens list separated by;. The first token is the shape of the geometry, the rest of the tokens (if any) depends on the shape
coords	String	List of coordinates separated by; are interpreted depending on the geometry parameter depends on the shape.
attrs	String	Comma-separated list of attribute names which data will be included in the response. If this parameter is not included, all the attributes are retrieved.

Response

Name	Type	Description
objects[].id	String	Object id
objects[].type	String	Object type
objects[].location	geo:json	Object location
objects[].<xxxxxx>	Json	Object attribute

Example

Request

Method	URL
GET	https://api.shapes.fiot.cloud/inventory/v1/objects
REQUEST HEADERS	
X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzI...1NilsInR5cCI6IkpXVC"	

Response

Status	Response
200	<pre>{ "status": 200, "result": { "objects": [{ "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "Timestamp": { "type": "ISO8601", </pre>

	<pre> "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, "temperature": { "type": "Number", "value": "20", "metadata": { "TimeInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" } } }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] } }, "metadata": {} } } } </pre>
--	--

Create Object

The payload is an object representing the entity to be created. The object follows the JSON entity representation format (described in a section above).

Supported Data Types

Type Name	Description	Example Value
Boolean	Boolean	True
DateTime	ISO8601 timestamp Zulu time (UTC)	2018-09-12T20:17:46Z %Y-%m-%dT%H:%M:%SZ
Integer	Integer	10
Number	Float (any number value)	0.54
Text	String	"Hello world"
geo:json	GeoJSON encoded geographic data (Points, Polygons, etc...) RFC7946	{"type": "Point", "coordinates": [37.996, 23.815]}
File	File Reference URL	http://static.local/a7d55d0a

Request

Method	URL
POST	https://api.shapes.fiot.cloud/inventory/v1/objects

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name “ shapes ”	Yes
Authorization	String	JWT <Token>	Yes

Response

Name	Type	Description
object	json	Object

Example

Request

Method	URL
POST	https://api.shapes.fiot.cloud/inventory/v1/objects
REQUEST HEADERS	
X-Tenant: “shapes” Authorization: “JWT eyJhbGciOiJIUzIuLCI6IkpXVC”	
REQUEST BODY	
<pre>{ "type": "MyShapesObject", "attributes": [{ "name": "temperature", "type": "Number", "value": "20" }], "static_attributes": [{ "name": "location", "type": "geo:json", "value": { "type": "Point", "coordinates": [37.996523, 23.814874] } }] }</pre>	

```

    }
  }
}
}

```

Response

Status	Response
201	<pre> { "status": 201, "result": { "object": { "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "TimeInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, "temperature": { "type": "Number", "value": "20", "metadata": { "TimeInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" } } }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] } }, "metadata": {} } } } </pre>

Retrieve Object

The response is an object representing the entity identified by the ID. The object follows the JSON entity representation format.

This operation must return only one entity element, but it may happen that there are more than one entity with the same ID (e.g. entities with same ID but different type). In those cases an error message is returned, specifying in the description the URL that could be used to get the list of conflicting entities, i.e. all entities with such an ID.

Request

Method	URL
GET	<a href="https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>">https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name “ shapes ”	Yes
Authorization	String	JWT <Token>	Yes

Response

Name	Type	Description
object	json	Object

Example

Request

Method	URL
GET	https://api.shapes.fiot.cloud/inventory/v1/objects/ce331003-0069-48b9-a784-d09222a842ee
REQUEST HEADERS	
X-Tenant: “shapes” Authorization: “JWT eyJhbGciOiJIUzIu....1NilsInR5cCI6IkpXVC”	

Response

Status	Response
201	<pre>{ "status": 201, "result": { "object": { "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "TimeInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, }, "temperature": { "type": "Number", "value": "20", "metadata": { "TimeInstant": {</pre>

	<pre> "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" }, }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] } }, "metadata": {} } } } </pre>
--	--

Update Object Attributes

The request payload is an object representing the attributes to update. The object follows the JSON entity representation format except that id and type are not allowed.

Request

Method	URL
PATCH	<a href="https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>">https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name "shapes"	Yes
Authorization	String	JWT <Token>	Yes

Response

Name	Type	Description
object	json	Object

Example

Request

Method	URL
--------	-----

PATCH	https://api.shapes.fiot.cloud/inventory/v1/objects/ce331003-0069-48b9-a784-d09222a842ee
REQUEST HEADERS	
X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzI...1NilsInR5cCI6IkpXVC"	
REQUEST BODY	
<pre>{ "temperature": { "type": "Number", "value": 20.5 } }</pre>	

Response

Status	Response
200	<pre>{ "status": 200, "result": { "object": { "id": "ce331003-0069-48b9-a784-d09222a842ee", "type": "MyShapesObject", "TimeInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z", "metadata": {} }, "temperature": { "type": "Number", "value": "30", "metadata": { "TimeInstant": { "type": "ISO8601", "value": "2019-12-06T13:37:36.00Z" } } }, "location": { "type": "geo:json", "value": { "type": "Point", "coordinates": [45.466377, 13.654166] } }, "metadata": {} } } }</pre>

Remove object

Remove the object and the historical data.

Request

Method	URL
PATCH	<a href="https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>">https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name “ shapes ”	Yes
Authorization	String	JWT <Token>	Yes

Example

Request

Method	URL
DELETE	https://api.shapes.fiot.cloud/inventory/v1/objects/ce331003-0069-48b9-a784-d09222a842ee
REQUEST HEADERS	
X-Tenant: “shapes” Authorization: “JWT eyJhbGciOiJIUzIu....1NilsInR5cCI6IkpXVC”	

Response

Status	Response
204	

Retrieve Historical Data

Request

Method	URL
GET	<a href="https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>/data">https://api.shapes.fiot.cloud/inventory/v1/objects/<objectId>/data

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name “ shapes ”	Yes

Authorization	String	JWT <Token>	Yes
---------------	--------	-------------	-----

URL Parameters: all URL parameters are optional

Name	Type	Description
attrs	String	Comma-separated list of attribute names whose data are to be included in the response. The attributes are retrieved in the order specified by this parameter. If not specified, all attributes are included in the response in arbitrary order.
aggrMethod	String	The function to apply to raw data filtered by the query parameters. If not given, the returned data are the same raw inserted data. <i>Available values : count, sum, avg, min, max</i>
aggrPeriod	String	If not defined, the aggregation will apply to all the values contained in the search result. If defined, the aggregation function will instead be applied N times, once for each period, and all those results will be considered for the response. For example, a query asking for the average temperature of an attribute will typically return 1 value. However, with an aggregationPeriod of day, you get the daily average of the temperature instead (more than one value assuming you had measurements across many days within the scope of your search result). aggrPeriod must be accompanied by an aggrMethod, and the aggrMethod will be applied to all the numeric attributes specified in attrs; the rest of the non-numerical attrs will be ignored. By default, the response is grouped by entity_id. See aggrScope to create aggregation across entities. <i>Available values : year, month, day, hour, minute, second</i>
fromDate	String	The starting date and time (inclusive) from which the context information is queried. Must be in ISO8601 format (e.g., 2018-01-05T15:44:34)
toDate	String	The final date and time (inclusive) from which the context information is queried. Must be in ISO8601 format (e.g., 2018-01-05T15:44:34)
lastN	Integer	Used to request only the last N values that satisfy the request conditions.
limit	Integer	Maximum number of results to retrieve in a single response.
offset	Integer	Offset to apply to the response results. For example, if the query was to return 10 results and you use an offset of 1, the response will return the last 9 values. Make sure you don't give more offset than the number of results.
georel	String	It specifies a spatial relationship between matching entities and a reference shape (geometry). This parameter is used to perform geographical queries.
geometry	String	This parameter defines reference shape to be used for geographical queries and has the same semantics. <i>Available values : point, line, polygon, box</i>
coords	String	Optional but required if georel is specified. This parameter defines the reference shape (geometry). Specification, except we only accept coordinates in decimal degrees e.g. 40.714,-74.006

Response

Name	Type	Description
data	json	Historical data response

Example

Request

Method	URL
GET	https://api.shapes.fiot.cloud/inventory/v1/objects/ce331003-0069-48b9-a784-d09222a842ee/data
REQUEST HEADERS	
X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzI...1NilsInR5cCI6IkpXVC"	

Response

Status	Response
200	<pre>{ "status": 200, "result": { "data": { "attributes": [{ "attrName": "temperature", "values": [12.612307695242075, 11.03249986966451, 11.437272678722035, 8.432499885559082] }] }, "entityId": "ce331003-0069-48b9-a784-d09222a842ee", "index": ["2019-12-03T00:00:00.000", "2019-12-04T00:00:00.000", "2019-12-05T00:00:00.000", "2019-12-08T00:00:00.000"] } }</pre>

Subscriptions Management

List Subscriptions

Request

Method	URL
GET	https://api.shapes.fiot.cloud/inventory/v1/subscriptions

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name " shapes "	Yes
Authorization	String	JWT <Token>	Yes

Response

Name	Type	Description
subscriptions	json	Subscription List

Example

Request

Method	URL
GET	https://api.shapes.fiot.cloud/inventory/v1/subscriptions
REQUEST HEADERS	
X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzIuLCI6IkpXVC"	

Response

Status	Response
200	<pre>{ "status": 200, "result": { "subscriptions": [{ "id": "5de7c0f787a29205d80e220c", "description": "my-subscription", "status": "active", "subject": { "entities": [{ "idPattern": ".*", "type": "MyShapesObject" }] } }] } }</pre>

	<pre> "condition": { "attrs": ["temperature"] }, }, "notification": { "timesSent": 54, "lastNotification": "2019-12-05T15:18:24.00Z", "attrs": ["temperature"], "onlyChangedAttrs": false, "attrsFormat": "normalized", "http": { "url": "http://myservice.com/notify" }, "metadata": ["TimeInstant"], "lastSuccess": "2019-12-05T15:18:24.00Z", "lastSuccessCode": 200 } }] } } } </pre>
--	--

Create Subscription

Request

Method	URL
POST	https://api.shapes.fiot.cloud/inventory/v1/subscriptions

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name “ shapes ”	Yes
Authorization	String	JWT <Token>	Yes

Response

Name	Type	Description
-	-	-

Example

Request

Method	URL
POST	https://api.shapes.fiot.cloud/inventory/v1/subscriptions
REQUEST HEADERS	
X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzIuLC1NiIsInR5cCI6IkpXVC"	
REQUEST BODY	
<pre>{ "description": "my-subscription", "subject": { "entities": [{ "idPattern": ".*", "type": "MyShapesObject" }], "condition": { "attrs": ["temperature>30"] } }, "notification": { "http": { "url": "http://myservice.com/notify" }, "attrs": ["temperature"], "metadata": ["TimeInstant"] } }</pre>	

Response

Status	Response
201	<pre>{ "status": 201, "result": {} }</pre>

Remove Subscription

Request

Method	URL
--------	-----

DELETE	<a href="https://api.shapes.fiot.cloud/inventory/v1/subscriptions/<subld>">https://api.shapes.fiot.cloud/inventory/v1/subscriptions/<subld>
--------	---

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name " shapes "	Yes
Authorization	String	JWT <Token>	Yes

Response

Name	Type	Description
-	-	-

Example

Request

Method	URL
DELETE	https://api.shapes.fiot.cloud/inventory/v1/subscriptions/5de7c0f787a29205d80e220c
REQUEST HEADERS	
X-Tenant: "shapes" Authorization: "JWT eyJhbGciOiJIUzI...1NilsInR5cCI6IkpXVC"	

Response

Status	Response
204	{ "status": 204, "result": {} }

File Download

Retrieve file from Middleware

Request

Method	URL
GET	<a href="https://api.shapes.fiot.cloud/inventory/v1/files/<objectId>/<filename>">https://api.shapes.fiot.cloud/inventory/v1/files/<objectId>/<filename>

Parameters

Name	Type	Description	Required
X-Tenant	String	Tenant name “ shapes ”	Yes
Authorization	String	JWT <Token>	Yes

Example

Request

Method	URL
DELETE	https://api.shapes.fiot.cloud/inventory/v1/files/6d8078ac-23e7-4001-a9f2-4467d9878140/9a4c8808-4b2c-44e4-b8de-08453926ec1f.png
REQUEST HEADERS	
X-Tenant: “shapes” Authorization: “JWT eyJhbGciOiJIUzIuLCI6IkpXVC”	

Response

Status	Response
200	...BINARY DATA... (FILE)

Annex 2 Instructions for symbloTe compliance

This section provides instructions about how to make an IoT platform L1- and L2-compliant for symbloTe.

Cloud deployment

The process of making an IoT platform symbloTe-enabled consists of downloading and setting up symbloTe Cloud components (shown in red in the figure below), integrating these components with your IoT platform, and registering your platform and resources to symbloTe Core Services (shown in blue in the figure below). We will refer to the symbloTe Core Services platform that includes centralized components implementing the symbloTe semantic IoT search engine (deployed by ICOM) as *symbloTe Core* and the symbloTe Cloud Services platform which include all the needed components that extend the existing IoT platform with interoperability-related features to become symbloTe-enabled as *symbloTe Cloud*.

For the needs of the SHAPES project, L2-compliance level supported by symbloTe is the desired compliance level. symbloTe Core is deployed for testing purposes at: <https://intracom-core.symbloTe-h2020.eu/>.

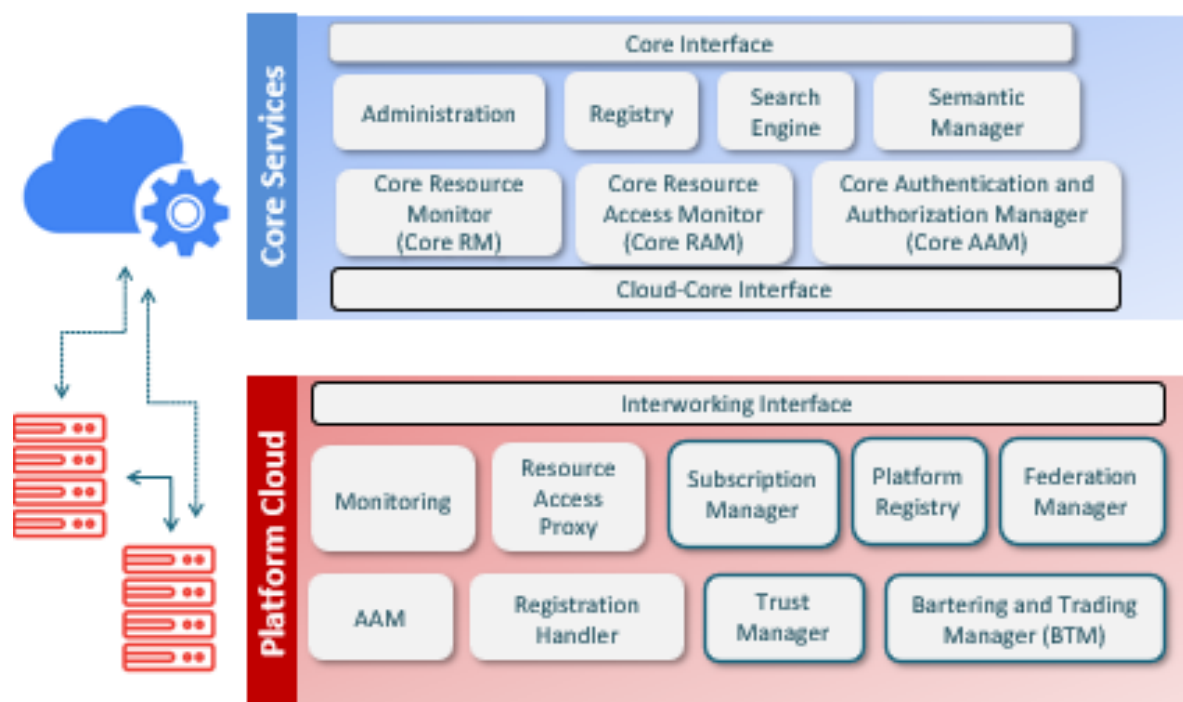


Figure 44: *symbloTe components for I2-compliance*

This document explains the steps to deploy, configure and run symbloTe Cloud (L2-compliance) with Docker directly on Linux. It provides up-to-date instructions based on the online documentation of symbloTe available at: <https://github.com/symbloTe-h2020/symbloTeCloud/wiki/symbloTeCloud-from-docker>.

Preparation steps

Prerequisites

Install the following on Linux:

- Docker (18.03.x)
- Docker-compose (1.21.x)
- Docker-machine (0.14.x)
- bash
- curl
- wget

Registration of platform owner

In this step it is necessary to register the platform owner (user) to symbloTe Core through the symbloTe Core Administration webpage. Visit the symbloTe Core Administration web page at the following URL: <https://intracom-core.symbloTe-h2020.eu/administration/>, as shown in the figure below. Press the **Register** button.

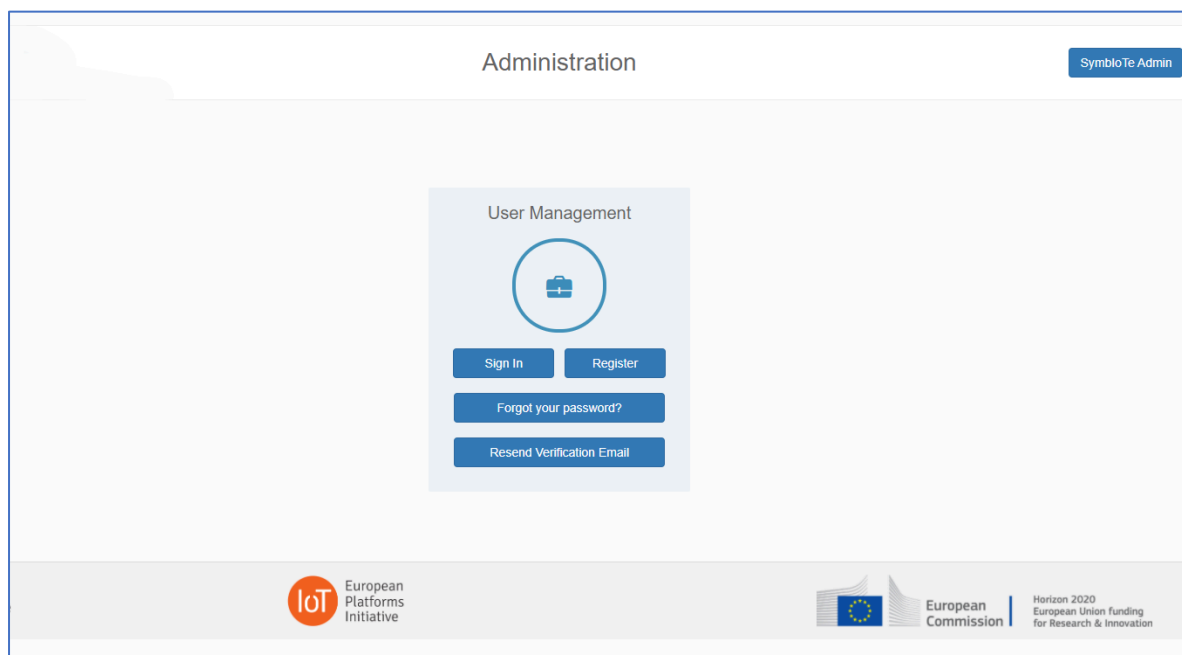
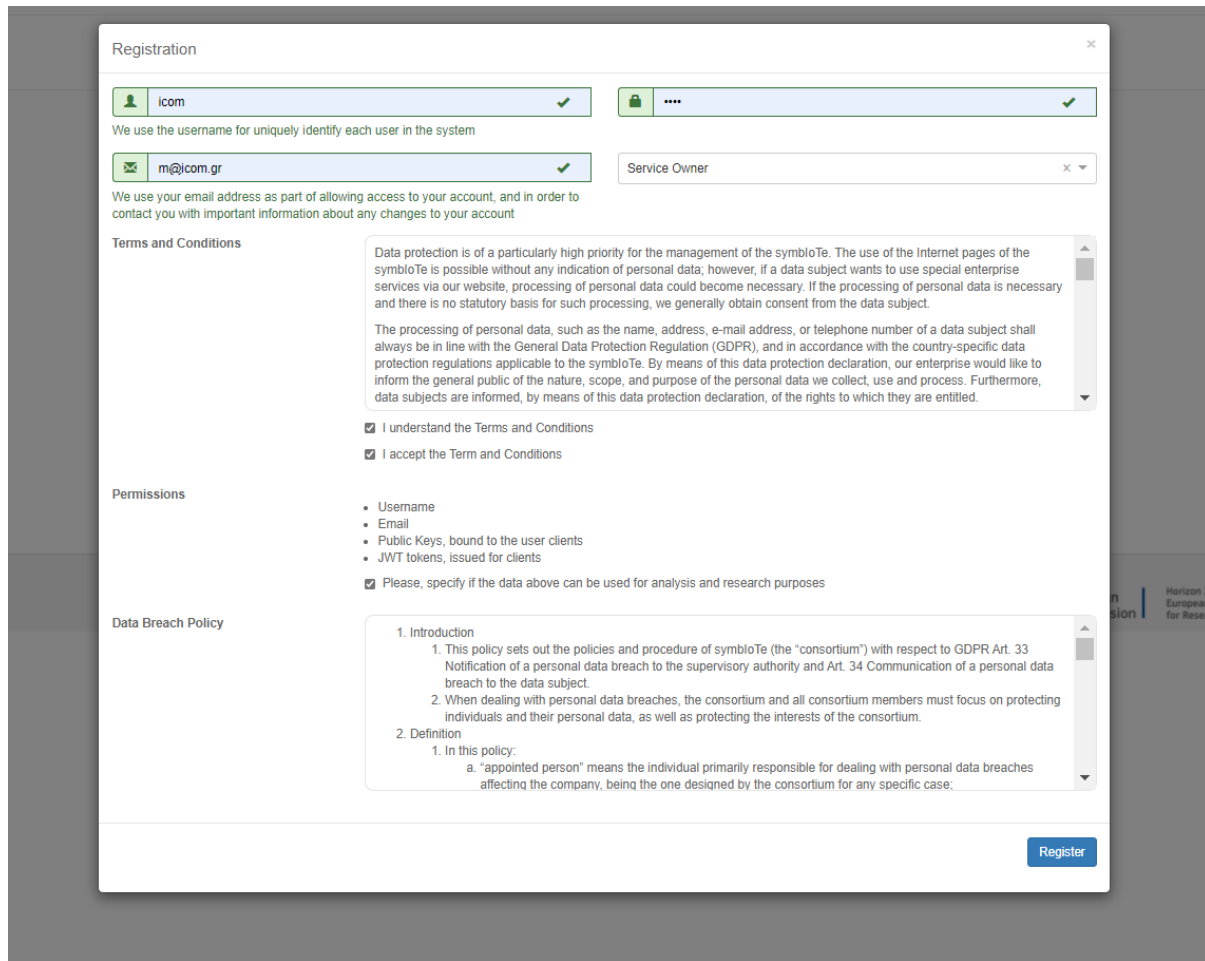


Figure 45: Access to symbloTe core administration GUI

As shown in the following figure, during registration you must provide:

- username
- password
- email
- user role (Service Owner in this case)



Registration

Username: ✓

We use the username for uniquely identify each user in the system

Password: ✓

Email: ✓

We use your email address as part of allowing access to your account, and in order to contact you with important information about any changes to your account

Service Owner:

Terms and Conditions

Data protection is of a particularly high priority for the management of the symbloTe. The use of the Internet pages of the symbloTe is possible without any indication of personal data; however, if a data subject wants to use special enterprise services via our website, processing of personal data could become necessary. If the processing of personal data is necessary and there is no statutory basis for such processing, we generally obtain consent from the data subject.

The processing of personal data, such as the name, address, e-mail address, or telephone number of a data subject shall always be in line with the General Data Protection Regulation (GDPR), and in accordance with the country-specific data protection regulations applicable to the symbloTe. By means of this data protection declaration, our enterprise would like to inform the general public of the nature, scope, and purpose of the personal data we collect, use and process. Furthermore, data subjects are informed, by means of this data protection declaration, of the rights to which they are entitled.

☒ I understand the Terms and Conditions

☒ I accept the Term and Conditions

Permissions

- Username
- Email
- Public Keys, bound to the user clients
- JWT tokens, issued for clients

☒ Please, specify if the data above can be used for analysis and research purposes

Data Breach Policy

1. Introduction

1. This policy sets out the policies and procedure of symbloTe (the "consortium") with respect to GDPR Art. 33 Notification of a personal data breach to the supervisory authority and Art. 34 Communication of a personal data breach to the data subject.

2. When dealing with personal data breaches, the consortium and all consortium members must focus on protecting individuals and their personal data, as well as protecting the interests of the consortium.

2. Definition

1. In this policy:

a. "appointed person" means the individual primarily responsible for dealing with personal data breaches affecting the company, being the one designed by the consortium for any specific case;

[Register](#)

Figure 46: Registration of platform owner to symbloTe core

Registration of platform's L2 Cloud services

Next, after successful registration you can log in (with the account details used for the registration of the platform owner) and register your platform (i.e., the symbloTe Cloud Services platform). Click on the **Platform Details** panel and then click on **Register New Platform** button on the upper right corner.

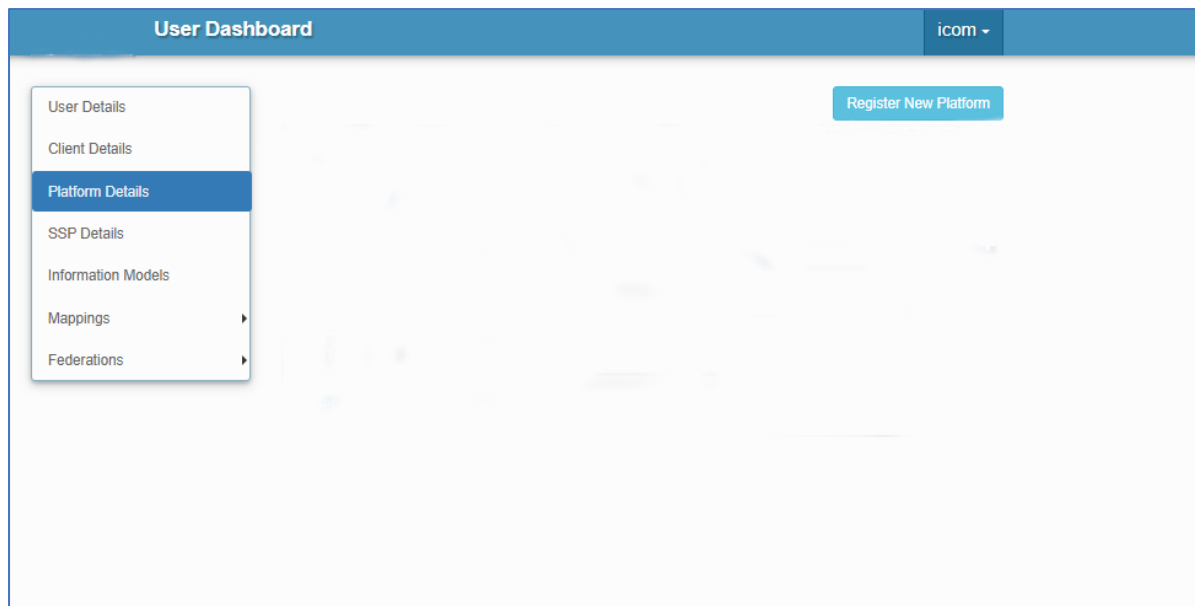
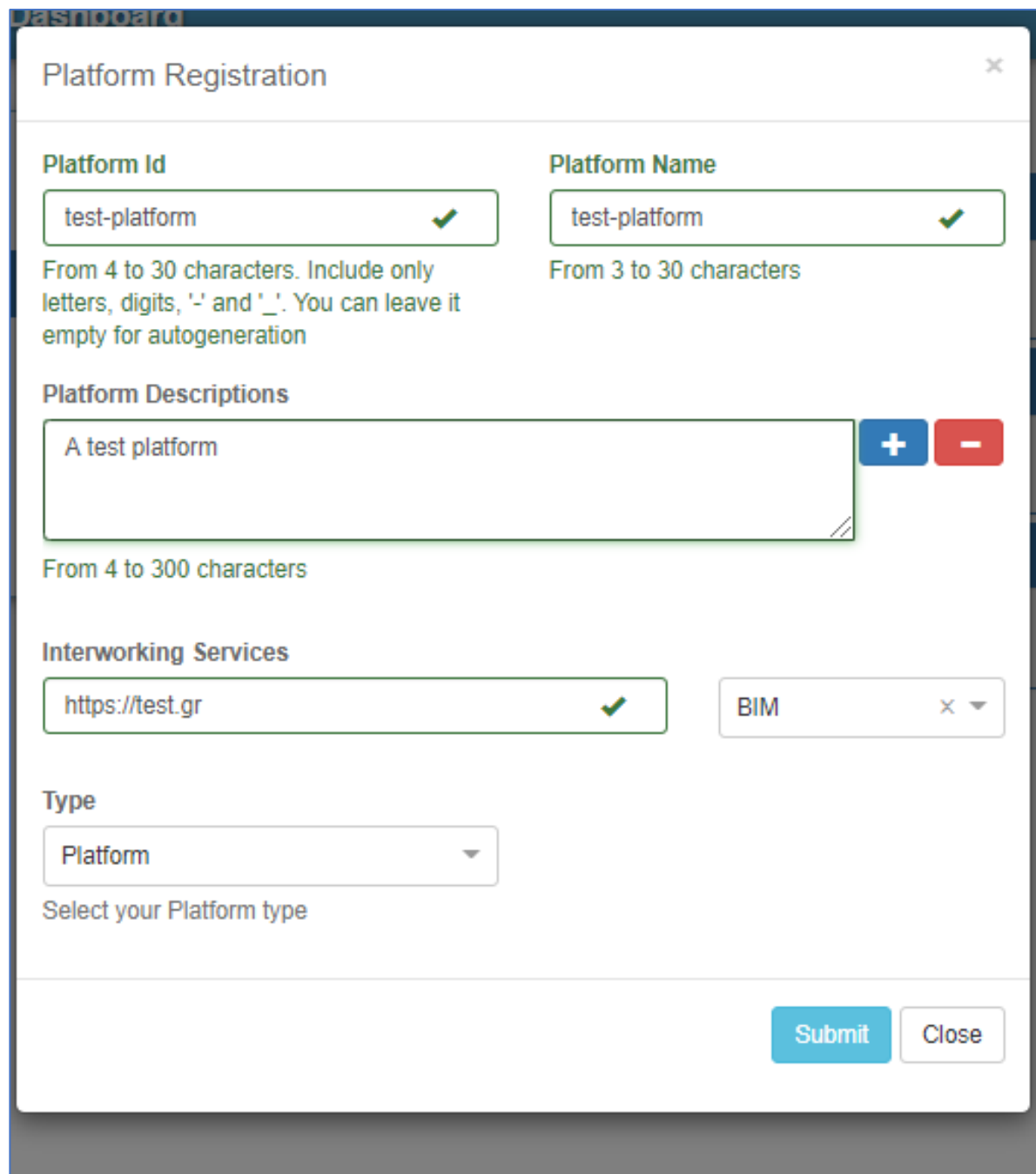


Figure 47: Registration of the platform

Then, the following details must be provided as shown in the figure below:

- **Preferable platform ID**
- **Platform Name**
- **Platform Description**
- **Interworking Services:** This is the valid URL to your Linux host where you will install L2 Cloud SHAPES services. It needs public IP address and DNS entry for that IP address. The alternative is to use ngrok tool which is good for experimentation but not for production.
- **Interworking Interface Information Model:** you can use BIM (Best practice Information Model) for out-of-the-box interoperability using the supported information model used in symbloTe or PIM (Platform Information Model) to register a platform-specific information model that extends the supported information model.
- **Type** (i.e. Platform or Enabler): in this case use Platform.



Platform Registration

Platform Id
test-platform ✓
From 4 to 30 characters. Include only letters, digits, '-' and '_'. You can leave it empty for autogeneration

Platform Name
test-platform ✓
From 3 to 30 characters

Platform Descriptions
A test platform + -
From 4 to 300 characters

Interworking Services
https://test.gr ✓ BIM x ▾

Type
Platform ▾
Select your Platform type

Submit **Close**

Figure 48: Example of platform details in registration

After completing this procedure, the IoT platform is now registered to the symbloTe Core. It can be seen by selecting the **Platform Details** menu item.

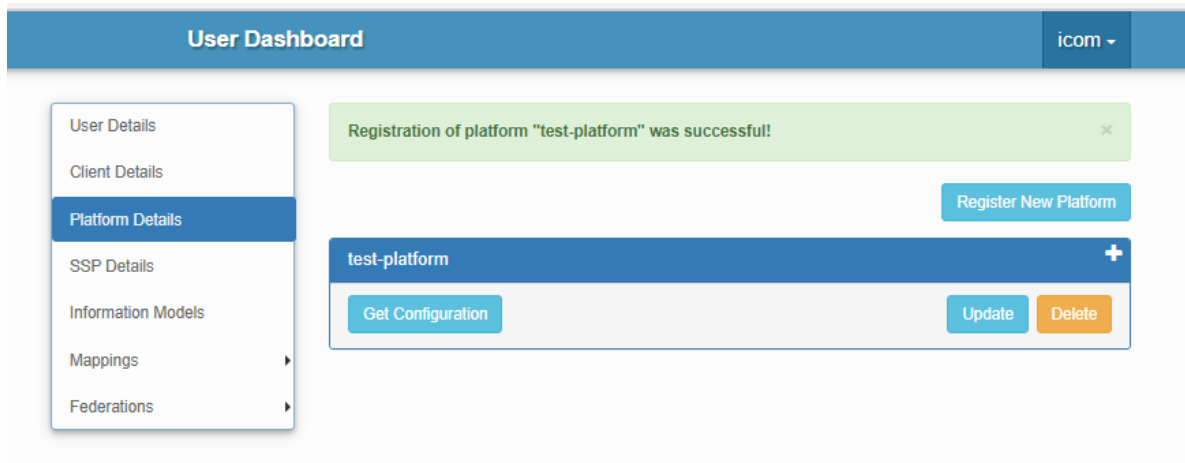


Figure 49: Confirmation of Successful platform registration

The panel of the newly registered platform and its details are available by clicking on its header or + sign.

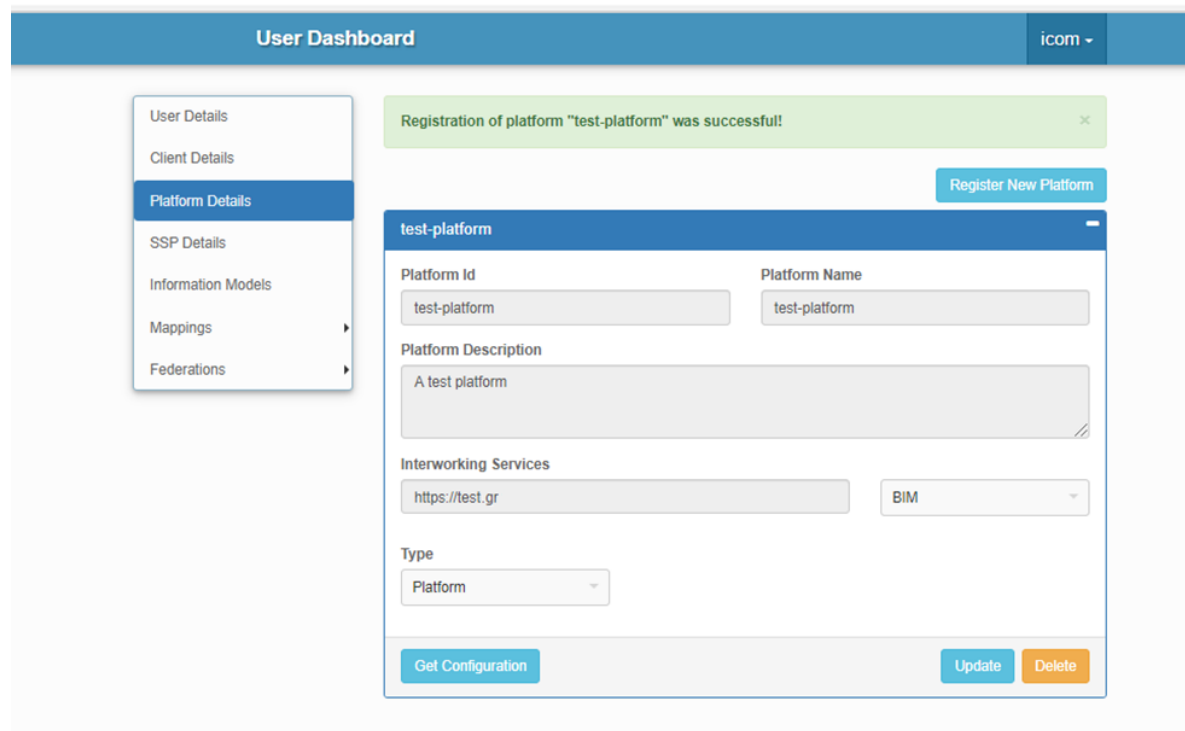
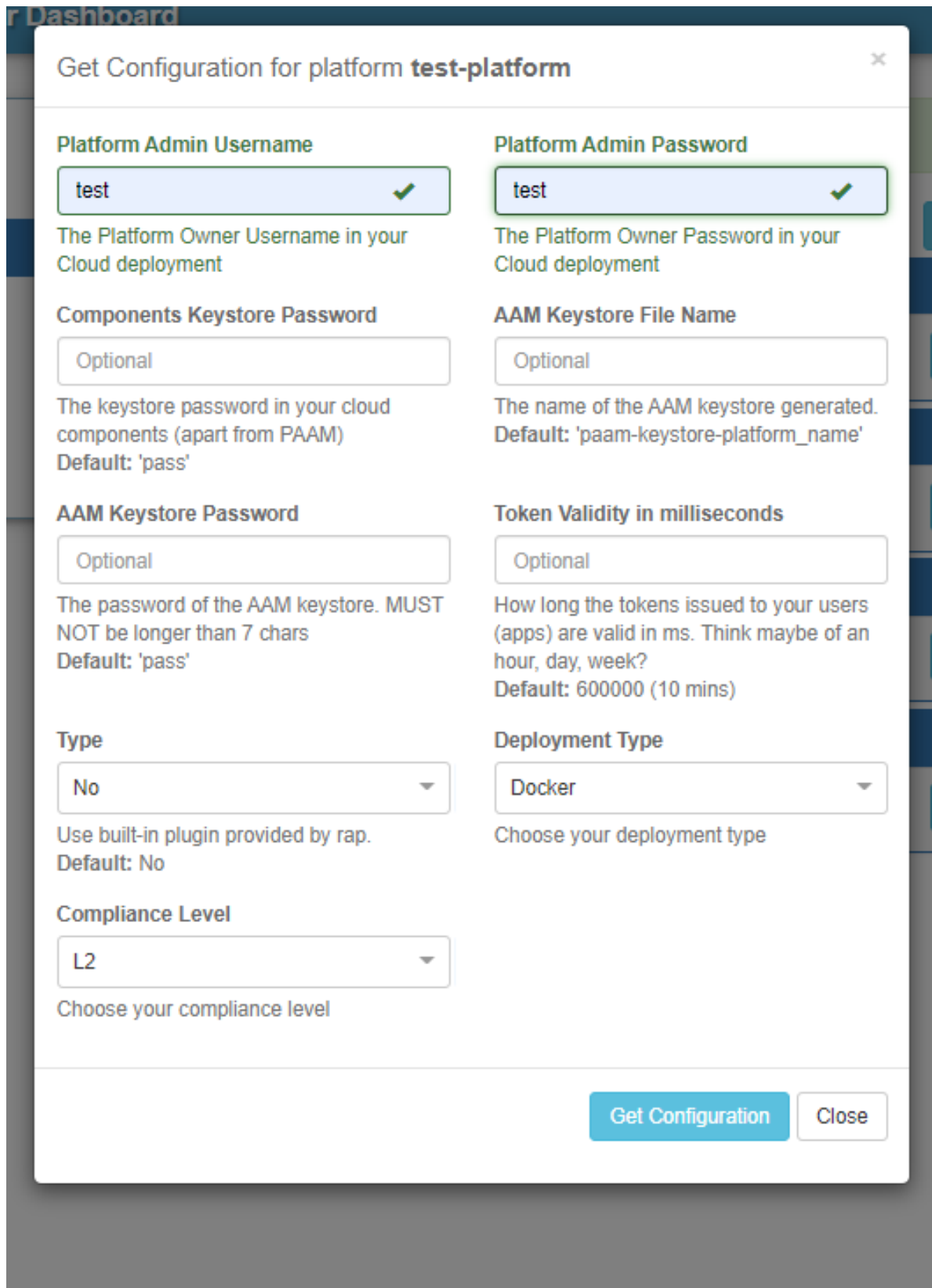


Figure 50: Panel of the platform registered

Download configuration files of registered platform's L2 Cloud services.

The next step is to download the necessary configuration for your deployment of the symbloTe L2-compliance Cloud services. When opening the panel of the platform registered, as shown in the previous section, you can download the platform's configuration files by clicking on the **Get Configuration** button and entering some details as will be explained below. Hence, a .zip file will be downloaded that contains platform configuration properties which simplify the services configuration process.



Get Configuration for platform test-platform

Platform Admin Username
test ✓
The Platform Owner Username in your Cloud deployment

Platform Admin Password
test ✓
The Platform Owner Password in your Cloud deployment

Components Keystore Password
Optional
The keystore password in your cloud components (apart from PAAM)
Default: 'pass'

AAM Keystore File Name
Optional
The name of the AAM keystore generated.
Default: 'paam-keystore-platform_name'

AAM Keystore Password
Optional
The password of the AAM keystore. MUST NOT be longer than 7 chars
Default: 'pass'

Token Validity in milliseconds
Optional
How long the tokens issued to your users (apps) are valid in ms. Think maybe of an hour, day, week?
Default: 600000 (10 mins)

Type
No
Use built-in plugin provided by rap.
Default: No

Deployment Type
Docker
Choose your deployment type

Compliance Level
L2
Choose your compliance level

Get Configuration **Close**

Figure 51: Platform configuration details

When pressing the **Get Configuration** button, a configuration form is displayed (see the figure below).

Fill the mandatory fields:

- Platform Admin Username.
- Platform Admin Password.

And set

- Type: No.
- Deployment Type: Docker.
- Compliance Level: L2

The optional fields can be left with their default values or you can set any other value that is necessary for the registered L2 cloud platform.

Press the **Get Configuration** button to download the **configuration.zip** file.

Create folder for your deployment and configure necessary property files

- Create a folder for your deployment, e.g. shapesL2-cloud:

```
mkdir shapesL2-cloud
```

- Change to that directory: e.g.

```
cd shapesL2-cloud
```

- Unzip in shapesL2-cloud folder the downloaded configuration.zip file.
- Enter the *CloudConfigProperties* directory.

In *CloudConfigProperties/application.properties* file set URLs of the interworking interfaces. There are two kinds of interworking interfaces at symbloTe Core:

1. *coreInterface* serves northbound traffic coming from 3rd parties (e.g. applications searching for resources):

```
symbIoTe.core.interface.url = https://intracom-core.symbIoTe-h2020.eu/coreInterface
```

2. *cloudCoreInterface* serves southbound traffic coming from IoT platforms (e.g. applications) running at the symbloTe cloud:

```
symbIoTe.core.cloud.interface.url = https://intracom-core.symbIoTe-h2020.eu/cloudCoreInterface
```

- Enter the *shapesL2-cloud* directory.
- In *AuthenticationAuthorizationManager/cert.properties* set the address of the symbloTe core AAM (available through the *coreInterface*):

```
coreAAMAddress = https://intracom-core.symbIoTe-h2020.eu/coreInterface
```

Commit the changes

Inside the *CloudConfigProperties* directory execute the following commands:

```
rm -r .git
git init
git config user.email you@example.com
```

```
git config user.name "Your User Name"
git add .
git commit -m "Your platform's name"
```

Create a Docker volume to hold settings stored in CloudConfigProperties folder

- Enter to the shapesL2-cloud directory.
- Replace the **{docker stack name}** in the next Docker command with your selected Docker stack name (e.g., shapesL2_cloud).
- Inside the shapesL2-cloud directory execute the Docker volume command:

```
docker container run --rm -v $PWD/CloudConfigProperties:/source -v {docker stack name} _symbIoTe-vol-config:/home/CloudConfigProperties -w /source alpine cp -r . /home/CloudConfigProperties/
```

Configure the Docker compose files

Copy the following docker compose files into the shapesL2-cloud directory (the docker compose files to be used can be found in the provided documentation folder):

```
docker-compose-swarm-L2.yml
docker-compose-prod-swarm-L2.yml
```

In case your deployment (symbIoTe Cloud L2 services) runs behind a proxy then the **docker-compose-swarm-L2.yml** file must be configured to set the proxy settings. In case of proxy existence uncomment the lines beginning with:

- JAVA_HTTP_PROXY
- JAVA_HTTPS_PROXY
- JAVA_NON_PROXY_HOSTS

Set the content of lines (the respective environmental variables for the proxy settings) according to your proxy server setup.

Configuration of NGINX micro-service

At the deployed symbIoTe Cloud L2 services docker stack the NGINX microservice will also run. The NGINX microservice is configured using the **nginx.conf** and **nginx-prod.conf** files that are inside the shapesL2-cloud folder. The steps below need to be followed:

- Remove **nginx.conf**, **nginx-prod.conf** and **nginx-ngrok** stored in the **configuration.zip** file.
- Use **nginx.conf** and **nginx-prod.conf** files from provided documentation folder.
- Change the **server_name** setting in **nginx.conf** and **nginx-prod.conf** files: This is the platform name provided during registration such as test-platform.
- Create **fullchain.pem** and **privkey.pem** certificate files as described in <https://github.com/symbIoTe-h2020/symbIoTeCloud/wiki/2.1-Configuration-of-NGINX> or by some other provider.

You can also use the certbot command to create a certificate manually with dns validation:

```
certbot --manual certonly --preferred-challenges dns -d <domain name> --
email <admin email>
```

In that case you will be asked to deploy a DNS TXT record under a specific name. After the creation of the certificates, letsencrypt informs the user about the location of the new files.

This is the location where the files have to be copied from. The target location has to be synchronized with the nginx configuration. Normally, it is the nginx-certificates folder, which needs to be created as described next.

- In the **shapesL2-cloud** directory create the folder **nginx-certificates**, where nginx is configured to find **fullchain.pem** and **privkey.pem** files, executing the command:

```
mkdir nginx-certificates
```

Copy the created **fullchain.pem** and **privkey.pem** certificate files to **nginx-certificates** folder (alternatively symbolic links can be used):

```
sudo cp /etc/letsencrypt/live/{your domain}/fullchain.pem nginx-
certificates/
sudo cp /etc/letsencrypt/live/{your domain}/privkey.pem nginx-certificates/
sudo chown -R {user}:{group} nginx-certificates
```

Run the Docker stack of the symbloTe L2 Cloud services

Before deploying the L2 Cloud docker stack (shapesL2-cloud) verify that the file structure under **shapesL2-cloud** folder is the same as shown in the next figure.

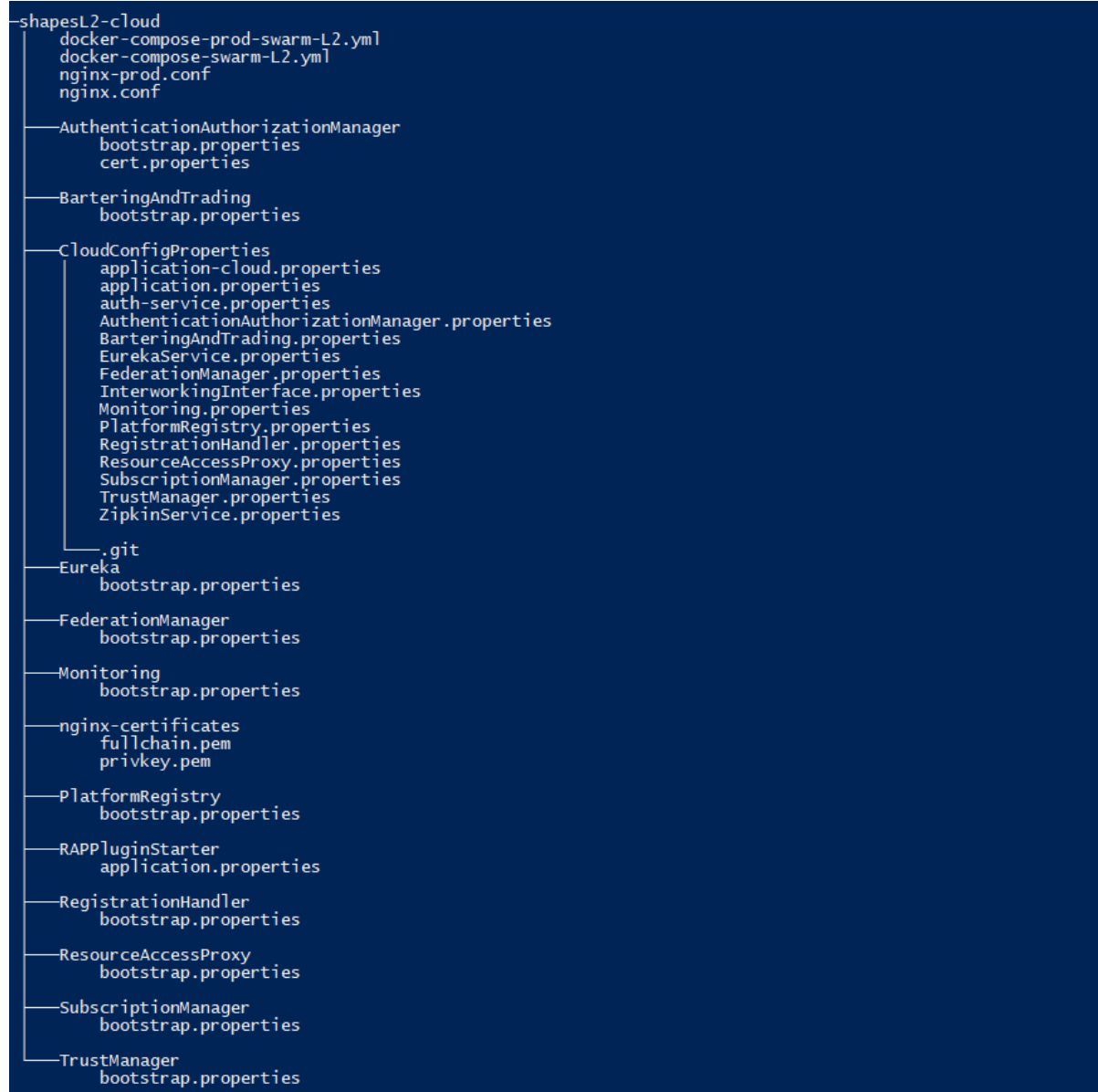


Figure 52: Structure of the deployment folder

Run symbloTe L2 Cloud services as a Docker stack

Follow the next steps to deploy symbloTe Cloud L2 as a Docker stack.

- Enter to the **shapesL2-cloud** directory.
- Run **docker swarm init** if the node is not a swarm manager. We use the swarm mode so that secrets are encrypted during transit and at rest. Docker secrets are only available to swarm services and not to standalone containers.
- Run **docker stack deploy -c docker-compose-swarm-L2.yml -c docker-compose-prod-swarm-L2.yml {docker stack name}**
{docker stack name} is the name of the service stack to be used and is assigned in the step described in 1.2.5 paragraph e.g. shapesL2-cloud.
- Now you can run **docker stack ls** to list the stack and check the number of services used.
- After a few seconds run **docker ps** to check the list of micro services running in the *{docker stack name}* stack.
- Run **docker image ls** to check that all images have been created. It may take a while to pull all the images from Docker Hub for the first time. Docker service ls to list the services and check their status. Wait until the actual number of tasks (replicas) for each service is not 0.
- Run **docker logs <container id> -f** to get access to and follow the logs of a service. A component is ready when a message similar to 'Started in 105.045 seconds (JVM running for 112.933)' appears in the logs of the container.
- Run **docker stack rm {docker stack name}** to stop the application and remove the service stack. Services, networks, and secrets associated with the stack will be removed.
- You can run **docker swarm leave --force** to leave the swarm.

Annex 3 Digital Solutions- Technical Supplement

This **Appendix** provides supplementary info regarding technical aspects of Digital Solutions from SHAPES that have not been included in deliverable D5.2, such that they can be made available to developers before the extended version of D5.3 will be submitted by WP5 by month M24.

Digital Solution	API/SDK (Link to relevant info)	Deployment Options (e.g. cloud service, on-premise server deployment etc)	HW/SW pre- requisites for deploying and using a given DS
HFPRED Heart failure decompensation predictive module (VICOM)	Proprietary API (to be included in D5.3) The API describes two endpoints that will be called from the SHAPES platform and will return both the risk (probability) of decompensation and the parameters that are critical for decompensation in the patient	HFPRED Docker Container to be added to the "VICOM's Data Analytics Engines" section in the Big Data Platform.	Software requirements: Docker. HFPRED must be run every time new data is obtained from each user and at least once a day (at 12:00 PM). Hardware requirements: does not require any special hardware.
CWDSS Clinical and Wellbeing Decision Support system (VICOM)	Proprietary API (Not available at the moment, to be defined in a later stage). The API describes an endpoint (HTTP POST request) which receives the input data (in JSON format) and then returns obtained recommendations (in JSON format). Both JSON objects can follow the FHIR standard.	CWDSS Docker Container to be added to the "VICOM's Data Analytics Engines" section in the Big Data Platform.	Software requirements: Docker. CWDSS has to be triggered, and its endpoint has to be used to provide the data and receive the corresponding recommendations. Hardware requirements: does not require any special hardware.
DAML Anomaly detection system (VICOM)	Proprietary API (Not available at the moment, to be defined in a later stage). The API describes an endpoint (HTTP POST request) which receives the input data (in JSON format) and then returns	DAML Docker Container to be added to the "VICOM's Data Analytics Engines" section in the Big Data Platform.	Software requirements: Docker. DAML has to be run every time new data is obtained from each user.

	the obtained anomaly level (in JSON format).		Hardware requirements: does not require any special hardware.
ARI Robot (PAL)	<p>Rest API, included as part of D4.1 (to be included in D5.3), and will be extended in D5.3 due to M24. The API is a wrapper on the Robotics Operating System (ROS) for integrators to use. It provides a set of endpoints which receive input data (in JSON format) and then return a response. For example, to say a text, go to dock station, execute motions, or send a navigation goal. Also available custom libraries to integrate JavaScript applications with ARI commands to produce touch-screen apps.</p> <p>On lower-level, partners can use ROS nodes to integrate their software.</p>	<p>On-premise local deployment. The robot is physically taken to each pilot site, but will already have digital solutions integrated, most of them running locally, except for those running on SHAPES Cloud, accessed through their respective APIs.</p>	<p>Hardware considerations:</p> <ul style="list-style-type: none"> -space for dock station 3x3 + wall to support base, at a safe area when it is not being used -power outlet to charge the robot (100-240 V 50/60 Hz) -restrictions for navigation: no stairs, ledges, slopes, away from wires, thresholds, carpets or dark settings. Ideally flat floor, 100 cm max wide, 60 cm radius area - laptop / phone for robot control -equipment to carry robot case from apartment to apartment depending on the user-case - 4G/5G connectivity for remote connection and support to the robots instead of the Wi-Fi. The robot functions in access mode so it is a produced Wi-Fi network -> module to switch from Wi-Fi to 4G inside the hospital depending on the requirements.

			Software considerations: DS may be integrated using API or as ROS Nodes
FACECOG Face Recognition Solution for Heterogeneous IoT Platforms - VICOM	Proprietary API (to be included in D5.3) The API describes a set of endpoints which receive the input data (in JSON format) and then return the requested response (get aligned facial images, evaluate facial pose, add user data, get registration info, remove user, verify user, identify person) (in JSON format).	Service to be installed on the local machine (robot, gateway, PC). Smartphones and tablets can connect to the gateway to use the available service	Robot, gateway or PC with Ubuntu or Windows operating systems. Connectivity to the private local network to give access to the provided service. If Intel RealSense camera is used, the functionality for user verification will have a better accuracy for spoofing attacks detection
OROFACE Orofacial gesture training tool (VICOM)	Proprietary API (to be included in D5.3) The API describes an endpoint which receives the input data (in JSON format) and then returns the pose quality assessment and the performed gesture's score (in JSON format).	Service to be installed on the local machine (robot, gateway, PC). Smartphones and tablets can connect to the gateway to use the available service	Robot, gateway or PC with Ubuntu or Windows operating systems. Connectivity to the private local network to give access to the provided service
Sleep Quality Assessment (TREE)	Rest API included as a component of the Big Data Platform in section 5.2.4.2. The analytic engines get raw data as input through POST and GET methods and returns another JSON file with the outputs of the sleep quality analytics.	In the analytic engines. The workflow is used to build a pipeline of the analytic to run within the Big Data Platform.	No special requirements assuming the analytic only has to be triggered once a day.
Physical Activity Intensity level (TREE)	Rest API included as a component of the Big Data Platform in section 5.2.4.2. The analytic engines get raw data as input through POST and GET methods and returns another JSON file with the	In the analytic engines. The workflow is used to build a pipeline of the analytic to run within the Big Data Platform.	The requisites are linked to the frequency with which the analytic has to be triggered.

	outputs of the physical activity analytics.		
Vitals Control (TREE)	Rest API included as a component of the Big Data Platform in section 5.2.4.2. The analytic engines get raw data as input through POST and GET methods and returns another JSON file with the control outputs.	In the analytic engines. The workflow is used to build a pipeline of the analytic to run within the Big Data Platform.	No special requirements assuming the analytic only has to be triggered once a day with the new measures recorded each day.
Routine & Anomaly Detection (TREE)	Rest API included as a component of the Big Data Platform in section 5.2.4.2. The analytic engines get raw data as input through POST and GET methods and returns another JSON file with the anomaly and routines reports.	In the analytic engines. The workflow is used to build a pipeline of the analytic to run within the Big Data Platform.	The requisites are linked to the frequency with which the analytic has to be triggered and the size of data to process into the analytic engines.
Emotion & Engagement detection (TREE)	PAL/KOM's Rest API to obtain JSON file with emotion and engagement metrics in a session.	Local deployment in robotic platform	These ROS nodes require signals of the start and end of processing within the robot ecosystem (dedicated topic).
People Detection (TREE)	Integrated ROS nodes without external interfaces. Dedicated ROS topics to internal communication in robotic platform.	Local deployment in robotic platform	No special requirements beyond the computing capacity to carry out the analysis in real time, as well as possible signs of initiation and termination of the use of the DS.
Fall Detection (TREE)	Integrated ROS nodes without external interfaces. Dedicated ROS topics to internal communication in robotic platform.	Local deployment in robotic platform	No special requirements beyond the computing capacity to carry out the analysis in real time, as well as possible signs of initiation and termination of the use of the DS.

DigiRoom (Omnitor)	Rest API with examples (to be included in D5.3)	Software-as a service (Saas), running as a Web application.	Client
"Memor-i" Game (SciFY)	Java Desktop application. It will be integrated to run as part of the ARI robot, by PAL. The app will make use of ARI's Text-To-Speech (TTS) REST API, in order to communicate effectively with the user. No REST API provided.	Software-as a service (Saas), running as a desktop application	Computer running Windows or Linux OS. Java v1.8 or higher is required.
"DiAnoia" mobile app (SciFY)	"DiAnoia" is an Android app that will run on an Android tablet which will be stored in the back pocket of the ARI robot (by PAL). DiAnoia will make use of ARI's Text-To-Speech (TTS) REST API, in order to communicate effectively with the user. No REST API provided.	Software-as a service (Saas), running as an Android app	A handheld device running Android version 6 or higher.
Kompaï robot (KOM)	<p>Developers can use the Kompaï robot in different ways:</p> <ul style="list-style-type: none"> • Using UDP API to control the robot at its lowest level • Using a ROS node to communicate with robot applications • - Using http requests API to access all robot functionality from external applications. 	The robot will already have digital solutions integrated, most of them running locally, except for those running on SHAPES Cloud, accessed through their respective APIs	<p>Have a clear space to install the charging station (against a wall with a 220V socket, 6A minimum) - An internet connection with a good signal (Wi-Fi or 4G key) - If the robot has to pass between 2 pieces of furniture, they must be spaced at least 1m apart - If the robot is called to pass through doors, the width of the door must be at least 830 mm - Have corridors at least 1m wide for better circulation of the robot - Do not have a clearance (door bottom for example) greater</p>

			<p>than 1.5 cm on the ground</p> <p>4G/5G connectivity for remote connection and support to the robots instead of the Wi-Fi. The robot functions in access mode so it is a produced Wi-Fi network -> module to switch from Wi-Fi to 4G inside the hospital depending on the requirements.</p>
Virtual Patient Scenarios (VPS) - AUTH	Not available	Open Labyrinth, open-source software, running as a web application hosted on AUTH premises.	No special HW needed. Software: php5.6, does not work with greater versions
Mobile Virtual Patients - MVP (AUTH)	Not available	A mobile progressive web app, it utilizes a back-end developed and hosted on AUTH premises	No special HW needed. Any mobile device will work.
MediMonitor (FNOL)	<p><u>Base URL</u> All endpoints can be accessed via the konzultace.fnol.cz host.</p> <p><u>Authentication</u> Compares submitted username/password with the database (passwords use SHA1 encryption). When authentication is successful, creates session with signed user's ID and one of 3 roles: Patient, Doctor, Administrator</p> <p><u>Methods:</u></p> <ul style="list-style-type: none"> POST / Pages/ LoginPage.aspx/ Submit My Card / Patient's Card <p>Displays patient's medical information. All POST</p>	<p>Mobile app developed by UHO and offered through store Google Play. Whole solution run on webserver Windows Server 2012 R2 on database MS SQL 2014 and on framework .NET Framework 4.6.1 (ASP.NET + C#). Client part is based on jQuery and Bootstrap. Videocall module uses open source platform Jitsi meet which is running on our UHO server.</p>	<p>Hardware requirements: does not require any special hardware - Handheld devices (for pilot testing is used Lenovo M 10 but it works also on other devices), running on Android version 5.0 and above. Telehealth portal runs on web-based browsers (Google Chrome, Mozilla Firefox, Microsoft Edge)HW/SW pre-requisites</p> <p>Software requirements: TM</p>

	<p>methods are called asynchronously via JavaScript functions.</p> <p><u>My Meetings</u></p> <p>Displays list of past and planned meetings (video calls), provides methods for adding, editing and removing the meetings.</p> <p><u>Scheduling</u></p> <p>Provides complex solution for patients' health records, tasks, medications/dosages, limits, exam rooms and user details. Most of POST methods are being used within modal window, triggered by JavaScript. For patient-role users, patient's ID is stored as a session variable on the server and cannot be modified by client.</p>		<p>web app runs in any modern web browser (desktop or mobile, cross-platform)</p>
Access Earth Platform (AELTD)	<p>Proprietary REST API which will be documented in D5.3. POST endpoint for authentication. GET endpoint delivers JSON of the user requested accessible places</p>	<p>Either through Android/iOS app access via single sign on. Ideal deployment in SHAPES platform is through a plugin which can be deployed via customer code integration (e.g. on local authority dashboard)</p>	<p>IOS 13 or above devices or Android 9.0 and above. Or alternatively any modern web browser as the plugin uses PWA architecture (e.g. service worker)</p>
Phyx.io (UCLM)	<p>This application will collect data during the exercise routine performance. This information will be made available via Restful API (not available yet)</p>	<p>The server will run on premise but it will expose a public interface for retrieving the performance data</p>	<p>Kinect (depth camera) plus a kiosk (big screen plus a processing device).</p>
Physical activity (UCLM)	<p>This application will collect data from the Xiaomi MiBand4. Data will be retrieved (periodically) from the smart band using the Bluetooth interface. Data will be stored in a database. These dataset can be remotely queried.</p>	<p>The server collecting the data plus the database server will run on-premises</p>	<p>Raspberry with Bluetooth interface (RPi 4) plus Xiami MiBand 4</p>

Smart Mirror (UCLM)	This application will collect data during the orofacial routine performance. This information will be made available via Restful API	The server will run on-premises but it will expose a public interface for retrieving the performance data	Our prototype of magic mirror
eHealthPass (Gnomon)	Proprietary REST API with documentation for supported modules. Documentation is available upon request	Software as a service, Play store (android, iOS)	Backend Services Memory: 16 GB, 4 CPU Docker Swarm or Kubernetes Client Mobile device with Android, IOS
COVIDShield (Gnomon)	Proprietary REST API with documentation for supported modules. Documentation is available upon request	Software as a service, Play store (android, iOS)	Backend Services Memory: 16 GB, 4 CPU Docker Swarm or Kubernetes Client Mobile device with Android, IOS
FINoT (FINT)	Rest API is documented in D4.1. Documentation is also available and provided to the consortium.	Software-as a service (Saas), running as a Web application.	No special requirements
ROSA (CH)	Phone application (D5.2)	Phone application with backend component (D5.2) as a tailored connection, proprietary of CH and currently hosted in their premises; The backend component manages users and telephone numbers; The backend component connects to Adilib (to be developed for use cases).	Android v8 or higher, iOS version v14 or higher