Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare

Project Acronym: **ASCLEPIOS**
Project Contract Number: **826093**

Programme**: Health, demographic change and wellbeing**
Call: **Trusted digital solutions and Cybersecurity in Health and Care to protect privacy/data/infrastructures**
Call Identifier: **H2020-SC1-FA-DTS-2018-2020**

Focus Area: **Boosting the effectiveness of the Security Union**
Topic**: Toolkit for assessing and reducing cyber risks in hospitals and care centres**
Topic Identifier: **H2020-SC1-U-TDS-02-2018**

Funding Scheme: **Research and Innovation Action**

Start date of project: 01/12/2018 Duration: 36 months

Deliverable:
D5.1 Technical Integration Points and Testing Plan

Due date of deliverable: 31/05/2020 Actual submission date: 23/06/2020

WPL: UBI

Dissemination Level: Public

Version: v1.0

ASCLEPIOS is supported by the H2020 Programme under contract no. 826093

Table of Contents

List of Figures and Tables

Status, Change History and Glossary

Status:	Name:	Date:	Signature:
Draft:	Giannis Ledakis	12/06/2020	Giannis Ledakis
Reviewed:	Gabriele Pierantoni	19/06/2020	Dr. Gabriele Pierantoni
Approved:	Tamas Kiss	23/06/2020	Tamas Kiss

Table 1: Status Change History

Version	Date	Pages	Author	Modification
V0.1	09/04/2020	7	Giannis Ledakis (UBI)	TOC and initial content
V0.1	24/04/2020	23	Giannis Ledakis, Panagiotis Parthenis (UBI)	Section 2 and 3 first content
V0.2	06/05/2020	33	Panagiotis Gouvas, Giannis Ledakis (UBI), Antonis Michalas (TUT)	First release with all sections in place
V0.3	14/05/2020	36	All partners	Defining the interfaces and complex flows
V0.4	21/05/2020	38	All partners	Updates on interfaces and UML schema, components description
V0.5	28/05/2020	41	Giannis Ledakis (UBI)	Section 2 and 3 updates and refactoring
V0.6	03/06/2020	41	Giannis Ledakis (UBI)	Last version of interfaces to be checked by partners
V0.7	09/06/2020	43	Giannis Ledakis, Konstantinos Theodosiou (UBI)	Added Complex testing flows sequence diagrams, and defined testing process
V0.8	17/06/2020	43	Giannis Ledakis (UBI)	Ready for review
V0.9	19/06/2020	43	Gabriele Pierantoni (UOW)	Review Comments
V1.0	23/06/2020	46	Giannis Ledakis (UBI)	Final for submission

Table 2: Deliverable Change History

Glossary

AAPEM	ABAC and ABE Policy Enforcement Mechanism
ABAC	Attribute Based Access Control
ABE	Attribute Based Encryption
APAM	ASLEPIOS Privacy Analytics Module
API	Application Programming Interface
CA	Certificate Authority or Certification Authority
CEEA	ASCLEPIOS Cybersecurity, Encryption and Access Analytics for Healthcare Providers
CI	Continuous Integration
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
CSP	Cloud Service Provider
EU	European Union
FE	Functional Encryption
ITEE	Isolated Trusted Execution Environment
PM	Person Month
STEP	Systematic Test and Evaluation Process
SSE	Symmetric Searchable Encryption
TEEPD	Trusted Execution Environment Platform Deployer
UML	Unified Modelling Language
WSGI	Web Server Gateway Interface
XACML	eXtensible Access Control Markup Language

Table 3: Glossary

Executive Summary

This deliverable provides the documentation of the **technical architecture** for the ASCLEPIOS platform and the **definition of the integration points** among the framework's mechanisms. Also, it contains the **integration plan** to guide the creation of the ASCLEPIOS framework and prepare the deployment of the whole platform in the scope of WP6. The **testing and evaluation plan** to verify the proper functioning and performance of the integrated ASCLEPIOS platform provided, in order to guide both the integration and the testing in the scope of the Work Package 5.

1 Introduction

In this document, we provide an integrated analysis of the software resources developed in WP2, WP3 and are the inputs for the implementation of the ASCLEPIOS platform. Based upon the conceptual architecture of D1.2, and the advancements in the technical work-packages since then, the technical architecture for the ASCLEPIOS platform and the specific integration points are defined.

To further guide the integration, a set of tools has been used already (Git, Gitlab, KeyCloak, Docker), while the integration plan to be followed has been prepared. Based on the integration plan, this document provided the initial testing plan that verifies the proper functioning of the integrated ASCLEPIOS platform, based on STEP methodology.

1.1 Objectives

The primary goal of this deliverable is to provide the integrated architecture description along with the integration and test plan that will guide the integration of ASCLEPIOS framework.

1.2 Relationship to ASCLEPIOS Deliverables

This deliverable is utilizing the information provided in the deliverable D1.2[1], but also the deliverables of WP2 and WP3. D5.1 provides the integration and testing plan for the ASCLEPIOS framework; therefore, it will be the basis for WP5 tasks regarding integration and testing, that will be reported in D5.2 and D5.3.

1.3 Organization

The rest of the document is organized as follows: In section 2, we present the technical architecture of ASCLEPIOS framework, by describing the components and their interfaces. In section 3, we provide additional information regarding the work to be performed and the plan for delivering the two platform releases. In section 4, we elaborate on the testing and evaluation of the ASCLEPIOS framework. Finally, section 5 provides a wrap-up and concludes the document.

2 ASCLEPIOS Integrated Framework Architecture

The ASCLEPIOS Integrated Framework encapsulates all the components needed for the creation of the main asset of ASCLEPIOS and allows the context-aware, attribute-based access and sharing of encrypted data. This ambitious goal will be achieved through different encryptions techniques (ABE, SSE, FE). This part is covered by the components in WP2 and WP3, while WP4 and WP6 outcomes, and will allow the proper deployment of the ASCLEPIOS Integrated Platform over secure and trusted cloud resources for the needs of the use cases covered by the ASCLEPIOS project.

2.1 Methodology

In this section, we introduce the methodology that has been used to generate the technical architecture of ASCLEPIOS. In general, several approaches were used to complete the definition of the architecture as it is provided in this document. Basically, relevant information was collected by interacting with other technical WPs in the project (mainly WP2, WP3 and WP4. All this information was incorporated in the conceptual architecture of ASCLEPIOS in D1.2, and we further elaborated for the creation of the technical architecture in the scope of WP5.

The technical architecture defines the technologies that are used to implement and support a solution that fulfils the information and data architecture requirements [2], and the same time it details the components and the communication between them.

2.2 The ASCLEPIOS Framework architecture

At D1.2, the high-level overview of ASCLEPIOS architecture was described, as illustrated in Figure 1.

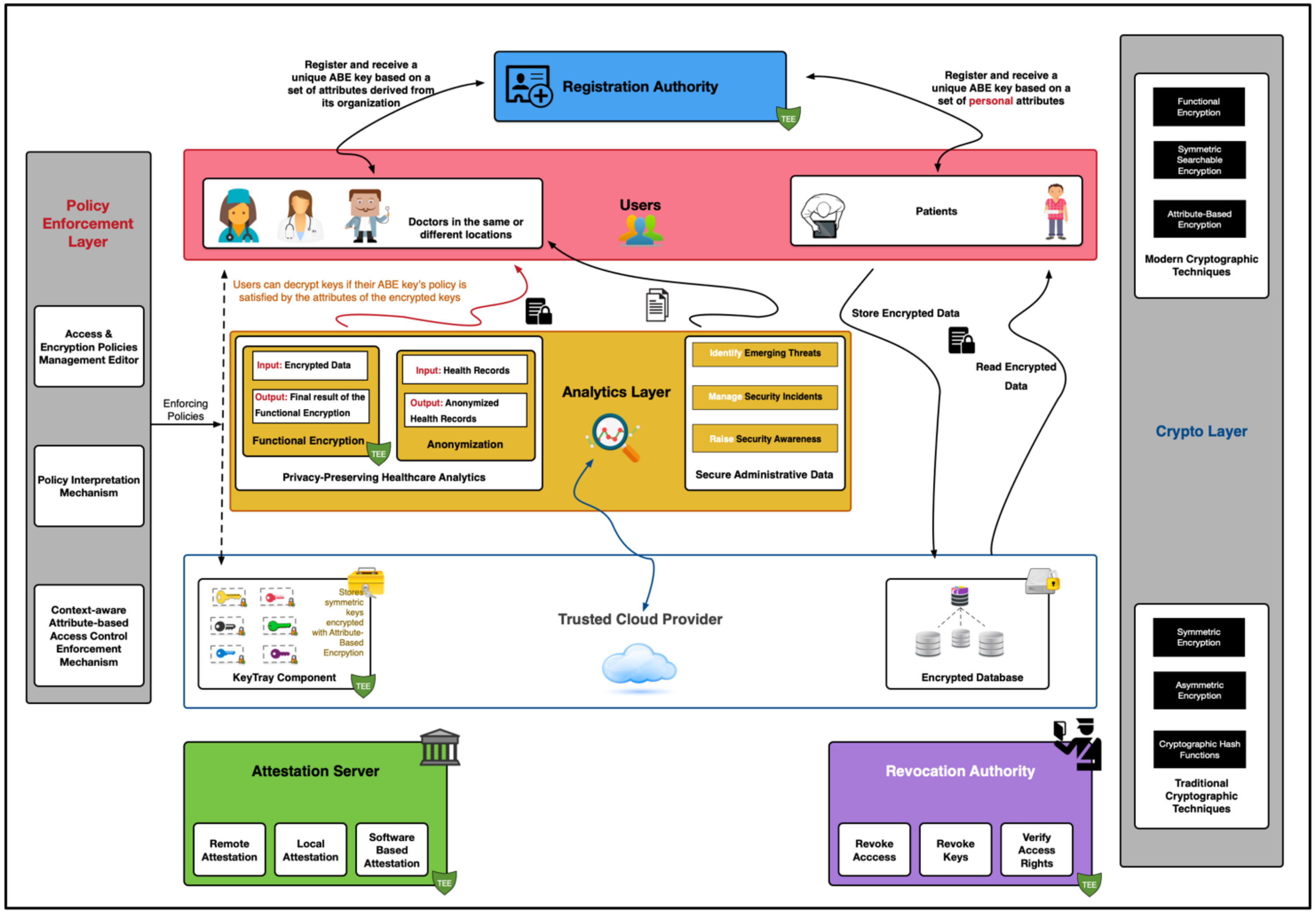

Register and receive a unique ABE key based on a set of attributes derived from its organization
Registration Authority
TEE
Register and receive a unique ABE key based on a set of personal attributes
Policy Enforcement Layer
Doctors in the same or different locations
Users
Patients
Access & Encryption Policies Management Editor
Users can decrypt keys if their ABE key's policy is satisfied by the attributes of the encrypted keys
Store Encrypted Data
Read Encrypted Data
Enforcing Policies
Input: Encrypted Data
Output: Final result of the Functional Encryption
Functional Encryption
Input: Health Records
Output: Anonymized Health Records
Anonymization
Privacy-Preserving Healthcare Analytics
Analytics Layer
Identify Emerging Threats
Manage Security Incidents
Raise Security Awareness
Secure Administrative Data
Policy Interpretation Mechanism
Context-aware Attribute-based Access Control Enforcement Mechanism
Stores symmetric keys encrypted with Attribute-Based Encryption
KeyTray Component
Trusted Cloud Provider
Encrypted Database
Attestation Server
Remote Attestation
Local Attestation
Software Based Attestation
Revocation Authority
Revoke Acccess
Revoke Keys
Verify Access Rights
Crypto Layer
Functional Encryption
Symmetric Searchable Encryption
Attribute-Based Encryption
Modern Cryptographic Techniques
Symmetric Encryption
Asymmetric Encryption
Cryptographic Hash Functions
Traditional Cryptographic Techniques

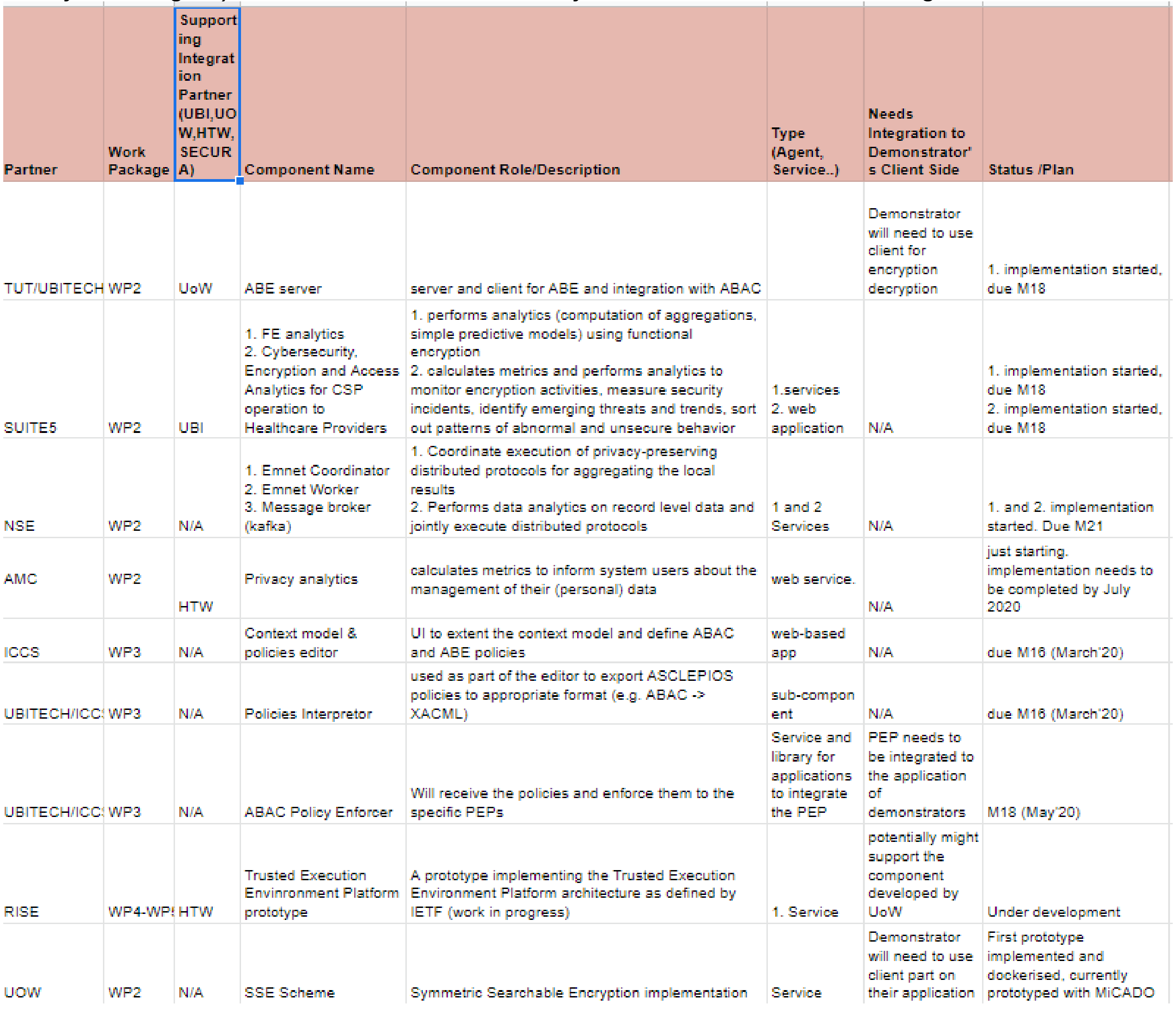
Figure 1: ASCLEPIOS overall architecture (source D1.2)

The whole ASCLEPIOS platform consists of the following eight discrete layers:

1. Trusted Cloud Provider,
2. Crypto Layer,
3. Analytics Layer,
4. Policy Enforcement Layer,
5. Registration Authority,
6. Users,
7. Attestation Layer,
8. Revocation Layer.

These layers represent the primary logical separation of the overall ASCLEPIOS platform. However, in this deliverable, we are focusing on the actual software resources (e.g. mechanisms, modules, components, services) that constitute the ASCLEPIOS framework and their interactions.

The first step is to identify the software resources of the ASCLEPIOS framework. Based on the architecture of D1.2, the advancements in the technical work packages WP2-WP4 we performed aggregation exercises in both WP5 and WP7. Initially, each partner was requested to define the components developing based on the work performed in the work packages. Based on the actual requirements collected in WP1, we were able to describe what type of software resource shall be created (e.g. a service, a backend only service, a library or an agent) and what kind of functionality it shall offer, as shown in Figure 2.



Partner	Work Package	Supporting Integration Partner (UBI,UOW,HTW, SECURA)	Component Name	Component Role/Description	Type (Agent, Service..)	Needs Integration to Demonstrator's Client Side	Status /Plan
TUT/UBITECH	WP2	UoW	ABE server	server and client for ABE and integration with ABAC		Demonstrator will need to use client for encryption decryption	1. implementation started, due M18
SUITE5	WP2	UBI	1. FE analytics 2. Cybersecurity, Encryption and Access Analytics for CSP operation to Healthcare Providers	1. performs analytics (computation of aggregations, simple predictive models) using functional encryption 2. calculates metrics and performs analytics to monitor encryption activities, measure security incidents, identify emerging threats and trends, sort out patterns of abnormal and unsecure behavior	1.services 2. web application	N/A	1. implementation started, due M18 2. implementation started, due M18
NSE	WP2	N/A	1. Emnet Coordinator 2. Emnet Worker 3. Message broker (kafka)	1. Coordinate execution of privacy-preserving distributed protocols for aggregating the local results 2. Performs data analytics on record level data and jointly execute distributed protocols	1 and 2 Services	N/A	1. and 2. implementation started. Due M21
AMC	WP2	HTW	Privacy analytics	calculates metrics to inform system users about the management of their (personal) data	web service.	N/A	just starting. implementation needs to be completed by July 2020
ICCS	WP3	N/A	Context model & policies editor	UI to extent the context model and define ABAC and ABE policies	web-based app	N/A	due M16 (March'20)
UBITECH/ICC!	WP3	N/A	Policies Interpretor	used as part of the editor to export ASCLEPIOS policies to appropriate format (e.g. ABAC -> XACML)	sub-component	N/A	due M16 (March'20)
UBITECH/ICC!	WP3	N/A	ABAC Policy Enforcer	Will receive the policies and enforce them to the specific PEPs	Service and library for applications to integrate the PEP	PEP needs to be integrated to the application of demonstrators	M18 (May'20)
RISE	WP4-WP!	HTW	Trusted Execution Envinronment Platform prototype	A prototype implementing the Trusted Execution Environment Platform architecture as defined by IETF (work in progress)	1. Service	potentially might support the component developed by UoW	Under development
UOW	WP2	N/A	SSE Scheme	Symmetric Searchable Encryption implementation	Service	Demonstrator will need to use client part on their application	First prototype implemented and dockerised, currently prototyped with MiCADO

Figure 2: Early identifying of ASCLEPIOS components and their connections

Based on this work, the following list of software components and mechanisms have been collected.

- ASCLEPIOS Models and PoLicies Editors (AMPLE)
- ASCLEPIOS ABAC and ABE Policy Enforcement Mechanisms (AAPEM)
- Attribute-Based Encryption (ABE) server and client
- Symmetric Searchable Encryption (SSE) server and client
- Functional Encryption (FE) Analytics Server and Client
- ASCLEPIOS Privacy Analytics Module (APAM)
- Privacy-preserving distributed statistical computation (Emnet)
- Cybersecurity, Encryption and Analytics on CSP operations for Healthcare Providers (CEAA)
- Trusted Authority

The mapping of the aforementioned components to the tasks Work Packages are presented in Table 4.

Software resource	Relevant Task(s)
SSE Server / Client	T2.1
ABE server / Client	T2.2 / T3.5
FE Server / Client	T2.3
CEAA	T2.4
EMNET	T2.5
APAM	T2.6
Model Editor part of AMPLE	T3.1
AMPLE	T3.2, T3.1, T3.3
Policy Interpreter of AMPLE	T3.3
AAPEM	T3.4
Trusted Authority	T2.1/ T2.2 / T2.3

Table 4: Services, Components and Mechanisms of ASCLEPIOS

Finally, it has to be stated that some of these components are combined to “assets” that the ASCLEPIOS consortium can exploit, as defined in deliverable D7.4[3].

2.2.1 Description of ASCLEPIOS components

Technical description of components is provided in the following subsections, covering their functionality and role in the overall platform and also providing implementation details. A more detailed description of the components is provided in the corresponding deliverables of WP2 and WP3.

2.2.1.1 ABE Service/Client

The ASCLEPIOS ABE Server and Client provide the implementation of the asymmetric encryption scheme that allows users with different keys to decrypt the same ciphertext. More precisely, in a CP-ABE scheme, a user can encrypt data based on a policy. In addition to that, each user gets a unique private key that also contains a list of personal attributes. By acquiring this private key, the user can decrypt the generated ciphertext *if and only if* the attributes that are attached to the key satisfy the policy that is bound to the underlying ciphertext. In ASCLEPIOS, CP-ABE will play an essential role in ***storing*** and ***managing the symmetric keys*** that will be generated by the users and will be used for the encryption of their medical records with an SSE scheme.

More details about ABE Server and Client can be found in D2.2 [4].

2.2.1.2 SSE Server/Client

The Symmetric Searchable Encryption (SSE) schemes can be utilized by the ASCLEPIOS demonstrators to achieve data sharing feature. The **SSE scheme implementation** offers a) a Server-side part of the SSE scheme (SSE Server) which is responsible for storing encrypted, sensitive data in MySQL[1] database on ASCLEPIOS. The SEE Server provides a RESTful API that is used by b) SSE Client. SSE Client is a JavaScript program running on the client-side, and implementing the required SSE functionality based on a JSON object that is given as input.

More details about ASCLEPIOS SSE can be found in deliverable D2.1[5].

2.2.1.2.1 Trusted Authority

The Trusted Authority component has been implemented for storing keys and metadata, which are needed for uploading/ searching data. The trusted authority, however, will be used as part of the general integration with other encryption schemes implemented in ASCLEPIOS (ABE and FE). It is implemented using Python, Django framework[2] and Gunicorn[3] Web Server Gateway Interface (WSGI) HTTP server.
More information for Trusted Authority can be found in the GitLab repository of SSE[11].

2.2.1.3 FE Server/Client

The ASCLEPIOS FE Analytics provides a set of services that can be used by healthcare providers to perform statistical computations over encrypted numeric data. The component has a server-client architecture. The client is responsible for the data encryption actions that happen locally on the premises of the users who provide their data. The server handles the key generation and distribution process, as well as the computation of the unencrypted result of the function that is applied over the encrypted data.

The encryption, decryption and key generation functionalities of the FE Analytics are implemented mainly in C++, although C is also used in some cases (depending on the FE scheme used). Flask[4], a lightweight WSGI web application framework, is used to implement the RESTful API through which the FE Analytics services can be invoked. A Trusted Authority is implemented to handle the key generation process using Intel SGX, but in the integrated platform, we consider the usage of a joint Trusted Authority for all components.

More details about ASCLEPIOS FE Analytics can be found in deliverable D2.3[6].

2.2.1.4 EMNET

Emnet is a distributed system for running statistical algorithms on confidential data divided among two or more different healthcare institutions where an algorithm runs on the combined data of the parties' databases without allowing any party to view the private data of a healthcare institution. The statistics generated from the combined data for a group of healthcare institutions are revealed, which are not sensitive information. Emnet supports statistical computations on both encrypted and unencrypted data distributed across healthcare institutions.

Emnet is implemented using Akka[5], Jackson[6], Django, Django Rest Framework and Apache Kafka[7], and will utilize the SSE scheme and FE analytics for computing on encrypted data.

[1] https://www.mysql.com/
[2] https://www.djangoproject.com/
[3] https://gunicorn.org/
[4] https://pypi.org/project/Flask/
[5] https://akka.io/
[6] https://github.com/FasterXML/jackson
[7] https://kafka.apache.org/

More details about Emnet will be provided in deliverable D2.4.

2.2.1.5 CEAA

The ASCLEPIOS Cybersecurity, Encryption and Access Analytics for Healthcare Providers (CEAA) component is responsible for delivering insights about data access patterns that emerge from the usage of CSP operations by healthcare providers who leverage new data encryption, decryption and fine-grained access control mechanisms, as the ones offered by the ASCLEPIOS framework. In order to provide its functionalities, CEAA monitors the logs created by these services and processes them to offer a better understanding regarding encryption and decryption activities, data access patterns, normal and abnormal behaviours, cyber threats and security incidents.

CEAA is implemented using Python and leverages the following open source technologies: Filebeat[8] (to monitor and ship the logs), Elasticsearch[9] for indexing and querying, Kibana[10] for the interactive visualizations.

More details about ASCLEPIOS CEAA can be found in deliverable D2.3[6].

2.2.1.6 APAM

The **ASCLEPIOS Privacy Analytics Module**, resulting from Task 2.6, is a web-based application that offers insights for data subjects about the usage of their data through ASCLEPIOS services that are provided to Healthcare Providers through a CSP.

This module collects all data access related logs for the various deployed services and monitors performed requests and corresponding responses. Transparency enhancing tools are defined based on these logs as a means to reveal the mentioned insights in a user-friendly fashion through an interactive interface.

More details about APAM will be provided in deliverable D2.4.

2.2.1.7 AMPLE

The purpose of AMPLE is to provide all necessary design-time tools for creating, maintaining and verifying access control policies of the ASCLEPIOS platform. AMPLE provides two editors for developing the corresponding ABAC and ABE policies (**ABAC Policies Editor** and **ABE Policies Editor)**. Furthermore, it provides a **Policy Validation component**, where policy developers can define rules for checking policy correctness, completeness or for security awareness.

Moreover, AMPLE provides a configurable, common vocabulary for application-related attributes, which can be further tailored to each application's needs. This is crucial since both access control methods (ABAC and ABE) relay on the use of attributes. This common vocabulary is called **Context-Aware Security Model (CASM)** and is stored in AMPLE's internal repository. Finally, it offers an editor for displaying and modifying CASM. The models created using AMPLE will serve the development of the Data Access Policies Interpretation and will be exploited by the AAPEM (ABAC Enforcement mechanism as well as by the ABE service).

Last, AMPLE includes a **Policy Interpreter** that translate ABAC policies to XACML format and ABE policies to appropriate XML format which are appropriate for implementation (i.e. input to AAPEM)

Lombok Java Library[11], Apache Jena[12] and Fuseki[13] have been used for the implementation of this component. More information for AMPLE can be found in deliverable D3.2[8].

[8] https://www.elastic.co/beats/filebeat
[9] https://www.elastic.co/elastic-stack
[10] https://www.elastic.co/kibana

2.2.1.8 AAPEM

The ABAC and ABE Policy Enforcement Mechanism (AAPEM) is responsible for the enforcement of ABAC policies for authorization based on the contextual information and also performing encryption and decryption using the ABE policies. AAPEM has been implemented as a Spring Boot[14] application that uses WSO2 Balana[15].

More information for AAPEM can be found in deliverable D3.3[9].

2.2.2 Component View of ASCLEPIOS

For the development of the integrated platform, we had to identify the interactions between the components and here provide more details about the actual components and interfaces, using a collaborative online spreadsheet. Based on this information, we present the component view of ASCLEPIOS framework. For readability purposes, we have split the UML schema into two parts and added mentions of the specific interfaces that are presented below in detail. In Figure 3 we present the two analytics components of ASCLEPIOS; all the other components of the framework are providing logs to them, but this is not depicted in the figures to avoid unnecessary links that would make the figure unreadable.

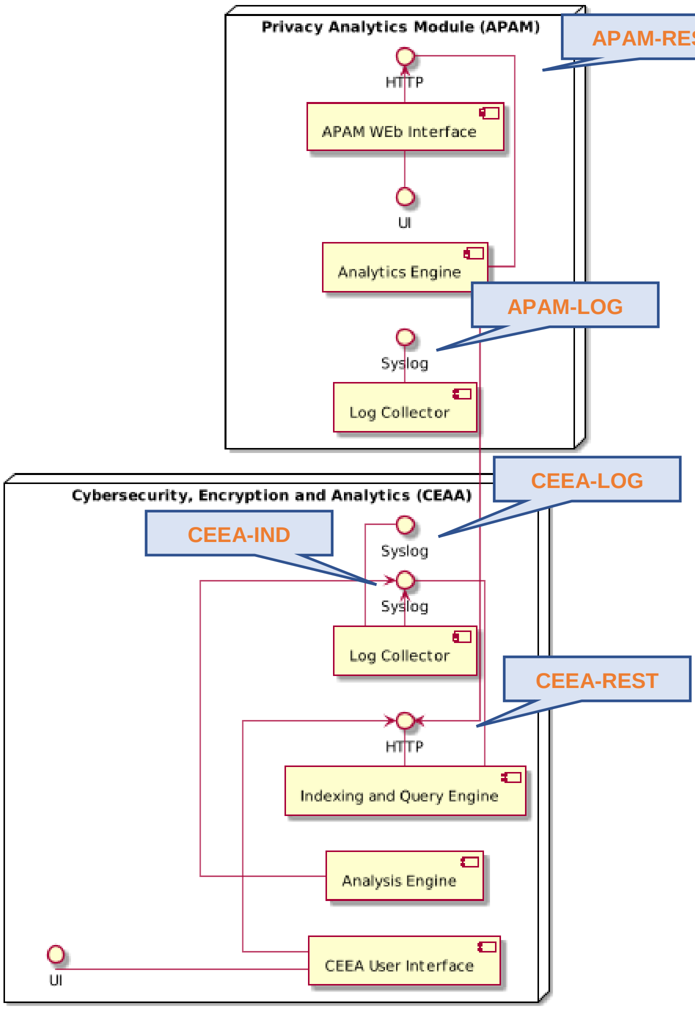


Figure 3: ASCLEPIOS technical architecture – development view (part 1/2)

[11] https://projectlombok.org/
[12] https://jena.apache.org/
[13] https://jena.apache.org/documentation/fuseki2/
[14] https://spring.io/projects/spring-boot
[15] https://github.com/wso2/balana

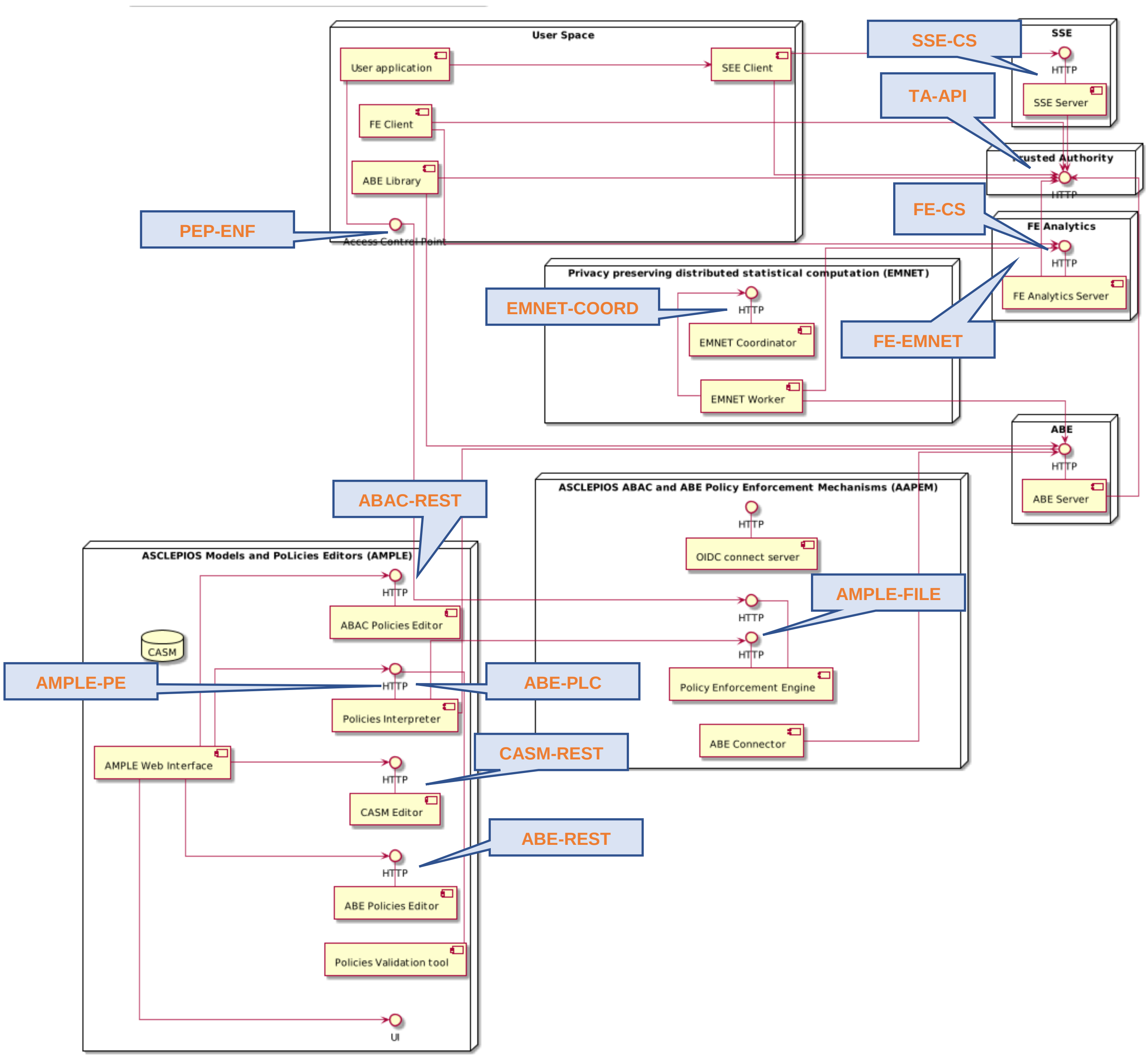


Figure 4: ASCLEPIOS technical architecture – development view (part 2/2)

The source of this figure and a link to view UML graph online is provided in Appendix I - Asclepios Technical Architecture UML Schema.

The following interfaces have been identified and presented in Figure 4:

- **ABE-CS:** ABE Server Interface for facilitating Client-Server communication.
- **SSE-CS:** SSE Server Interface for facilitating Client-Server communication.
- **FE-CS:** FE Server Interface for facilitating Client-Server communication.
- **TA-API:** Interface of the Trusted Authority component built initially for SSE usage.
- **EMNET-COORD:** Interface exposed by the EMNET Coordinator.
- FE-EMNET: Interface of FE used by EMNET Workers.
- **CEAA-LOG:** Syslog interface of CEAA for collecting logs.
- **CEAA-IND**: Internal interface of CEEA that provides the logs to the CEAA Indexing and Query Engine and also to APAM.

- **CEAA-REST:** Interface of CEAA Analytics Engine used by the web interface of CEEA.
- **APAM-LOG:** Syslog interface of APAM for collecting logs.
- **APAM-REST:** Interface of APAM Analytics Engine used by the web interface.
- **CASM-REST:** Interface of the Model editor that is used by the AMPLE UI.
- **ABAC-REST:** Interface of the ABAC policies editor that is used by the AMPLE UI.
- **ABE-REST:** Interface of the ABE policies editor that is used by the AMPLE UI.
- **ABE-PLC:** Interface of the ABE policies editor that is used ABE service.
- **AMPL-FILE:** Interface for the exchange of the policy file.
- **AMPLE-PE:** Connecting AMPLE policy Interpreter to the AAPEM.
- **PEP-ENF:** Communication between Policy Enforcement Point and the Policy Enforcement engine.

These interfaces are the backbone of the integration process and are described with more details in the following.

2.2.3 Description of ASCLEPIOS interfaces

In this section, we present with more details the information gathered about the interfaces required for the implementation of the integrated platform by defining the communication between the components developed in WP2 and WP4.
The following subsections describe these interfaces by detailing the following information:

- **Description:** describes the purpose of the interface.
- **Component providing the interface:** describes the component that is offering the described interface.
- **Consumer components:** describes the components that are using the described interface.
- **Type of interface:** REST, XML-RPC, GUI, Java API etc.
- **Input data:** describes data that is required by the described interface (e.g., Methods or Endpoints, values and parameters of the interface)
- **Output data:** describes the data that is returned by the described interface (e.g., the returned data of methods or REST call)
- **Constraints:** Any other constraints (e.g. specific prerequisites, data-types, encoding, transfer rates) which apply to the interface.
- **State:** Synchronous/Asynchronous, Stream
- **Responsibilities:** Partner that is responsible for the implementation and usage of the interface

2.2.3.1 ABE-CS Interface

Name: ABE-CS			
Description	Interface provided by ABE server used by a) the ABE Client and b) by the AMPLE Policies Interpreter to provide the ABE encryption policies		
Component providing the interface	ABE server		
Consumer components or External Entities	ABE Client, AMPLE Policies Interpreter		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	ABE setup	N/A	pub_key, master_key
	ABE keygen	pub_key, master_key, attributes	private_key

	ABE encrypt	pub_key, file, policies	Encrypted file
	ABE decrypt	Encrypted file, attributes, public key, private_key	file
Constraints	-		
Responsibilities	TUT, UBITECH		

Table 5: Details of the ABE-CS Interface

2.2.3.2 SSE-CS Interface

Name: SSE-CS			
Description	The interface of SEE Server that covers APIs relevant to uploading, search, and update data. Client part is implemented in JavaScript in order to be used by the applications of demonstrators. It includes: • API that allows a user to encrypt a Json object, then send its ciphertext to SSE server. • API that searches for encrypted data in SSE server by providing a search content. SSE server will return encrypted files which contain the searched keyword. • API that allows a user to update the whole or part of a Json object which is identified by its file_id. • API that allows a user to delete a Json object which is identified by its file_id. • API that allows a user to upload a shared key to Trusted Authority.		
Component providing the interface	SEE Server		
Consumer components or External Entities	SEE CLIENT, EMNET Worker		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /uploadData	(data,file_id,pwd1,pwd2)	True if uploaded successfully False if failed to upload data
	GET /search	(data,pwd1,pwd2)	Json object contains the number of found objects, and their content. {count: <number of found objects>, objects: <Array of Json

			objects, which contain decrypted data>}
	GET /updateData	(data,file_id,pwd1,pwd2)	True if updated successfully False if failed to update
	GET /deleteData	(file_id,pwd1,pwd2)	True if deleted successfully False if the provided file_id does not exist
	GET /uploadKeyG	(pwd1)	True
Constraints	N/A		
Responsibilities	UOW		

Table 6: Details of the SEE-CS Interface

For more information on SEE interfaces can be found in the dedicated page in the project's GitLab page [11].

2.2.3.3 FE-CS Interface

Name: FE-CS			
Description	This interface allows the upload of encrypted data to be used in the FE analytics		
Component providing the interface	FE Analytics Server		
Consumer components or External Entities	FE Analytics Client		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	POST /[function]/data	encrypted data	
Constraints	-		
Responsibilities	Suite5		

Table 7: Details of the FE-CS Interface

2.2.3.4 TA-API Interface

Name: PEP-ENF	
Description	The "interface" of Trusted Authority. During early development phase components used own implementations of Trusted Authority (for SSE, FE and ABE). We consider at this stage the implementation of TA provided by UOW as most complete and provide its API in this table.

Component providing the interface	Trusted Authority		
Consumer components or External Entities	ABE Server/Client, SSE Server/Client, FE Server/Client		
Type of Interface	REST API		
API description	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /api/v1/fileno/	TBD	TBD
	GET /api/v1/longrequest/	TBD	TBD
	GET /api/v1/search/	TBD	TBD
	GET /api/v1/searchno/	TBD	TBD
Constraints	-		
Responsibilities	UOW		

Table 8: Details of the PEP-ENF Interface

2.2.3.5 EMNET-COORD Interface

Name: EMNET-COORD			
Description	Interface exposed by the EMNET Coordinator and used for the registration of EMNET Workers		
Component providing the interface	EMNET Coordinator		
Consumer components or External Entities	EMNET Worker		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /register	WorkerID	Registration Result
	GET /health	WorkerID, Status	Status 200
	POST /report	WorkerID, Results	Status 200
Constraints	-		
Responsibilities	NSE		

Table 9: Details of the EMNET-COORD Interface

2.2.3.6 FE-EMNET Interface

Name: FE-EMNET	
Description	This interface handles the requests to perform FE functions over data
Component providing the interface	FE Analytics Server
Consumer components or	FE Analytics Client, EMNET Worker

External Entities			
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /[function]	identifiers of data on which to apply the function	{“value”:[result (numeric)]}
Constraints	-		
Responsibilities	Suite5		

Table 10: Details of the FE-EMNET Interface

2.2.3.7 CEAA-LOG Interface

Name: CEAA-LOG			
Description	This interface handles the collection of the logs generated by the ASCLEPIOS services		
Component providing the interface	CEEA Log Collector		
Consumer components or External Entities	CEEA		
Type of Interface	Syslog		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	One method constantly running to monitor and collect logs based on the filebeat.yml configuration file	paths to get the logs from	log contents
Constraints	-		
Responsibilities	Suite5		

Table 11: Details of the CEAA-LOG Interface

2.2.3.8 CEAA-IND Interface

Name: CEAA-IND			
Description	This interface is used to ship the logs to the CEAA Indexing and Query Engine		
Component providing the interface	CEEA Log Collector		
Consumer components or External Entities	CEEA Indexing and Query Engine, APAM		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	Filebeat sends the transactions directly to Elasticsearch by using the Elasticsearch HTTP API	Indexing and Query Engine IP and port	N/A
Constraints	-		
Responsibilities	Suite5		

Table 12: Details of the CEAA-IND Interface

2.2.3.9 CEAA-REST Interface

Name: CEAA-REST			
Description	This interface corresponds to the ElasticSearch Search API and enables the execution of queries over the indexed data using the Elasticsearch Query DSL, and also the API for storing results back in ElasticSearch(either per entry or with batches).		
Component providing the interface	CEEA Indexing and Query Engine		
Consumer components or External Entities	CEEA Web Interface, CEAA Analytics Engine, APAM		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	POST /[index name]/_search	The query is provided in JSON format as payload body of the request	Query results in JSON format
	POST /<index>/_update/<_id	The query is provided in JSON format as payload body of the request	Query results in JSON format
	POST /<index>/_bulk	The query is provided in JSON format as payload body of the request	Query results in JSON format
Constraints	-		
Responsibilities	Suite5		

Table 13: Details of the CEEA-REST Interface

2.2.3.10 APAM-LOG Interface

Name: APAM-LOG	
Description	This interface handles the collection of the logs generated by the ASCLEPIOS services
Component providing the interface	APAM Log Collector
Consumer components or External Entities	APAM
Type of Interface	Syslog / REST API

Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	REST interface to receive the logs on APAM	paths to get the logs from	log contents
Constraints	-		
Responsibilities	AMC		

Table 14: Details of the APAM-LOG Interface

2.2.3.11 APAM-REST Interface

Name: APAM-REST			
Description	Internal Interface that is used by the web interface of APAM		
Component providing the interface	APAM Analytics Engine		
Consumer components or External Entities	APAM Web interface		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	Component under implementation, so this internal list of methods to be reported in D2.4	N/A	N/A
Constraints	-		
Responsibilities	AMC		

Table 15: Details of APAM-REST Interface

2.2.3.12 CASM-REST Interface

Name: CASM-REST			
Description	Interface of the Model editor that is used by the AMPLE UI		
Component providing the interface	CASM		
Consumer components or External Entities	AMPLE UI		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /opt/attributes/	{attr_id} or “all”	<SchemaObject> or <SchemaObject> array
	GET /opt/attributes/{attr_id}/subattributes	{attr_id}	<SchemaObject> array
	GET /opt/attributes/search/by-name/{term}	{term}	<SchemaObject> array
	GET/opt/attributes/search/properties/by-attribute/{attr_id}	{attr_id}	<SchemaObject> array

	PUT /opt/attributes/	application/json SchemaObject	String message
	POST /opt/attributes/{attr_id}	{attr_id}	String message
	DELETE /opt/attributes/{attr_id}	{attr_id}	String message
	DELETE /opt/attributes/{attr_id}/all	{attr_id}	String message
Constraints	-		
Responsibilities	ICCS		

Table 16: Details of the CASM-REST Interface

2.2.3.13 ABAC-REST Interface

Name: ABAC-REST			
Description	Interface of the ABAC policies editor that is used by the AMPLE UI		
Component providing the interface	ABAC Policies Editor		
Consumer components or External Entities	AMPLE UI		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /opt/abac-policies/	{policy_id} or all	<AbacPolicy> or <AbacPolicy> array
	PUT /opt/abac-policies/	<AbacPolicy>	String message
	POST /opt/abac-policies/{policy_id}	{policy_id}, <AbacPolicy>	String message
	DELETE /opt/abac-policies/{policy_id}	{policy_id}	String message
	DELETE /opt/abac-policies/{policy_id}/all	{policy_id}	String message
	GET /opt/abac-policies/{policy_id}/rules	{policy_id}	<AbacRule> array
	GET /opt/abac-policies/rule/{rule_id}	{rule_id}	<AbacRule> array
	PUT /opt/abac-policies/rule/	<AbacRule>	String message
	POST /opt/abac-policies/rule/{rule_id}	{policy_id}, <AbacRule>	String message
	DELETE /opt/abac-policies/rule/{rule_id}	policy_id}	String message
	POST /opt/abac-policies/rule/{rule_id}	{rule_id}, <AbacRule>	String message
	DELETE /opt/abac-policies/rule/{rule_id}	{rule_id}	String message
Constraints	N/A		
Responsibilities	ICCS		

Table 17: Details of the ABAC-REST Interface

2.2.3.14 ABE-REST Interface

Name: ABE-REST			
Description	Interface of the ABE policies editor that is used by the AMPLE UI		
Component providing the interface	ABE Policies Editor		
Consumer components or External Entities	AMPLE UI		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /opt/abe-policies/	{policy_id} or “all”	<AbePolicy> or <AbePolicy> array
	PUT /opt/abe-policies/	<AbePolicy>	String Message
	POST /opt/abe-policies/{policy_id}	{policy_id}, <AbePolicy>	String Message
	DELETE /opt/abe-policies/{policy_id}	{policy_id}	String Message
	GET /opt/interpreter/abe-policy-to-text/{policy_id}	{policy_id}	String Message
Constraints	-		
Responsibilities	ICCS		

Table 18: Details of the ABE-REST Interface

2.2.3.15 ABE-PLC Interface

Name: ABE-PLC			
Description	Interface of the ABE policies editor that is used ABE service		
Component providing the interface	AMPLE Policy Interpreter		
Consumer components or External Entities	ABE service		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /opt/interpreter/abe-policy-to-text/{policy_id}	{policy_id}	Attributes for ABE, i.e.: (SecurityProtocolCertificate ='TLS' and (NetworkLocation_hasSubnet = '10.10.1.0/24' or PhysicalLocation_address = 'Building-1'))
Constraints	-		
Responsibilities	ICCS		

Table 19: Details of the ABE-PLC Interface

2.2.3.16 AMPL-FILE Interface

Name: AMPL-FILE			
Description	Interface for the exchange of the policy file and enforcement of policy		
Component providing the interface	AAPEM		
Consumer components or External Entities	AMPLE Policy Interpreter		
Type of Interface	Filesystem, SOAP		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	/addPolicy	policy version policyId	addPolicyResponse[16]
Constraints	File shall be validated and provided in XACML by the policy interpreter		
Responsibilities	UBITECH		

Table 20: Details of the AMPL-FILE Interface

2.2.3.17 AMPLE-PE Interface

Name: AMPLE-PE			
Description	Interface responsible for providing the file from AMPLE policy Interpreter to AAPEM		
Component providing the interface	AMPLE Policy Interpreter		
Consumer components or External Entities	AAPEM		
Type of Interface	REST		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	GET /opt/interpreter/abac-policy-to xacml/	Policy_id	XACML Policy (in text/xacml+xml/ format)
Constraints	-		
Responsibilities	ICCS/UBITECH		

Table 21: Details of the AMPLE-PE Interface

2.2.3.18 PEP-ENF Interface

Name: PEP-ENF	
Description	The “interface” for the creation of the access control point of facilitating the ABAC logic.
Component providing the interface	AAPEM

[16] https://is.docs.wso2.com/en/5.9.0/develop/entitlement-with-apis/#policy-administration-api

Consumer components or External Entities	Access Control Point of Application (PEP)		
Type of Interface	Java API, or REST API		
API description	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	pdp.evaluate (or POST /pdp for REST)	XACMLRequest	AbstractResult (Permit or Deny)
Constraints	JVM based application shall be used for the using the Java APIs		
Responsibilities	UBITECH		

Table 22: Details of the PEP-ENF Interface

3 Technical Integration and Planning

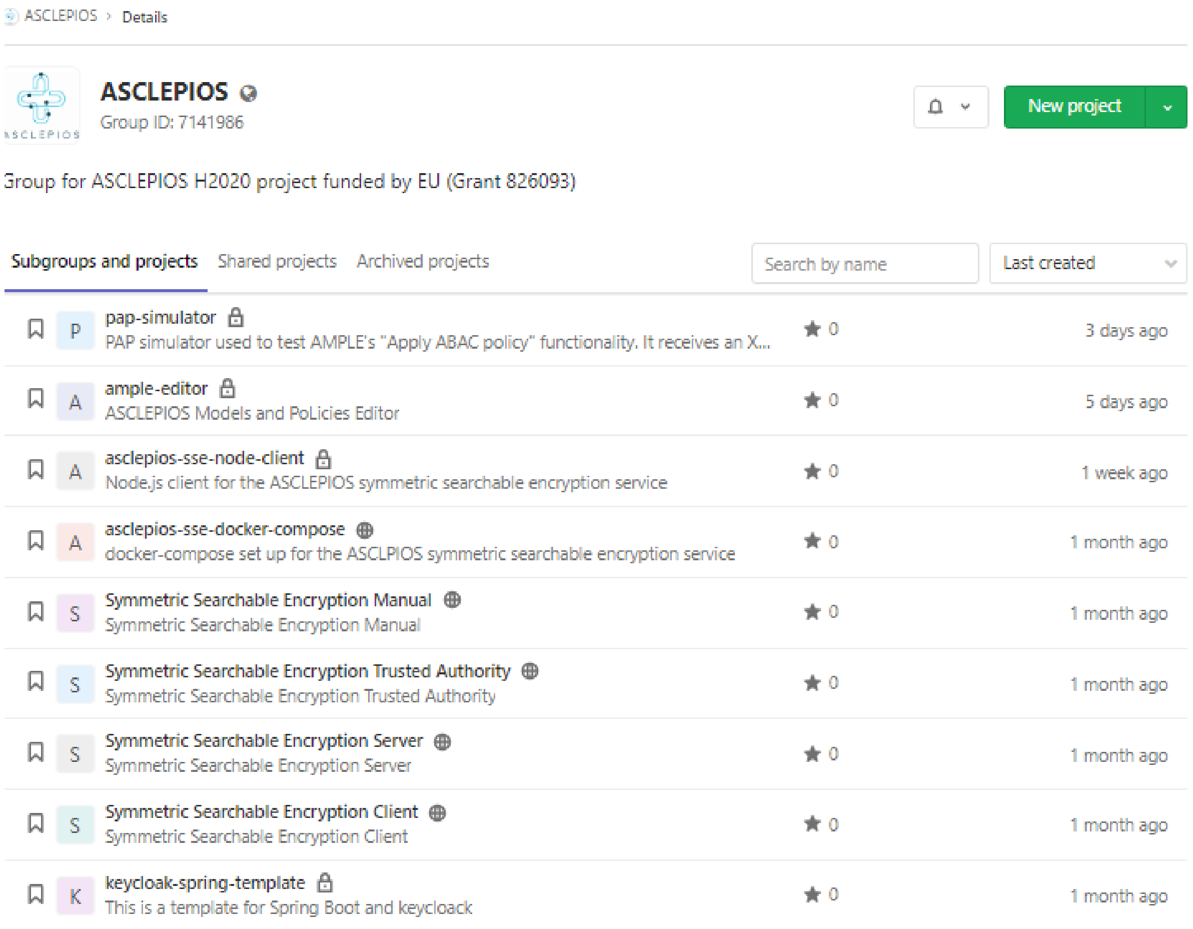
Based on the identified interfaces of the components, the integration of the framework shall take place. This means that a) the owners (or the responsible partners) of the components providing the interface shall deploy their components and make the interface accessible to the consumer components, and b) that owners (or the responsible partners) of the consumer interfaces will adapt their components accordingly in order to facilitate the usage of the interfaces. An integration plan has been prepared to guide the integration of the discrete framework's mechanisms and software components, as presented in section 3.3.

In addition to the integration plan, we present additional tools that are going for the integration (at section 3.1) and for the deployment (see section 3.2) of the framework.

3.1 Additional Modules and Supporting Tools

3.1.1 Code Level Integration

In the cases that multiple partners need to work on the same components, code-level integration is supported with a code repository that is available for all partners that need to work together or to store their component's code safety. The source code repositories are available at https://gitlab.com/asclepios-project, while a snapshot of the project group is provided in Figure 5.


ASCLEPIOS › Details

ASCLEPIOS
Group ID: 7141986

Group for ASCLEPIOS H2020 project funded by EU (Grant 826093)

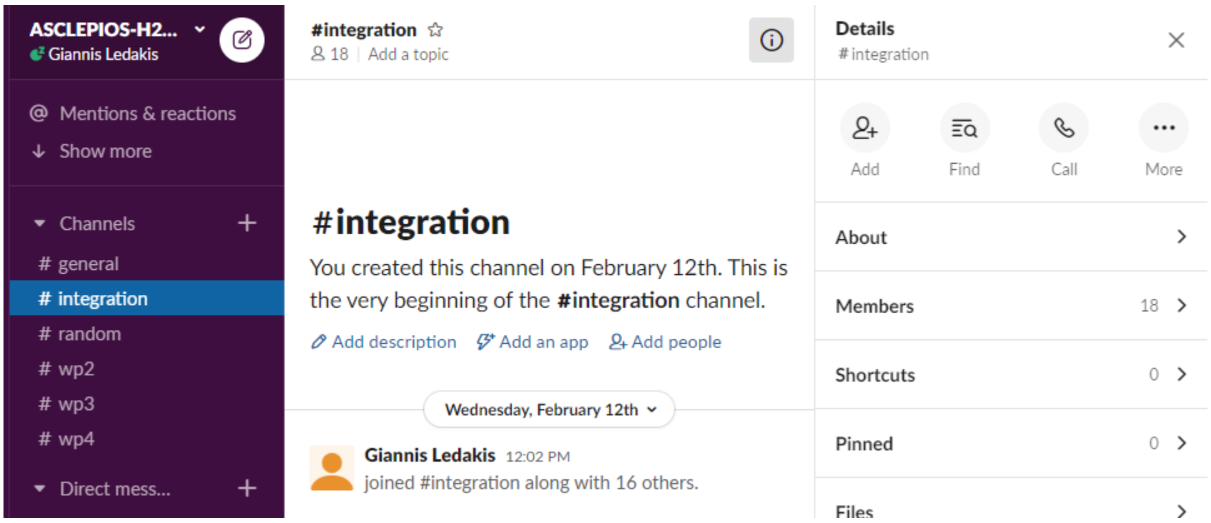
pap-simulator PAP simulator used to test AMPLE's "Apply ABAC policy" functionality. It receives an X...	★ 0	3 days ago
ample-editor ASCLEPIOS Models and PoLicies Editor	★ 0	5 days ago
asclepios-sse-node-client Node.js client for the ASCLEPIOS symmetric searchable encryption service	★ 0	1 week ago
asclepios-sse-docker-compose docker-compose set up for the ASCLPIOS symmetric searchable encryption service	★ 0	1 month ago
Symmetric Searchable Encryption Manual Symmetric Searchable Encryption Manual	★ 0	1 month ago
Symmetric Searchable Encryption Trusted Authority Symmetric Searchable Encryption Trusted Authority	★ 0	1 month ago
Symmetric Searchable Encryption Server Symmetric Searchable Encryption Server	★ 0	1 month ago
Symmetric Searchable Encryption Client Symmetric Searchable Encryption Client	★ 0	1 month ago
keycloak-spring-template This is a template for Spring Boot and keycloack	★ 0	1 month ago

Figure 5: ASCLEPIOS project group in GitLab

Furthermore, we suggest using a Continuous Integration (CI) scheme based on Gitlab. CI scheme will be documented further in deliverables D5.2 and D5.3

3.1.2 *Teams Communication – Slack*

For the easier communication during development and integration phase, we created a dedicated slack[17] group where technical partners and use case partners have joined.



#integration

You created this channel on February 12th. This is the very beginning of the **#integration** channel.

Giannis Ledakis 12:02 PM
joined #integration along with 16 others.

Figure 6: ASCLEPIOS Slack Workspace

In the same time, weekly calls have been performed for the technical progress of ASCLEPIOS. As most work packages are now finished, we plan to keep the Regular Calls of ASCLEPIOS with focus on integration.

3.1.3 *Keycloak as Authentication Proxy*

For the proper set of ASCLEPIOS platform, an Authentication server and a Trusted Authority must be present. Based on the study and experiments that were also made in the scope of WP3, we consider Keycloak[18] to be used as an Authentication Server. Keycloak is not an authorization server, but it is a powerful authentication proxy for micro-services and legacy systems. As such, it abstracts the functionality of identity extraction and identity verification for different systems and for different protocols. In parallel, it is able to map users and roles from existing legacy systems in what it calls authentication realms.

Keycloak is able to centralize the login-process of various systems through the implementation of many protocols such as oAuth2.0[13] and OpenIDConnect[14] (a.k.a. OIDC), and therefore it can be used a) for secure service-to-service communication for the services of ASCLEPIOS, b) for authentication of the users of the demonstrator and c) for securely providing the attributes of the demonstrators' users when ABAC is enforced.

3.2 Deployment of ASCLEPIOS Platform

We consider that each component that is developed or updated shall be created as a docker based container image. In comparison to a virtual machine that needs to include infrastructure configuration and the whole OS, a container image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files. A container is a runtime instance of an image—what the image becomes in memory when actually executed. In comparison to a Virtual Machine (VM) that is completely isolated, a container is partially isolated from the host environment, as it uses the kernel calls and commands of the host OS, but accessing host files and ports is possible only if configured to do so.

[17] https://slack.com/intl/en-gr/
[18] https://www.keycloak.org/

For dockerizing an application, a Dockerfile[19] is needed. The Dockerfile contains instruction on how to construct the instance of an image (called container). The Dockerfile contains directives like:

- defining a base image to be used (e.g. ubuntu 18.04). The developer can login and use ASCLEPIOS registry order to re-use the images already created as base images, by using the command (FROM *registry.gitlab.com/asclepios-project/framework/<image_name>:tag*)
- adding local files to the file system of the container
- commands to be executed upon initialization of the container (e.g. packages to be installed)
- Ports to be exposed
- Commands that launch the applications

3.2.1 ASCLEPIOS Docker Registry usage

A docker registry has been set for the purposes of ASCLEPIOS development using Gitlab. Developers that want to push an image to the repository should first tag it with the repository and then push it using the following commands.

```
$ docker login registry.gitlab.com
$ docker build -t registry.gitlab.com/asclepios -project/framework/<image_name>:<tag>  .
$ docker push registry.gitlab.com/asclepios -project/framework/<image_name>:<tag>
```

where:

- *image_name*: the name of the image. Typically, this is the same as the local image name.
- *tag:* Optional as parameter but need to properly support the continuous integration workflow. It is used for creating versions of the same image. For the first iteration version 0.1 will be used as tag for all images. If developer does not specify the tag, the docker will automatically set the tag latest.

Similarly, an image can be pulled by executing:

```
$ docker pull registry.gitlab.com/asclepios-project/framework/<image_name>:<tag>
```

Optionally, dedicated registry per component can also be used.

3.2.2 MICADO and TEEPD (Trusted Execution Environment Platform Deployer)

Regarding the deployment of ASCLEPIOS, the resources deployed in the scope of WP6 will be used, while we will also use the outcomes of WP4 regarding trusted execution. Isolated Trusted Execution Environment (ITEE) provides advanced protection for Healthcare applications. RISE developed TEEPD (Trusted Execution Environment Platform Deployer) based on IETF standards for deploying and managing workloads in TEEs. RISE and UoW implemented TEE using Software Guard Extensions (SGX). SGX allows defining private regions of memory, called enclaves, whose contents are protected and unable to be either read or saved by any process outside the enclave itself. SGX provides mechanisms to manage workloads in an ITEE instance including actions to deploy and update, start and stop the workload process, as well as clean up the environment once the workload is not needed.

UoW developed the MiCADO framework[20] (Microservices-based Cloud Application-level Dynamic Orchestrator) in the H2020 COLA (Cloud Orchestration at the Level of Application)

[19] https://www.docker.com/
[20] https://micado-scale.eu/

project. MiCADO is a highly customizable multi-cloud orchestration and auto-scaling framework for Docker containers and Virtual Machines orchestrated by Kubernetes.
UoW extended MiCADO to incorporate support for TEE.

Several ASCLEPIOS security services, such as the SSE Implementation and the Trusted Authority (TA) can be deployed inside SGX environment to protect data.

3.3 Integration Planning

Following the aspects presented above, internal artefacts produced by the technical work packages WP2 and WP3 will be continuously integrated, and WP4 advancements will allow the deployment of the platform.

3.3.1 Development and Integration Iterations

Iteration #: 2020-06-01 to 2020-06-30   Iteration lead: Giannis L.
Prototype feature:
Expected partner contributions: ICCS: UBITECH: UOW: TUT: AMC: SUITE5: NSE:
Result:
Working prototype feature description:
Presentables (what and how):.
Open questions: N/A
Open problems: N/A
Learnings/decisions: TBD

The Regular Calls of ASCLEPIOS with focus on integration will be used as checkpoints through the usage of iterations.

Finally, It has to be stated that as some technical partners of WP2 are not participating in WP5 or participating with a very small number of PMs. This could cause and issue by not having those partners able to adapt their components for integration purposes, but we have already identified this issue we have agreed on a scheme

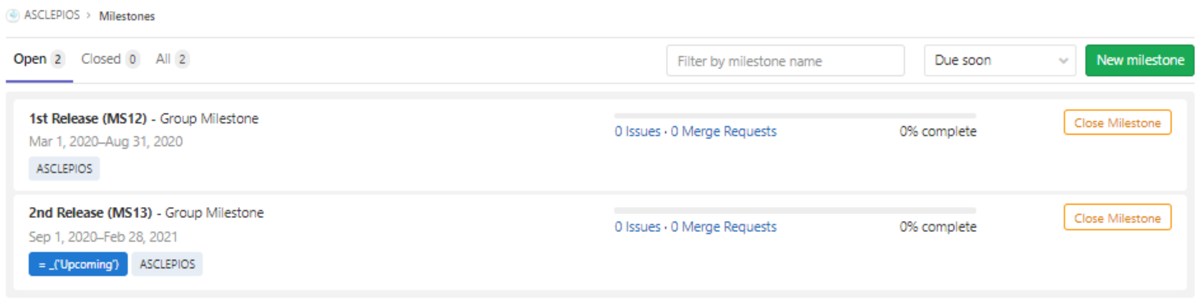
3.3.2 Platform Releases

In the same time, we plan two major releases of the ASCLEPIOS integrated platform for M21 and M27. The aim is to integrate the components into ASCLEPIOS platform and instantiate it for each use cases.

The first major release of ASCLEPIOS will include the integration of the basic components developed in the technical work packages. The goal is to present basic capabilities integrated until M21. Then by M27, further enhancements or modifications will be provided in the final release of the ASCLEPIOS platform. However, smaller iterations with changes in the architectural design and the development will be used through the project duration.

Based on the plan of having to two releases of ASCLEPIOS framework, the consortium had to make decisions regarding the specific functionalities that will be supported on each release, in order to coordinate the development, iteration and testing process.

To properly support this during development, we have created on the project GitLab dedicated milestones in order to be able to plan and track the advancements on all developed components and the integrated platform.


ASCLEPIOS › Milestones

Open 2 Closed 0 All 2 | Filter by milestone name | Due soon | New milestone

1st Release (MS12) - Group Milestone
Mar 1, 2020–Aug 31, 2020
ASCLEPIOS
0 Issues · 0 Merge Requests 0% complete Close Milestone

2nd Release (MS13) - Group Milestone
Sep 1, 2020–Feb 28, 2021
= _('Upcoming') ASCLEPIOS
0 Issues · 0 Merge Requests 0% complete Close Milestone

Figure 7: ASCLEPIOS Milestones as part of the development and integration plan

3.3.2.1 1st Platform Release

The first release of ASCLEPIOS is due on M21. The goal is to complete the first cycle of integration. The actual deadline is dictated by the deliverable D5.2 that will document the platform status and provide installation and usage instructions. As already stated, the first release will focus on the integration of the basic components of the platform and will allow testing and evaluation at the perspective of the use cases until the second release is officially provided.

The following points shall be covered in the first release.

- Full integration and testing of flows for the creation and enforcement of policies for ABAC, and ABE and ABAC Synergy
- The first integration of SSE
- The first integration of FE
- The first integration of analytics services (CEEA)
- Define with details the flows for ABAC/SSE Synergy and ABAC/FE Synergy

3.3.2.2 2nd Platform Release

The second platform release will be provided on M32. It will include the full implementation of the components in order to allow full testing and evaluation at the perspective of the use cases until M36.

The final release of the integrated platform will include:

- Full integration and testing of the flows for ABAC/SSE Synergy and ABAC/FE Synergy
- Full integration of logs to the analytics services
- Full integration of Keycloak for service to service communication, and user authentication
- User interface finalization
    - Containerization of ASCLEPIOS services for deployment with WP6 toolkit

4 Testing and Technical Evaluation Plan

This section presents the platform testing as part of the overall evaluation strategy in the context of ASCLEPIOS. It has to be mentioned that the technical testing and evaluation will be based on STEP (Systematic Test and Evaluation Process), a well-established industry methodology for testing and evaluation activities in information technology and software projects. The testing will be performed to verify the proper functioning and performance of the integrated ASCLEPIOS platform, in the scope of task 5.3, and the results will be reported in deliverables D5.2 and D5.3.

STEP assumes that the total testing job is divided into levels during planning. A level represents a particular testing environment (e.g., unit testing usually refers to the level associated with program testing in a programmer's personal development library). Simple projects, such as minor enhancements, may consist of just one or two levels of testing (e.g., unit and acceptance), while for complex projects, more levels might be needed(e.g., unit, function, subsystem, system, acceptance testing, alpha, beta, etc.) [10].

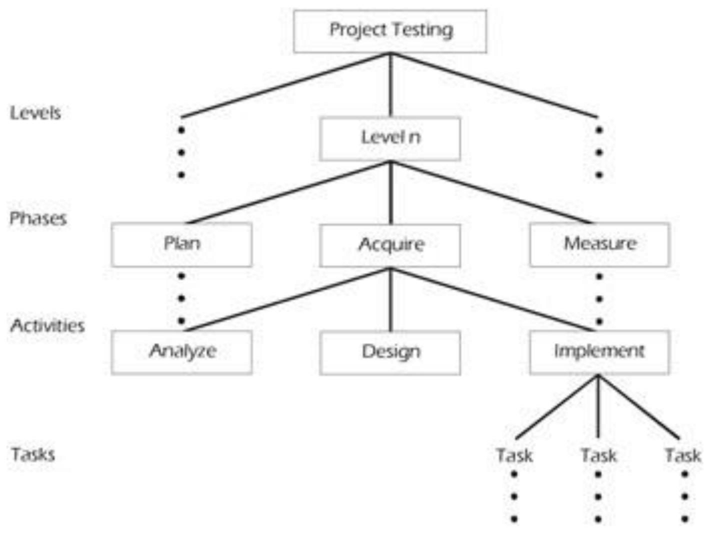


Figure 8: Activity Timing at each level of test

STEP provides a model that can be used as a starting point in establishing a detailed test plan, but it is intended to be tailored and revised, or extended to fit each particular test situation.

The three major phases in STEP that are employed at every level include: **planning** the strategy (selecting strategy and specifying levels and approach), **acquiring** the testware (specifying detailed test objectives, designing and implementing test sets), and **measuring** the behaviour (executing the tests and evaluating the software and the process). The phases are further broken down into eight major activities, as shown in Figure 8.

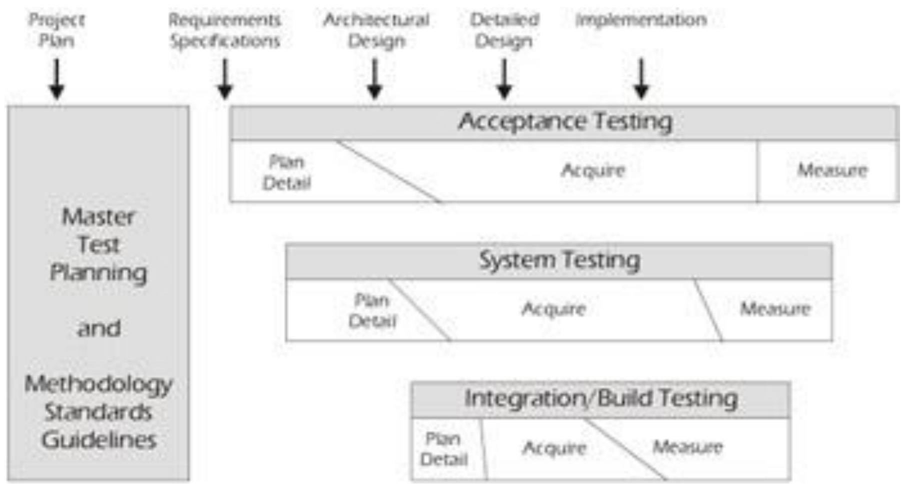


Figure 9: STEP Phases

STEP specifies when the testing activities and tasks are to be performed, as well as what the tasks should be and their sequence, as shown in the figure above. The timing emphasis is based on getting most of the test design work completed before the detailed design of the software. The trigger for beginning the test design work is an external, functional, or black box specification of the software component to be tested. For this reason, it is important to plan early the tests to be performed.

In ASCLEPIOS, we define the following facets of testing:

- Unit testing that can be performed by the separate development teams when new functionalities are developed.
- Integration testing performed by the development teams in order to test the smooth co-operation between the various layers and components. The integration tests and also any unit tests that will be created for the project validation will be continuously executed based on continuous integration (CI) scheme
- Testing of a set of advanced scenarios based on demonstrators' needs.

These testing facets are presented in the following sections.

4.1 Unit Testing

An important part of both the integration and the validation process is the execution of unit tests. Unit tests are the tool to test the functional modules of software. A suitable unit test is applied to the piece of code without any dependencies on other code parts. Therefore, the developer of the particular layers will test their components by means of unit tests before integrating them into the full application. In the case of ASCLEPIOS framework, development is based on the development of standalone components but also on the adaptation and integration of existing components. Therefore, unit testing at the lowest level not a primordial part of the general testing methodology of ASCLEPIOS, as the main focus is the integration testing.

Unit testing will be used as an additional mechanism of validating the developed code, as a task that each component developer can use in order to verify proper functionality before the integration of the component in ASCLEPIOS Framework. Usually, all unit-tests are executed during the build-process, unless they are defined to be ignored (marked as @Ignored for Java applications). This practically means that each release of a component has is guaranteed regarding its stability and the same time allows developers to better control the level of test coverage on their components.

The exercise of creating unit tests is still in early stages and the documentation of Unit tests will be provided with more details for all components in deliverable D5.3.
.

Table 23 – Unit Test documentation example

Unit Test Case Documentation Form	
Unit Test Reference Code	#UT1
Component	AAPEM
Tester	Junit
Short Description	
This test case is responsible for testing policy enforcement.	
Input Data	
Output Data	
Success response code	

Apart from the tests that guarantee the functional correctness of the components, it is important to make tests at the integration level for a complete testing and validation process. This means that integration tests shall be created and used for all identified interfaces and to some major platform functionalities. This can be done using unit testing on the methods that are implementing the integration, in order to make them part of continuous integration and continuous testing process.

4.2 Testing for the Integrated Platform

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing in ASCLEPIOS can also be seen as an extension of unit testing. The main idea of integration testing is to start from two components to test the interface between them. In some cases, more than two components can participate in a common test. Eventually, this process will be expanded in order to test all the integrated components of the platform.

The goal of integration testing is to identify problems that occur when components are combined. By using a test plan that suggests the usage of unit tests before combining components, the errors discovered when in integration tests are most probably related to the interface between them. This method reduces the number of possibilities of errors to a far simpler level of analysis.

In general, integration testing can be done in a variety of ways, but the following are three of the most common strategies:

- The top-down approach of integration testing requires the highest-level modules to be tested and integrated first. This allows high-level logic and data flow to be tested early in the process, and it tends to minimize the need for drivers. However, the need for stubs complicates test management, and low-level utilities are tested relatively late in the development cycle. Another disadvantage of top-down integration testing is its poor support for early release of limited functionality.
- The bottom-up approach requires the lowest-level units to be tested and integrated first. These units are frequently referred to as utility modules. By using this approach,

utility modules are tested early in the development process, and the need for stubs is minimized. The downside, however, is that the need for drivers complicates test management and high-level logic and data flow are tested late. Like the top-down approach, the bottom-up approach also provides poor support for early release of limited functionality.

- The third approach, sometimes referred to as the umbrella approach[12], requires testing along with functional data and control-flow paths. First, the inputs for functions are integrated into the bottom-up pattern discussed above. The outputs for each function are then integrated in a top-down manner. The primary advantage of this approach is the degree of support for the early release of limited functionality. It also helps minimize the need for stubs and drivers. The potential weaknesses of this approach are significant, however, in that it can be less systematic than the other two approaches, leading to the need for more regression testing.

For the integration testing of ASCLEPIOS, we chose the last option (umbrella approach), as it combines the best of both approaches. It allows all participating entities, to execute simultaneously multiple testing in several components. In the next section, the basic integration tests that have been identified and tested so far, are presented.

4.2.1 Integration Tests

As it is important for ASCLEPIOS to ensure the proper integration of the components, tests that are based on functions that cover different components will be used. In these tests methods from different components are combined in order to achieve the needed functionality, so the focus is given to the combination of pieces that create a basic integrated functionality.

Table 24 – Identified and Planned Integration Tests

Test ID	Test	Interface(s) Tested	Components Used	Short Description
IT1	ABE basic test	ABE-CS	ABE Client, ABE Server	Testing the ABE Client-Server communication
IT2	SSE basic test	SSE-CS	SSE Client, SSE Server	Testing the SSE Client-Server communication
IT3	FE basic test	FE-CS	FE Client, FE Server	Testing the FE Client-Server communication
IT4	EMNET basic test	EMNET-COORD	EMNET Coordinator, EMNET Workers	Testing the EMNET Coordinator-Workers communication
IT5	FE/EMNET test	FE-EMNET	EMNET Workers, FE Server	Testing the EMNET FE Server communication
IT6	CEAA basic log test	CEAA-LOG	CEAA, all components	Testing the logs Syslog interface of CEAA for collecting logs\
IT7	APAM- basic log test	APAM-LOG	APAM, all components	Testing the proper log collection throuh the Syslog interface of APAM
IT8	APAM/CEEA test	APAM-CEEA	APAM, CEEA	Testing the communication between APAM and CEEA
IT9	CASM editor test	CASM-REST	AMPLE UI, CASM	Testing of AMPLE UI and communication with

				CASM
IT10	ABAC editor test	ABAC-REST	AMPLE UI, ABAC policies editor	Testing of AMPLE UI and ABAC policies editor
IT11	ABE editor test	ABE-REST	AMPLE UI, ABE policies editor	Testing of AMPLE UI and the ABE policies editor
IT12	Policy file exchange test	AMPL-FILE	AAPEM, AMPLE Policy Enforcer	Testing the proper exchange of the policy file
IT13	Policy interpretation test	AMPLE-PE	AAPEM, AMPLE Policy Enforcer	Testing the AMPLE policy Interpreter connectivity with AAPEM
IT14	ABAC enforcement test	PEP-ENF	AAPEM Policy Enforcement Engine, Access Control Point	Testing the proper Communication between Policy Enforcement Point and the Policy Enforcement Engine

The results of these tests will be reported in D5.2 and D5.3, based on the actual integration of the components at each stage.

4.3 ASCLEPIOS Complex Flows testing

Based on the discussions we had in the consortium, it is important to allow the combination of the offered access control mechanism and the security/ encryption mechanisms. For this reason, dedicated tests will be created for the validation of more complex flows. The results of these tests will be reported in D5.2 and D5.3, according to the plan presented in section 3.3.2. These flows will also be evaluated through the pilot usage of the platform; our goal in the scope of this deliverable is to assure the proper function of the platform based on the need to support such scenarios.

4.3.1 ABAC based access control

Authorization based on ABAC will be tested in the scope of the testing of the integrated platform. As presented in Figure 10, the flow to be tested includes the creation of a new policy through the UI and then test the proper enforcement on an access control point of a test application. As an additional step, we can update the policy in order to ensure that the enforcement is dynamically applied.

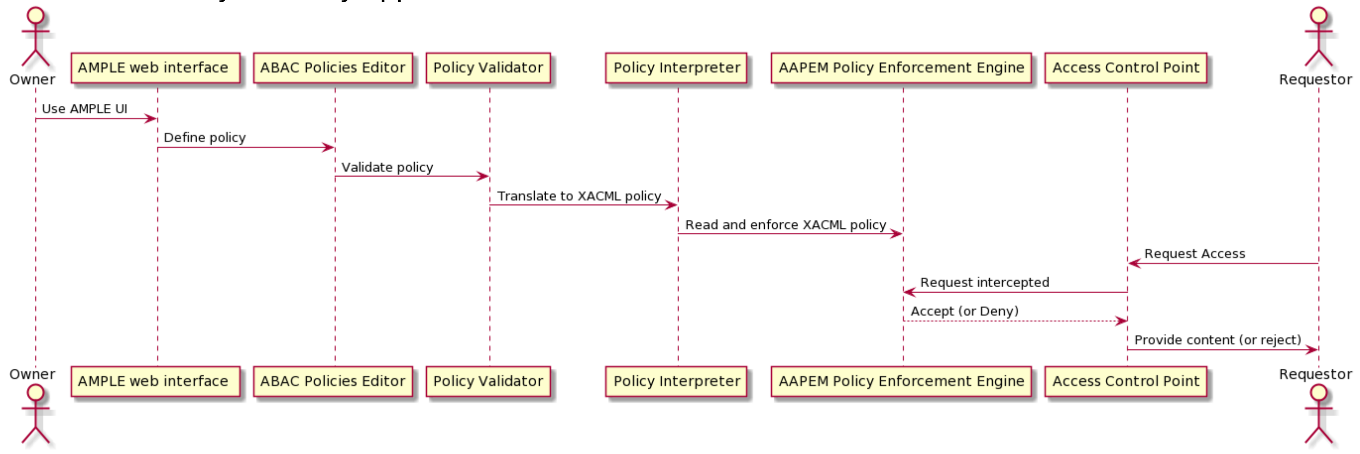


Figure 10: ABAC Policy Creation and Enforcement flow for testing

4.3.2 ABAC and ABE based flow

In the context of ASCLEPIOS, we propose the combined use of ABAC and ABE policies for authorization in ABE decryption. We consider a two-step process where ABAC policy is first applied on access attempts to resources (either data or functionality). Subsequently, if an ABAC permit is granted, ABE policy is applied in order to recover the resource symmetric decryption key. This process is shown in Figure 11.

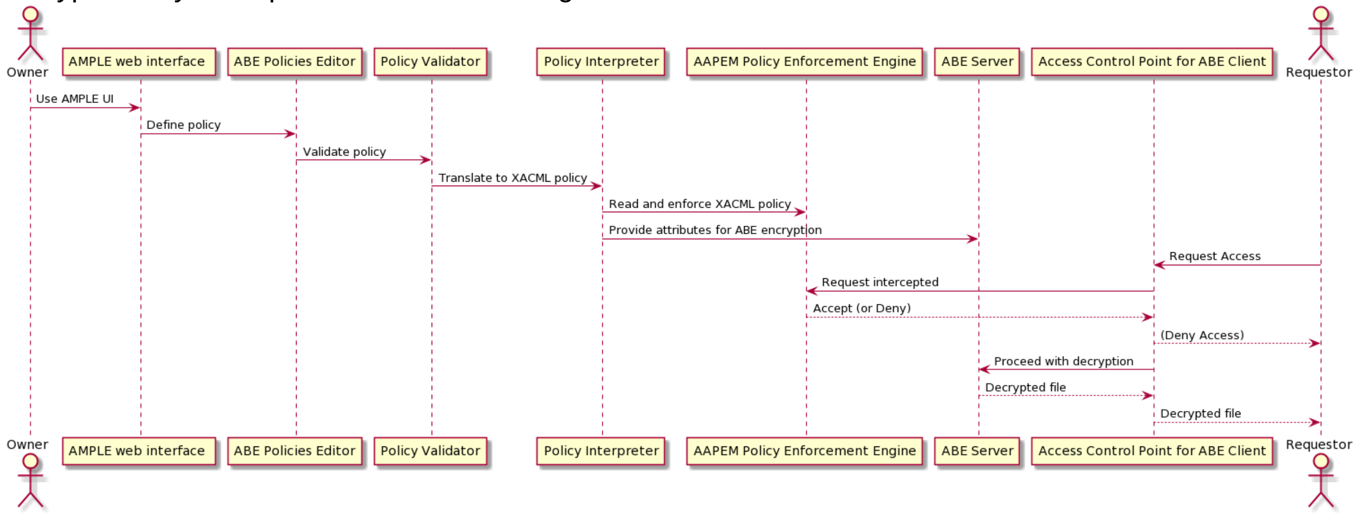


Figure 11: ABAC + ABE flow for testing

4.3.3 ABAC and SSE based flow

Similar to the flow presented in 4.3.2, ABAC will also be used for SSE usage. We consider a two-step process where ABAC policy is first applied on access attempts to SEE server and subsequently, if an ABAC permit is granted, SEE service is used. This process is shown in Figure 12.

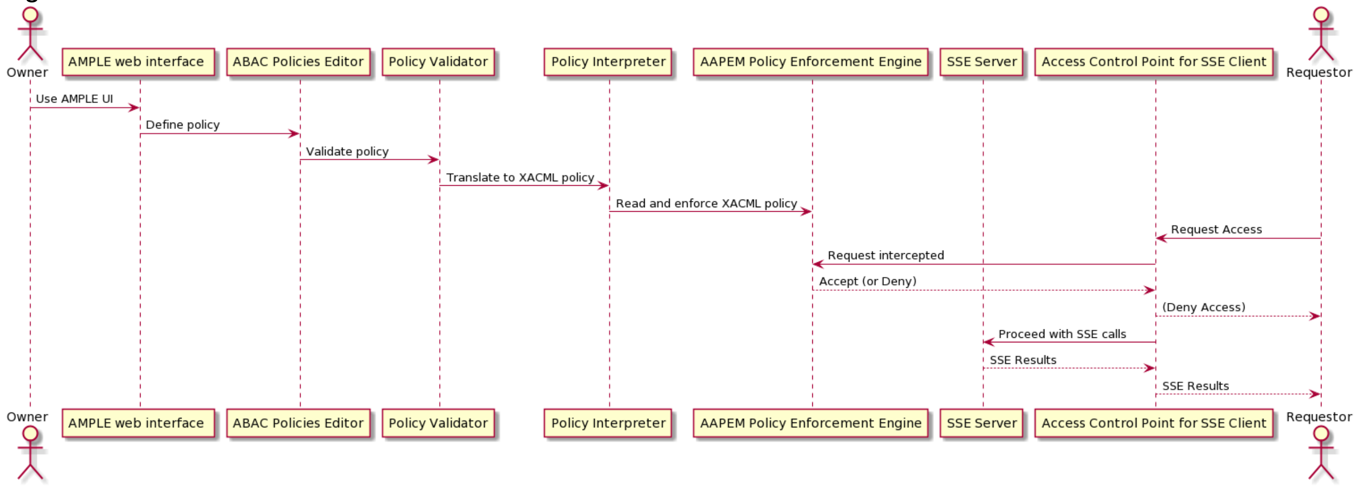


Figure 12: ABAC + SSE flow for testing

4.3.4 ABAC and FE based flow

Similar to the flow presented in 4.3.2, ABAC will also be used for FE usage. We consider a two-step process where ABAC policy is first applied on access attempts to FE server and subsequently if an ABAC permit is granted, FE service is used. This process is shown in Figure 13.

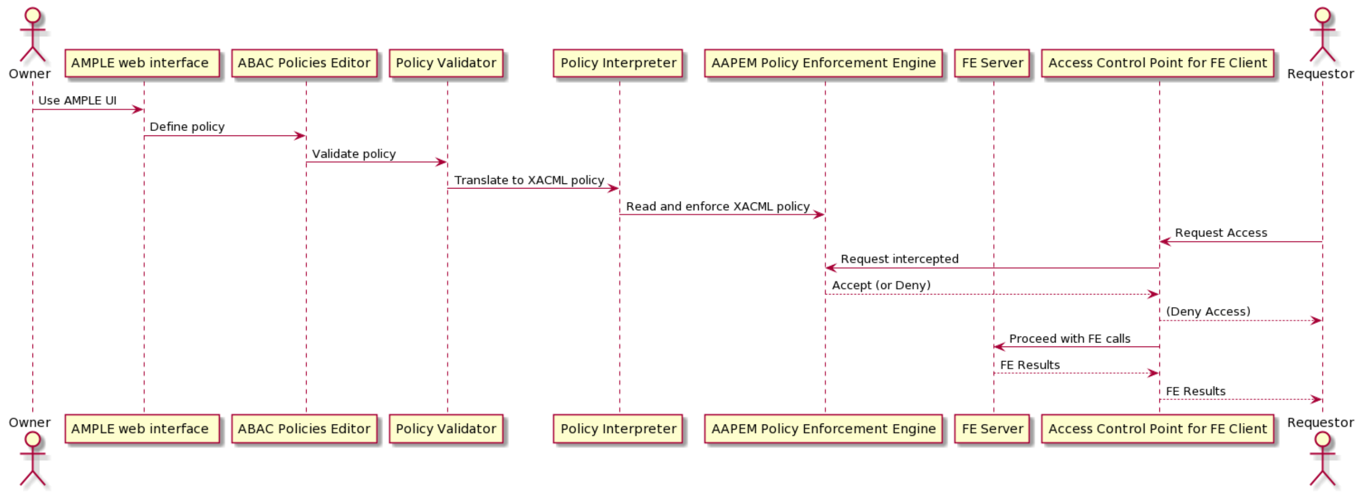


Figure 13: ABAC + FE flow for testing

5 Conclusions

This deliverable is the continuation of the work reported in the deliverable D1.2, where the initial, conceptual view of the architecture was presented. Here our work was focused on the analysis and refinement of every possible interaction in the system architecture; therefore, we provided an updated view of the components and the interfaces between them. The integration of ASCLEPIOS Framework will be reported in the upcoming deliverables D5.2 and D5.3, along with possible modifications made in the interfaces.

In section 3 we provide useful material and information regarding the toolkit that is already used for the integration and deployment of the platform. The planning of the integration is also included, as created through the discussions among consortium partners; two main releases will be provided, first at M21 and the second, final release of ASCLEPIOS at M27.

Finally, the testing and evaluation plan to verify the proper functioning and performance of the integrated ASCLEPIOS platform are presented in section 4, while the testing results (an outcome of Task 5.3), the detailed information about the CI (an outcome of Task 5.2), and the Quality Assurance (an outcome of task T5.4) will be provided in D5.2 and D5.3 that are reporting the outcomes of the work package t.

6 References

1. A., Michalas et al., 2019. D1.2 ASCLEPIOS Reference Architecture, Security and E-health Use Cases, and Acceptance Criteria. ASCLEPIOS Deliverable
2. https://www.sciencedirect.com/topics/computer-science/technical-architecture
3. ASCLEPIOS D7.4
4. R., G., Roessink et al., 2020. D2.2 Attribute-Based Encryption, Dynamic Credentials and Ciphertext Delegation and Integration in Medical Devices. ASCLEPIOS Deliverable
5. A., Michalas et al., 2019. D2.1 Symmetric Searchable Encryption and Integration in Medical Devices. ASCLEPIOS Deliverable
6. Biliri et al., 2020 D2.3 GDPR-compliant and Privacy-Preserving Analytics for Healthcare Providers. ASCLEPIOS Deliverable
7. Y., Verginadis et al., 2019. D3.1 ASCLEPIOS Security and Policies Model. ASCLEPIOS Deliverable
8. Y., Verginadis et al., 2020. D3.2 ASCLEPIOS Models Editor and Interpretation Mechanism. ASCLEPIOS Deliverable
9. P.Gouvas et al., 2020. D3.3 Context-aware ABAC Enforcement Mechanism. ASCLEPIOS Deliverable
10. An Overview of the Testing Process | Preface. https://flylib.com/books/en/2.174.1/an_overview_of_the_testing_process.html
11. ASCLEPIOS SSE GitLab : https://gitlab.com/asclepios-project/ssemanual
12. http://www.technofunc.com/index.php/erp/178-what-is-integration-testing
13. OAuth 2.0, https://oauth.net/2/
14. OpenIDConnect, https://openid.net/connect/

Appendix I - Asclepios Technical Architecture UML Schema

The following code in PlantUML language[21] can be used in any PlantUML server in order to generate a high-resolution figure of the architecture.[22]

```
@startuml
' ---------------------------------------------------
skinparam defaultTextAlignment center
' ---------------------------------------------------
'top to bottom direction
left to right direction
scale 2/3
skinparam linetype polyline
skinparam linetype ortho

node "Trusted Authority" {
interface "HTTP " as tainterface
}

node "ABE" {
component "ABE Server" as abeserver
interface "HTTP" as abeAPI
abeserver - abeAPI
abeserver -> tainterface
}

node "ASCLEPIOS ABAC and ABE Policy Enforcement Mechanisms (AAPEM)" {
  component "Policy Enforcement Engine" as pepengine
  component "OIDC connect server" as caauthority
 interface "HTTP" as pepApi
 interface "HTTP" as caaAPI
 interface "HTTP" as fileshareAPI
caauthority - caaAPI
pepengine - pepApi
pepengine - fileshareAPI

[ABE Connector] --> abeAPI
}

node "ASCLEPIOS Models and PoLicies Editors (AMPLE)" {
database CASM
[Policies Interpreter] --> fileshareAPI
interface "HTTP" as abaceditApi
[ABAC Policies Editor] - abaceditApi

[Policies Interpreter] - abeAPI
interface "HTTP" as casmeditApi
[CASM Editor] - casmeditApi
interface "HTTP" as abeeditApi
[ABE Policies Editor] - abeeditApi
interface "HTTP" as validationApi
[Policies Validation tool] - validationApi
```

[21] https://plantuml.com/

[22] Direct Link to online version of the graph

```
interface "UI " as ampleui
[AMPLE Web Interface] --> validationApi
[AMPLE Web Interface] --> casmeditApi
[AMPLE Web Interface] --> abeeditApi
[AMPLE Web Interface] --> abaceditApi
[AMPLE Web Interface] --> ampleui
[Policies Interpreter] - validationApi
}

node "FE Analytics" {
component "FE Analytics Server" as feserver
interface "HTTP" as feserverapi
feserver - feserverapi
feserver -> tainterface
}

node "SSE" {
component "SSE Server" as sseserver
interface "HTTP" as sseserverapi
sseserver - sseserverapi
sseserver -> tainterface
}

node "Cybersecurity, Encryption and Analytics (CEAA)" {
component "Log Collector" as ceaalogs
interface "Syslog " as ceaalogsapi
ceaalogs - ceaalogsapi
component "Indexing and Query Engine" as ceaaindexer
interface "Syslog " as internallogsapi
ceaaindexer - internallogsapi
ceaalogs -> internallogsapi
[Analysis Engine] -> internallogsapi
interface "HTTP " as analinterface2
ceaaindexer  - analinterface2
[CEEA User Interface] -> analinterface2
[Analysis Engine]
interface "UI " as Interfceaa
Interfceaa -- [CEEA User Interface]
}

node "Privacy Analytics Module (APAM)" {
interface "UI " as Interfapam
Interfapam - [APAM WEb Interface]
Component "Analytics Engine" as apamanalytics
interface "HTTP " as apamanalinterface
apamanalytics - apamanalinterface
[APAM WEb Interface] -> apamanalinterface
interface "Syslog " as apamlogsapi
[Log Collector] - apamlogsapi
apamanalytics  -> analinterface2
}

node "User Space" {
  [SEE Client] --> sseserverapi
 [User application]      --> [SEE Client]
 Interface "Access Control Point" as PEP
 [User application] - PEP
PEP --> pepApi
[FE Client] --> feserverapi
component "ABE Library" as abelibrary
abelibrary --> abeAPI
abelibrary --> tainterface
[FE Client] --> tainterface
[SEE Client] --> tainterface
}

node "Privacy preserving distributed statistical computation (EMNET)" {
[EMNET Worker]
interface "HTTP " as emnetcoordination
[EMNET Coordinator] - emnetcoordination
[EMNET Worker] -> emnetcoordination
[EMNET Worker]  --> feserverapi
[EMNET Worker]  --> abeAPI

}
@enduml
```