D3.1.3

Draft proof of concept for home healthcare

Project number:	257243
Project acronym:	TClouds
Project title:	Trustworthy Clouds - Privacy and Resilience for Internet-scale Critical Infrastructure
Start date of the project:	1st October, 2010
Duration:	36 months
Program:	FP7 IP

Deliverable type:	Prototype
Deliverable reference number:	ICT-257243 / D3.1.3 PU / 1.0
Activity and Work package contributing to the deliverable:	Activity 3 / WP 3.1
Due date:	September 2012 – M24
Actual submission date:	28th September 2012

Responsible organization:	PHI
Editor:	Mina Deng
Dissemination level:	Public
Revision:	1.0

Abstract:	This document describes what has been accomplished during the development of the Home Healthcare Draft Proof of Concept as a Trusted Platform as a Service, on a commodity cloud.
Keywords:	Healthcare Trusted PaaS, cloud based healthcare platform, proof of concept

Editor

Mina Deng (PHI)

Contributors

Mina Deng, Zheyi Rong, Sebastian Banescu, Ya Liu (PHI)

Marco Abitabile, Marco Nalin (FSR)

Ninja Marnau (ULD)

Disclaimer

This work was partially supported by the European Commission through the FP7-ICT program under project TClouds, number 257243.

Executive Summary

This document describes the draft proof of concept for the TClouds home healthcare use case (that is, a Healthcare Trustworthy Platform as a Service, or TPaaS). It describes the design of the architecture and underlying implementation modules, requirements, implementation, and the user manual of the proof of concept, which is built on top of TClouds as a benchmark application and user-centric evaluation.

The implementation of TPaaS involves a service-oriented architecture, including services that guarantee security, manage databases, and provide a user interface. Besides, a Domain Specific Language was implemented to improve modularity and security. The resulting platform has a website to enable a user to access their personal health records, to maintain relationships between him and third-party applications, and to audit the behaviours of the applications that the user has authorized. These TPaaS functionalities fulfil the functional requirements reviewed and analysed before the implementation.

The Healthcare TPaaS offers a set of on-board software security modules, including RESTful APIs, the encryption module, and the secure logging and auditing module. The RESTful APIs implement OAuth2.0 protocol allowing users to share their personal health records stored in the TPaaS without having to disclose their private credentials to any third party applications that are deployed on the TPaaS. Besides OAuth2.0, these APIs implement the OpenID protocol to allow users to log on to third party applications with their TPaaS ID. The data files and communications within the TPaaS are encrypted by the encryption module by using, for instance, CP-ABE cipher. The secure logging and auditing module logs every access from third party applications, ensures the integrity of the log files, and analyses whether activities have been performed as promised. The databases used in the TPaaS comprise the Management Database, which stores all the data necessary to maintain and perform basic social functionalities of the platform, and the Data Database, which contains Personal Health Records of users.

This document also identifies a list of security and resilience requirements for the underlying cloud environments. However, neither Amazon nor OpenStack can fulfil as many security and resilience requirements as TClouds does. With the help of A2's security components, TClouds can:

- replicates data across different replicas and tolerate byzantine or crash faults (BFT-SMART/DivDB)
- leverages many different clouds in order to increase robustness and security of a remote storages system solution (Cloud of clouds storage)
- log and track events generated at multiple cloud layers (Log Service)
- assesses the integrity of physical hosts that are parts of the Cloud infrastructure (Remote Attestation Service).
- separates sensitive operations and secrets (Secure Key Management).

Therefore A3 is happy about the security and privacy requirements that could be supported by TClouds.

The document also provides a user manual on how the proof of concept of the Healthcare TPaaS may be operated. The distribution of the draft proof of concept of the Healthcare TPaaS is not provided in the form of binary files due to the complex installation. Instead, this platform is accessible online (i.e. on the SRX OpenStack cloud). With a graphical user interface, it enables functionalities such as login/sign-in, adding new friends, privacy policy specification, and auditing. In the third project year, there are improvements to be done, such as extending the user interface and the use of DSL.

To improve this platform, in the third project year, a more user-friendly and structured interface may be developed; documentation of APIs in this platform could be more detailed to alleviate new developers work.

Contents

List of tables

List of figures

Chapter 1

Introduction

Chapter Author:

Mina Deng (PHI)

1.1 TClouds — Trustworthy Clouds

TClouds aims to develop trustworthy Internet-scale cloud services, providing computing, network, and storage resources over the Internet. Existing cloud computing services today are generally not trusted for running critical infrastructure, which may range from business-critical tasks of large companies to mission-critical tasks for the society as a whole. The latter includes water, electricity, fuel, and food supply chains. TClouds focuses on power grids, electricity management and patient-centric health-care systems as main applications.

The TClouds project identifies and addresses legal implications and business opportunities of using infrastructure clouds, assesses security, privacy, and resilience aspects of cloud computing and contributes to building a regulatory framework enabling resilient and privacy-enhanced cloud infrastructure.

The main body of work in TClouds defines an architecture and prototype systems for securing infrastructure clouds, by providing security enhancements that can be deployed on top of commodity infrastructure clouds (as a cloud-of-clouds) and by assessing the resilience, privacy, and security extensions of existing clouds.

Furthermore, TClouds provides resilient middleware for adaptive security using a cloud-of-clouds, which is not dependent on any single cloud provider. This feature of the TClouds platform will provide tolerance and adaptability to mitigate security incidents and unstable operating conditions for a range of applications running on a cloud-of-clouds.

1.2 Activity 3 — Benchmark Applications & User-centric Evaluation

Activity 3 focuses on the applications and the evaluation of the TClouds platform. The activity's objective is to define and validate cloud applications' architecture and specifications (the medical and the smart grid use case). For this purpose, Activity 3 will specify cloud and application functionalities and requirements, define application prototypes to be implemented on the cloud platform, and validate the application prototypes and the TClouds platform. For this purpose, the requirements defined in Activity 1 will serve as a generic guideline, which will be refined and consolidated in Activity 3. Finally, there is a continuous and close interaction between Activity 3 and Activity 2 in order to make sure that the platform and applications match the specifications and that the TClouds project achieves its overall objectives.

1.3 Work package 3.1 — Cloud Applications Data Structures for Home Healthcare Benchmark Scenario

WP3.1's main objective is to define the cloud architecture and specification for running TClouds selected application, i.e., provides technical requirements for cloud computing for the medical sector, especially in the area of home healthcare devices and then turn the analysis into an architecture, API, and protocols for client side. This architecture will be closely linked and integrated with WP2.1, WP2.2, and WP2.3. The protocols will be validated in collaboration with the demonstrator implementation.

1.4 Deliverable 3.1.3 — Draft proof of concept for home healthcare

1.4.1 Overview

This document provides explanatory information of the draft proof of concept for home healthcare. The document details the implementation design and architecture, with embedded security components. It also considers security and resilience requirements for the underlying cloud environments. In addition, it highlights the cloud security and resilience components which are provided by Activity 2. Finally the document provides some instructions on how the prototype may be used.

1.4.2 Structure

Chapter 2 introduces design and implementation decisions of the draft proof of concept for home healthcare, which is the Healthcare TPaaS (Trusted Platform as a Service). Discussions on design changes and implementation decisions will be provided, with an overall architecture and underlying implementation modules.

Chapter 3 addresses the Healthcare TPaaS security, privacy, and resilience requirements when considering moving home healthcare platform and applications to a public cloud environment, and what the expectations for adopting TClouds security solutions jointly provided by Activity 2 and Activity 3 are.

Chapter 4 provides a user manual on how the proof of concept of home healthcare may be operated.

1.4.3 Deviation from Work plan

The implementation of this prototype was fully compliant with the defined work plan.

1.4.4 Target Audience

Target audience of this deliverable includes all TClouds partners, especially partners from Activity 2, who wish to understand the healthcare use case requirements, and evaluate and improve TClouds security solutions in the third year. This document is also relevant for Work package 3.3 for validation and evaluation. The deliverable, to be fully understood, requires some background in software engineering and healthcare IT systems.

1.4.5 Relation to Other Deliverables

D3.1.3 shares the requirements “Healthcare TPaaS RASD (Requirements Analysis Specification Document)” (in Appendix A) with WP3.3 (D3.3.3 – Validation Protocol and Schedule for the Smart Power Grid and Home Health Care Use Cases). Moreover, D3.1.3 provides a list of requirements for WP2.4, and analyzed the security and privacy risks of the home healthcare use case in D2.4.2 that can be covered by TClouds security components.

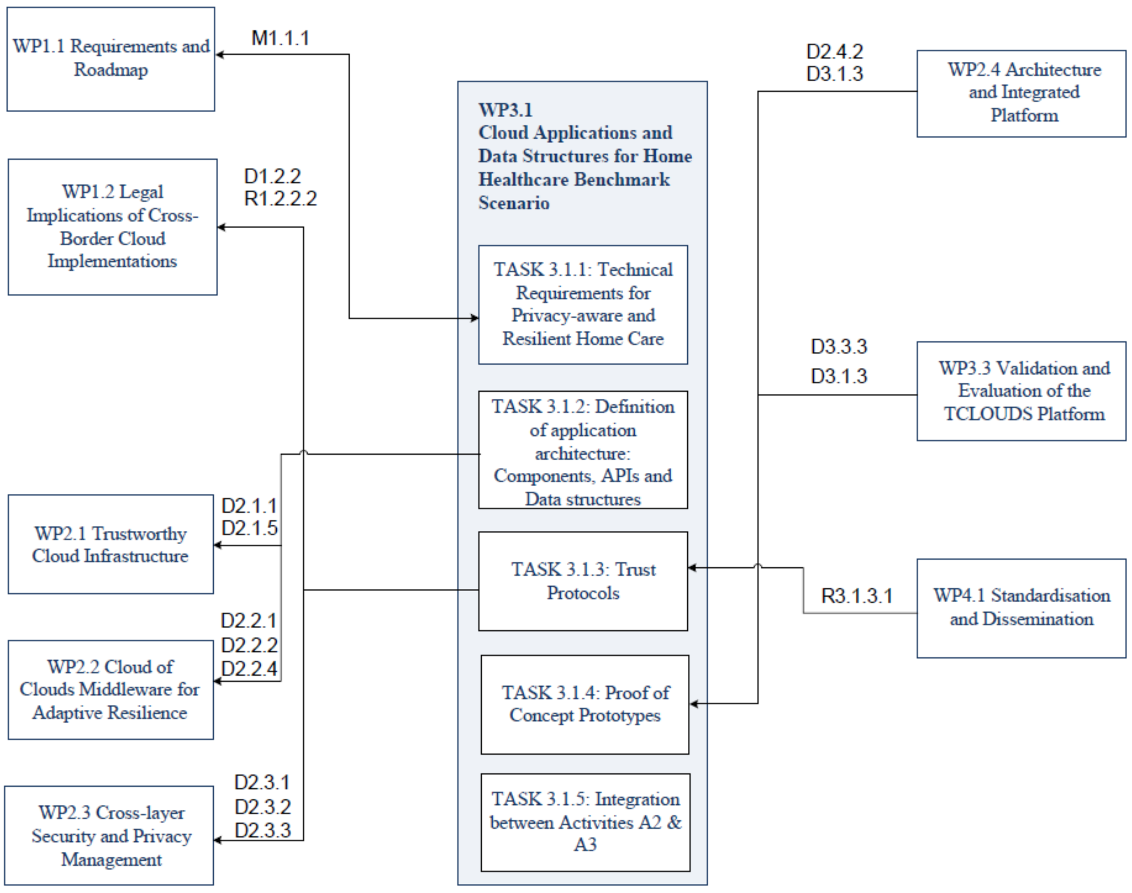

WP1.1 Requirements and Roadmap
M1.1.1
WP1.2 Legal Implications of Cross-Border Cloud Implementations
D1.2.2
R1.2.2.2
WP2.1 Trustworthy Cloud Infrastructure
D2.1.1
D2.1.5
WP2.2 Cloud of Clouds Middleware for Adaptive Resilience
D2.2.1
D2.2.2
D2.2.4
WP2.3 Cross-layer Security and Privacy Management
D2.3.1
D2.3.2
D2.3.3
WP3.1 Cloud Applications and Data Structures for Home Healthcare Benchmark Scenario
TASK 3.1.1: Technical Requirements for Privacy-aware and Resilient Home Care
TASK 3.1.2: Definition of application architecture: Components, APIs and Data structures
TASK 3.1.3: Trust Protocols
TASK 3.1.4: Proof of Concept Prototypes
TASK 3.1.5: Integration between Activities A2 & A3
D2.4.2
D3.1.3
WP2.4 Architecture and Integrated Platform
D3.3.3
D3.1.3
WP3.3 Validation and Evaluation of the TCLOUDS Platform
R3.1.3.1
WP4.1 Standardisation and Dissemination

Figure 1: Interdependency chart for WP3.1

Chapter 2

Home Healthcare proof of concept implementation

Chapter Author:

Marco Abitabile, Marco Nalin (FSR), Mina Deng, Zheyi Rong, Sebastian Banescu, Ya Liu (PHI)

On this chapter we bring up some insights over the implemented Home Healthcare Platform (namely the Healthcare TPaaS).

2.1 Design changes

While designing the Healthcare TPaaS the very first time, we proposed a design idea[1] that has been slightly changed over the time, in order to go for looser coupling, modularization and higher security.

The actual architecture is provided in Figure 2.

The main difference with the previous design is the introduction of a DSL (Fowler, 2010), Domain Specific Language. The main reasons of using a DSL are the following.

- Better reuse of code: since DSL means building a (simplified) language, the “phrases” of this language contains semantics and a proper parsing allow to reduce at minimum code duplication
- Higher modularity: DSL allows a sharp distinction among the different layers of the platform. The “core” of the elaboration stays into the DSL engine and whatever changes outside (new technology, new interfaces, new databases) will not interfere with the high level execution of an atomic action
- Security enhancement: it is well known that security breaches, especially the most critical, happen as a result of insiders. Building a modular system whose core works with a specific language means increase the overall security since internal developers can code and implement new functionalities only by writing well-formed sentences that the DSL engine can accept. Otherwise sentences are rejected by the DSL engine. Critical code remains closed only into the DSL development team and consequently is reduced the probability of presence of malicious code.

[1] See: Figure 6 on D3.1.2 (page 31) at https://svn.tclouds-project.eu/trunk/ActivityA3-EvaluationAndTrials/Documents/D-EU-Deliverables/TC-D3.1.2/TC-D3.1.2-PU-Application-API-1st-specification-on-application-side-trust-protocols_M18.pdf

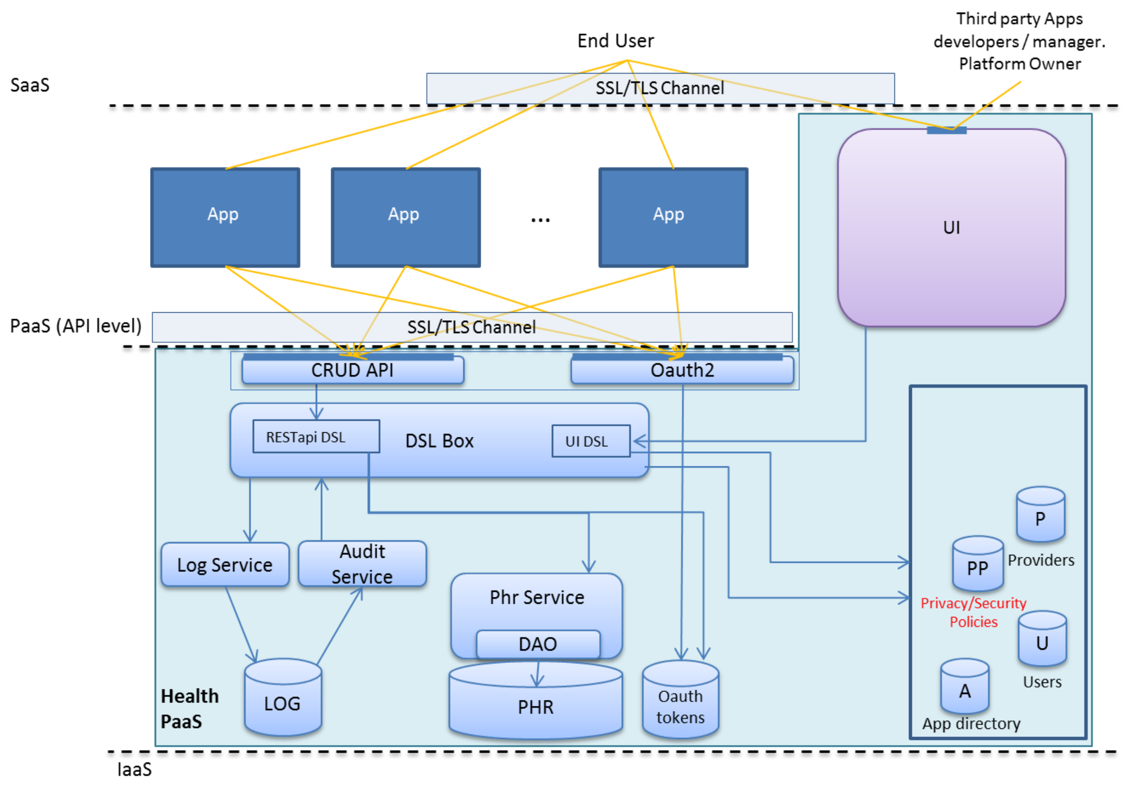


Figure 2 : Healthcare TPaaS architecture

2.2 Requirements Analysis Specification

This chapter describes the Healthcare TPaaS platform and its main functionalities. In addition, this chapter analyzes the requirements for A2 components. Please refer to:

- D3.1.2[2] to have a deeper understanding on the main characteristics of the Healthcare TPaaS and
- D2.4.2[3] to have a better understanding on the A2 components that A3 is willing to use in order to satisfy its requirements.

2.2.1 The user business or background of the project effort

The Healthcare TPaaS aims to be a platform that runs in the cloud in a safe and trustworthy fashion. Allowing third party applications to use it and benefit from the service it provides.

The idea is to build a trusted platform to manage health records on top of TClouds components (to fulfill the request we have as partner in the TClouds consortium).

[2] https://svn.tclouds-project.eu/trunk/ActivityA3-EvaluationAndTrials/Documents/D-EU-Deliverables/TC-D3.1.2/TC-D3.1.2-PU-Application-API-1st-specification-on-application-side-trust-protocols_M18.pdf

[3] https://svn.tclouds-project.eu/trunk/ActivityA2-TrustedCloudPlatform/Documents/D-EU-Deliverables/TC-D2.4.2/TC-D2_4_2.pdf

In this document will be described the requirements that Healthcare TPaaS needs to fulfill. At this stage (Alpha release) we will be describing TClouds requirements and, in addition, other extra requirements that will allow us to support a Beta release.

Figure 3 depicts the business model that underlies the Healthcare TPaaS idea in which actors are introduced. The actors will be described in detail in next chapters.

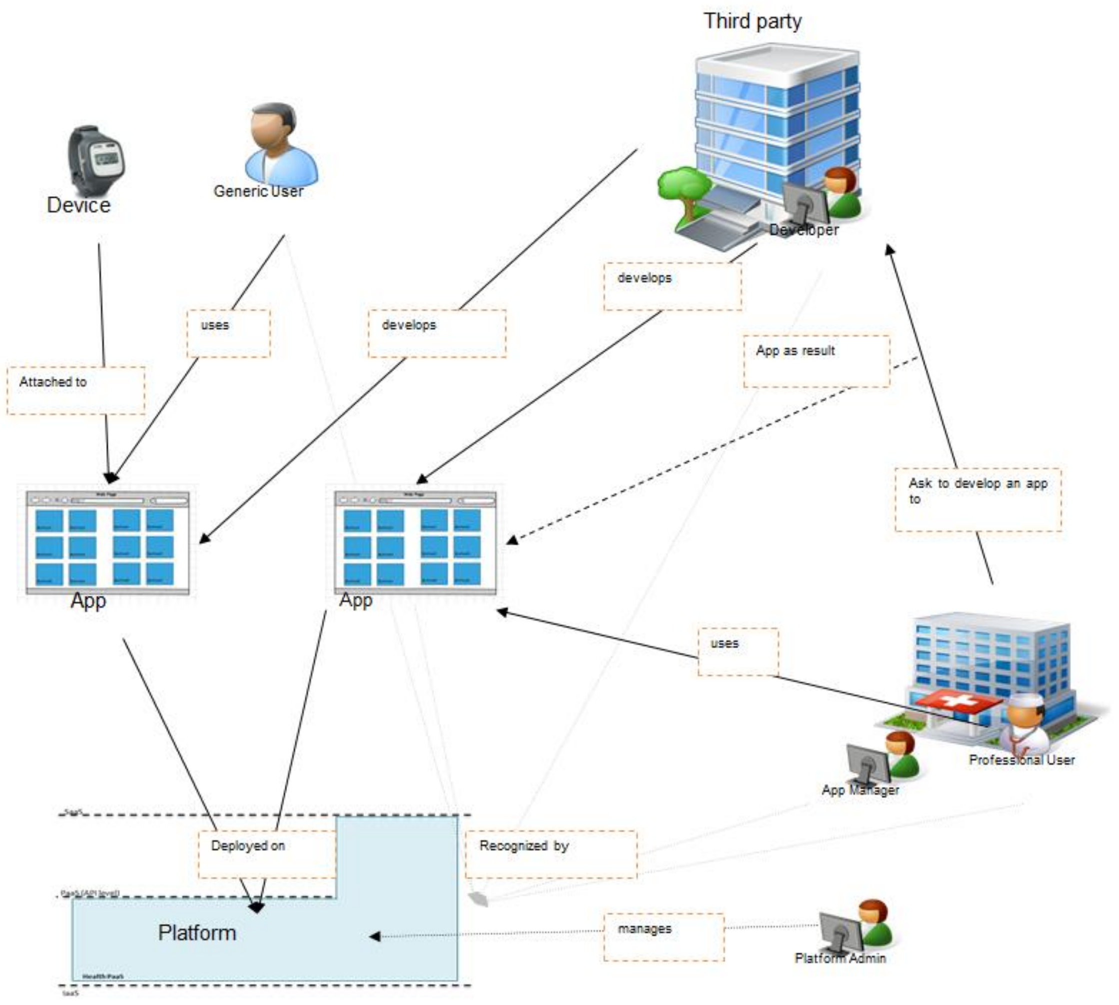


Figure 3 : High Level description of the Healthcare TPaaS and relation with users

2.2.2 Description of the Healthcare TPaaS

This section describes in detail the functionalities and services offered at the Platform as a Service (PaaS) layer. In this we try to outline what we call it “Healthcare Trusted PaaS”, or simply “Healthcare TPaaS”. Indeed, some of the services actually have an interface toward end users, and we consider them as Software as a Service (SaaS). The Healthcare TPaaS is a multilevel platform that aims to provide novel services such as:

- Store trustworthily health-related data (relying on a trusted IaaS);
- Provide API for 3rd party apps to access to users’ health data, in a privacy-preserving manner (PaaS);
- Provide API for 3rd party apps to use identity/role management services (PaaS);
- Provide an interface to allow 3rd party app developers to register their application(s) in order to access the users’ data (SaaS);
- Allow End Users to manage their data and specify privacy policy about which data a particular application/ user can access (SaaS).

A first draft architecture of this platform, as shown in Figure 2, has been produced in the TClouds project, and it is shown in Figure 3. Even if the picture is very high level, it shows the boundaries of the Healthcare TPaaS, with its interfaces both toward users (being actual end users of applications or developers of third parties applications) and toward applications using this platform, as well as toward the lower infrastructure layers, offered by a Trustworthy IaaS cloud.

2.2.3 Partner or collaborative applications

The platform will work in conjunction with the infrastructure features that A2 group are developing. Once the platform is completed it will interact with the application provided by third party developers. In the following we enumerate the A2 components and highlight their value to A3 Healthcare TPaaS. Please note that the name/feature of A2 security components below may involve as the project proceeds.

Req:	Log service
Description:	The Log Service guarantees that concerned events from the infrastructure layer should be logged. In addition, the Log Service shall ensure integrity, privacy, access control and availability of log entries.

Req:	Tailored Cloud Services
Description:	The memcached[4] is a simple network service to save and retrieve any kind of binary data in ephemeral RAM of computing nodes. It is typically used to save smaller bits of information that would otherwise require computational effort to regenerate on each request, for example rendered HTML documents from a content management system.
Rationale:	The improved memcached could possibly be used to improve performance for frequently accessed, dynamically updated web pages in the end-user web-frontend. The memcached can cache results of arbitrary functions.

Req:	R-BPEL
Description:	RBPEL[5] is a platform for the fault-tolerant execution of Web-service-based workflows particularly within clouds. All work necessary for the active replication is done totally transparent to the process definitions of the composite Web services. This means, existing process definitions can be reused without manual intervention and new definitions can be written as intended for standard, non-replicated BPEL platforms BPEL's main purpose is the definition of Web services that realize their functionality on basis of other Web services. Thus, it mainly offers means to receive and handle standard Web service requests from clients, to invoke other Web services and process the results of such invocations and to send replies to the calling clients. Furthermore, BPEL has many aspects from general purpose programming languages. It has control flow expressions, e.g. for branches and loops, variables, exception handling etc. Besides that, BPEL allows easy access to and transformation of SOAP messages, the protocol usually used for the communication with Web services.
Rationale:	RPBEL can be used whenever highly available composite Web services are needed. If there are multiple Web services that need to be coordinated, orchestrated or combined in whatever way, the required code required to put all these Web services together could be written in BPEL. The benefit would be that RBPEL can be used in order to make the whole system highly available and tolerant to crashes of subsystems. In the medical use case, one possible location for the integration of RBPEL could be the A3 middleware, maybe within the Health Management Application.

[4] Chapter 10 Tailored Cloud Services, TClouds deliverable D2.1.1 Technical Requirements and Architecture for Privacy enhanced and Resilient Trusted Clouds

[5] Chapter 8.2 Fault-tolerant Workflow Execution (FT-BPEL), TClouds deliverable D2.2.4 TClouds Prototype Architecture, Quality Assurance Guidelines

Req:	Secure Block storage
Description:	The SBS is able to provide security properties including confidentiality, integrity and authenticity, version control and replay attack prevention. Here we assume a secure and attestable hypervisor. Otherwise it must be blindly trusted. • For confidentiality, the data stored on block devices inside the VM shall be transparently encrypted by the hypervisor so that the stored data at least cannot be eavesdropped. Using encryption the stored data, mounted as a file system inside the VM, is only accessible in plaintext by those authorized to have access. • For Integrity and Authenticity, the data stored on block devices inside the VM shall be transparently integrity-protected by the hypervisor so that tampering with the stored data can be detected. This is achieved with digital signatures (or Message Authentication Codes, MACs) such that the stored data can be checked for authenticity and tampering. • SBS also provides Version Control (or Replay Attacks Prevention). Even though an adversary cannot read encrypted data, it is possible for her to replay previously saved encrypted data. Possible adversaries are: Local/remote administrators of the cloud provider. This is achieved with hardware/virtual counters (e.g. provided by the TPM), which is possible to enable version control for encrypted data chunks.
Rationale:	From the infrastructure layer, the SBS scheme requires trusted platform as required an external component. SBS will rely on a trusted platform (hardware) with a hardware root of trust. The platform shall provide a hardware Trusted Platform Module (TPM). As SBS is transparent to the application or platform layer, it will not influence Activity3. However, a slightly modified interaction with the actors from A3 is necessary, because they have to additionally provide a cryptographic key.

Req:	Trusted Virtual Domains
Description:	Resilient Object Storage[6] builds reliable and secure storage through a federation of object storage services from multiple providers. Multiple clients may concurrently access the same remote storage provider and operate on the same objects.

[6] Chapter 8.3 Resilient Object Storage, TClouds deliverable D2.4.1 TClouds Prototype Architecture, Quality Assurance Guidelines, Test Methodology and Draft API

	This is done through an interface that contains common operations of object cloud storage. The software is libraries run by each client before it accesses cloud storage; the management and setup is the same as for accessing one storage provider, and the library does not require client-to-client communication. The library requires some cryptographic credentials (public keys) of all clients to be present.
Rationale:	For year 2 there are no requirements to WP3.1 as we will deploy the complete application (possibly with several VMs) within the same TVD. Benefits from running the healthcare platform on a Trusted Infrastructure (TVD) are separation from other customers on the cloud, and encryption of data and of the communication between our virtual machines. The vision for year 3 can become more ambitious and separate the different virtual machines of healthcare platform and applications to different TVDs (e.g. one for each stakeholder) and define the allowed information flows between the TVDs. In this scenario, to allow communication between TVDs according to the information flow policy, the VMs cannot directly communicate but are intercepted by an information flow manager. A2 will provide a generic information flow manager framework that has to be instantiated to work with the communication channels of the healthcare TVDs. In case of using REST protocol within the healthcare platform and applications, this means that A2 and A3 jointly have to develop proxies that intercept the communication at the boundaries of a TVD, such that the TVD manager can take control and govern the communication

Req:	Resilient Object Storage
Description:	Resilient Object Storage[7] builds reliable and secure storage through a federation of object storage services from multiple providers. Multiple clients may concurrently access the same remote storage provider and operate on the same objects. This is done through an interface that contains common operations of object cloud storage. The software is a library linked by each client before it accesses cloud storage; the management and setup is the same as for accessing one storage provider, and the library does not require client-to-client communication. The library requires some cryptographic credentials (public keys) of all clients to be present.
Rationale:	The system can be used to store medical data that is critical in terms of availability, integrity and confidentiality. Moreover, this data can be shared by multiple (trusted) parties (such as PHI and FSR) using the untrustworthy clouds as coordination media.

[7] Chapter 8.3 Resilient Object Storage, TClouds deliverable D2.4.1 TClouds Prototype Architecture, Quality Assurance Guidelines, Test Methodology and Draft API

Req:	Access Control as a Service
Description:	ACaaS[8] provides a component that considers user requirements during normal operations as well as in incidents. Example of user requirements includes the following: enforce location restrictions, manage the hosting of dependent applications (e.g. group a set of applications to be hosted within physical proximity and, simultaneously ensure they do not run on the same Computing Node), and exclude certain physical properties from hosting a user application. The objective of ACaaS is to act as a policy decision point to manage the hosting of VM instances at an appropriate Computing Node. ACaaS component verifies that a Computing Node satisfies User requirements when hosting its VM instance. User requirements include technical properties, QoS/SLA requirements (e.g. system availability, reliability measures, and lower/upper resource limits), and security and privacy requirements (e.g. location of data distribution and processing).
Rationale:	Managing clouds' hosting environments based on real user requirements is one of the important A3 security and privacy requirements. WP3.1 need to supply the security and privacy requirements when instantiating the VM, or when there is a need to update the above requirements. In Year 2 the component takes two forms of input: Infrastructure properties and policies (will be provided manually at this stage). User properties defining application components dependencies, e.g. two VMs should be hosted in physical proximity but must not use the same physical platform; the VMs should not be hosted on certain VMs, etc.

Req:	Security Assurance for Virtual Environment
Description:	SAVE (Security Assurance for Virtual Environment)[9] is a tool for automated validation of isolation of cloud users. It extracts configuration data from multiple virtualization environments, transforming the data into a normalized graph representation, and subsequent analysis of its security properties. An information flow analysis on the virtualized infrastructure topology will be employed that forms the basis for isolation breach diagnosis.

[8] Chapter 9.1 Access Control as a Service (ACaaS), TClouds deliverable D2.4.1 TClouds Prototype Architecture, Quality Assurance Guidelines, Test Methodology and Draft API

[9] Chapter 9.5 Automated Validation of Isolation of Cloud Users, TClouds deliverable D2.4.1 TClouds Prototype Architecture, Quality Assurance Guidelines, Test Methodology and Draft API

Rationale:	The current technology will be integrated and adapted based on OpenStack. WP3.1 could use this component to validate isolation of cloud tenants.

2.2.4 List of requirements

In this subsection we provide a list of requirements that Healthcare TPaaS should address once on completion. Further clarifications about the requirements description and their relationships are provided in Appendix B.

Table 1 : List of requirements

Requirement #	Name
1	User creation
1.1	generic end user creation
1.1.1	Generic end user creation by invite
1.1.2	Generic end user creation by invite
1.2	professional end user creation
1.2.1	Professional end user creation by invite
1.3	creation of developer account
1.3.1	App developer self-creation
1.4	app manager creation
1.4.1	App manager creation by invite
1.5	Platform owner creation with TPaaS installation setup
2	Terms of Use
4	professional end user, app subscription
5	User Login
5.1	Generic end user login
5.2	Professional end user login
5.3	Developer login
5.4	App manager login
5.5	Owner login
6	Load balancing and performances
6.1	Application server mgmt.
6.2	DB server mgmt.
7	User Content Sharing
7.1	Manage someone else’s data
7.8	relationship
7.8.1	Update relationship
7.8.2	Create relationship
7.8.2.1	Creation of a generic end user

Requirement #	Name
7.8.2.2	Creation of a professional end user
7.8.2.3	Creation of an application
7.8.2.4	Temporary relationship
7.8.3	deletion of a relationship
9	User content management
10	CRUD API
11	CRUD operation on data
11.3	Creation
11.4	Update
12	data update Strategry
13	data deletion Strategy
14	Consistency checks
14.2.3	Data integrity in transit
15	Consistency rules
16	CRUD operation on app
16.1	dev_CRUD_on_app: CREATE
16.2	dev_CRUD_on_app: DELETE
16.3	dev_CRUD_on_app: READ
16.4	dev_CRUD_on_app: UPDATE
17	Minimum privacy policy requirements (for app)
18	Privacy policy specification
19	User Notification
20	Security APIs
20.1.1	Access token verification
20.1.2	Consumer Authentication
20.1.4	Consumer registration
20.1.5	Access token issuing
20.1.6	Consumer un-registration/deletion
20.1.7	Access token refresh
20.1.8	Access token revocation
20.2.1	Owner Audits
20.2.2	App manager audit
20.2.3	Generic End- user Audits
20.3.1	Relationship & privacy policy logging
20.3.1.1	Save relationship & privacy log
20.3.2	Data Logging
20.3.2.1	Save Data Log
20.3.3	App Logging

Requirement #	Name
20.3.3.1	Save App Log
20.3.4	Access Logging
20.3.4.1	Save Access Log
21	Identity Protection (Anonymous Credential)
30	Data encryption
30.1	Data at rest encryption for private use
30.2	Data at rest encryption for data sharing
30.3	Data confidentiality in transit
31	Password management
31.1	Password format
31.2	Password renewal/expiration
32	Scalability & extensibility
33	Data ontology definition
34	Request for Data ontology extension
35	Key management
35.1	Key generation
35.2	Key store
40	Restrictions due to privacy policy
100	Perceived security is high
	Access Control as a Service
	Log service
	R-BPEL
	Resilient Object Storage
	Security Assurance for Virtual Environment
	Secure Block storage
	Tailored Cloud Services
	Trusted Virtual Domains

2.3 Architecture

TPaaS architecture is built with the idea of creating a modular platform based on SOA (Service Oriented Architecture). The DSL box is packaged as an API under the shape of a library and can be used by any other component that resides in the same Java Virtual Machine (JVM).

The main services that can be found in Healthcare TPaaS are:

- Log service
- Audit service
- Management DB service
- UI service

DSL component, RESTful API component and Personal Health Record management component are built in the same JVM for sake of security.

2.4 Modules

In this chapter we briefly describe the modules that support the healthcare implementation and how they interact between themselves.

Health TPaaS follows the best practices in terms of software development. Extreme Programming (XP) techniques have been adopted to allow fast prototyping, allowing an easy evolution of the project into bigger dimensions. In addition, modularity has been taken into account in order to allow different development teams to interact among them and build the platform in parallel.

Modules are loosely coupled, and they communicate with each other with specific API that either in the form of libraries or web services.

In the remaining part of this subsection we identify 7 main modules as part of Health TPaaS which are loosely coupled and make the platform easy to maintain.

2.4.1 REST API (Security Component)

REST API is identified by OAuth2.0 and REST API box, as illustrated in Figure 2. REST API is in charge of receiving requests by third party apps and then processes the request. OAuth2.0, on the other hand, is in charge to release/refresh/destroy OAuth tokens and manages the OAuth handshake protocol. The REST interface for TPaaS is described in D3.1.2[10].

2.4.1.1 OAuth2.0

Integrated 3rd party applications access the platform data using secure APIs. To secure accessing the APIs, we use Open Authorization (OAuth2.0) to establish a relationship between a user and an application. In this, the user is able to authorize certain applications to access his/her data through API provided by the platform.

Building on OAuth1.0, OAuth2.0 was introduced to simplify the redirection between application and OAuth provider, especially when the application is a medical device or desktop client with limited Internet connection and I/O methods. OAuth2.0 has generally three roles: data owner, client application as OAuth2.0 consumer and OAuth2.0 provider. OAuth2.0 provider normally consists of an authorization endpoint and resources endpoint.

- **Resource owner is an Owner** user who possesses the resource and stores it in the resource server.

- **Application**

[10] https://svn.tclouds-project.eu/trunk/ActivityA3-EvaluationAndTrials/Documents/D-EU-Deliverables/TC-D3.1.2/TC-D3.1.2-PU-Application-API-1st-specification-on-application-side-trust-protocols_M18.pdf

- **Web application** is running on a remote web server. The secret of a client application is confidential which is assigned during the client application registration on the authorization server.
- **Native application** is, for instance, a desktop application or a mobile application with limited web access. The application secret and ID should be stored at user side (locally) in which case the secret is not confidential to the user any more.

- **Resource server** is responsible to store and operate a resource for a resource owner and provide an API for data access.
- **Authorization server** is responsible to generate an access token for the client application to access a resource from a resource server, which should be approved by a resource owner.

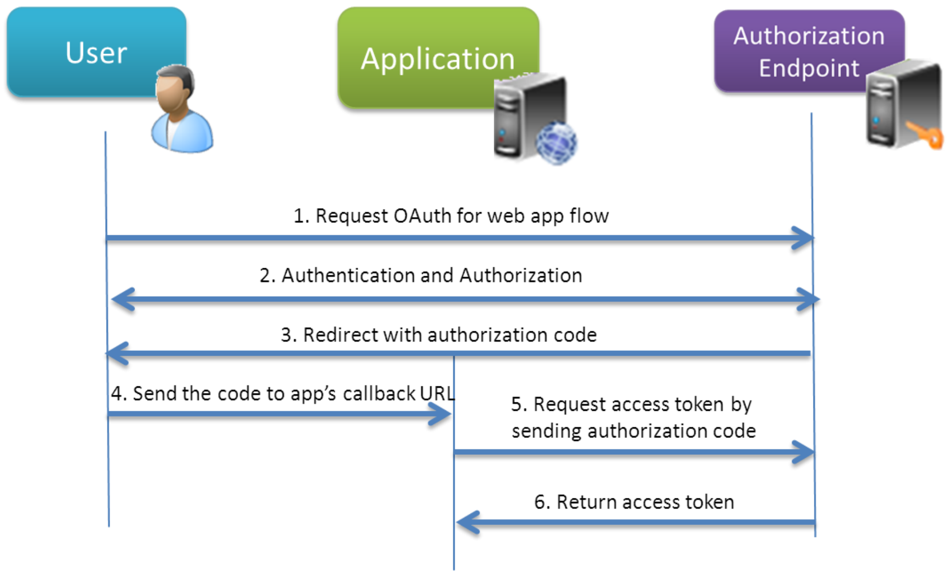

User
Application
Authorization Endpoint
1. Request OAuth for web app flow
2. Authentication and Authorization
3. Redirect with authorization code
4. Send the code to app's callback URL
5. Request access token by sending authorization code
6. Return access token

Figure 4 : Data flow of OAuth 2.0 for web application

In the web application scenario, the user first accesses the web application by the application side authentication. He/she then can send an OAuth request to authorization endpoint by providing the application ID and a token type which is “coded" in the web application case. Meanwhile the user will be redirected to the log-in page on the OAuth provider side. Once the authorization endpoint receives the request and the user is authenticated by the provider, the user can either approve or deny the access requirement from the client application. If it is approved, an authorization code is generated and sent back to the application callback URL. The application is able to exchange the authorization code with a valid access token by providing its credentials obtained during the application registration. In the native application scenario, the application can obtain the access token directly after the access is authorized by the end user.

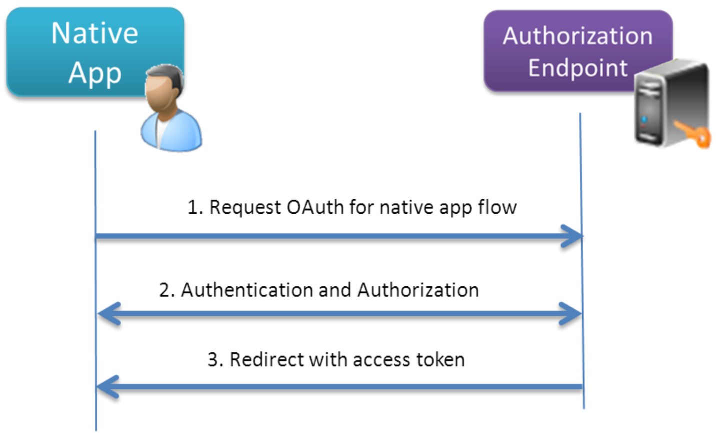

Native App
Authorization Endpoint
1. Request OAuth for native app flow
2. Authentication and Authorization
3. Redirect with access token

Figure 5 : Data flow of OAuth 2.0 for native application

Those two modes provided by OAuth 2.0 enable the interaction between client application and service provider with minimum number of redirections. For the client side implementation, the only difference between those two modes is the request parameters.

2.4.1.2 ID provider service based on OAuth

Besides protecting the web services, OAuth can be applied for user authentication, which is similar to OpenID protocol. Our platform, as a trustworthy identity provider, provides the basic user account information to 3rd party application. One scenario would be "a user can use his TPaaS identity to be authenticated by a 3rd party application". The basic data flow of ID provider service built above OAuth 2.0 is listed in Figure 6.

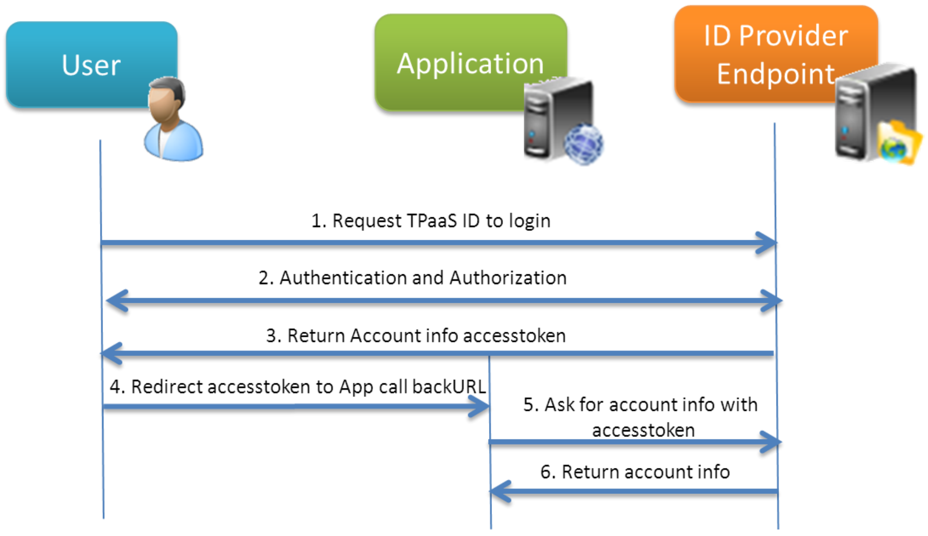

User
Application
ID Provider Endpoint
1. Request TPaaS ID to login
2. Authentication and Authorization
3. Return Account info accesstoken
4. Redirect accesstoken to App call backURL
5. Ask for account info with accesstoken
6. Return account info

Figure 6 : TPaaS ID provider data flow

2.4.1.3 OAuth2.0 service provider development

The implementation of OAuth 2.0 service provider is built on Spring MVC framework by providing RESTful web services.

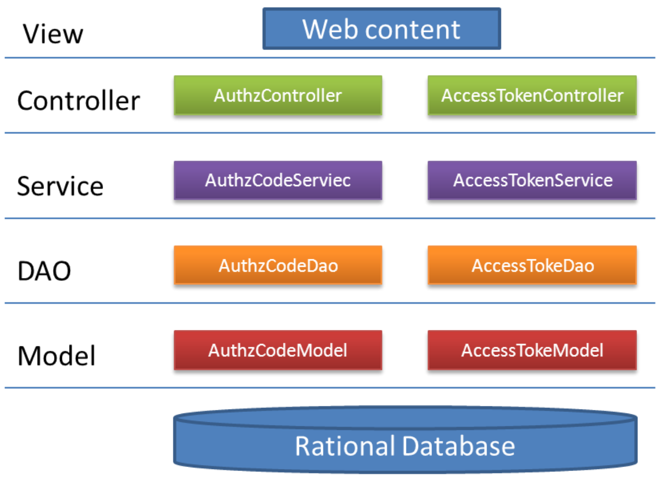


Figure 7 : MVC diagram of OAuth2.0 service provider

There are two major models in OAuth2.0. The first is for authorization code and the second for access token. Data access object (DAO) is the object that provides data query for the model, such as insert, update and delete. Above the DAO layer, we have a service layer offering specific functionalities to manage authorization codes and access tokens. Controllers here are the handlers for service requests by applying the functions from the service layer. At the end, the OAuth2.0 service is represented by web pages at the view layer.

2.4.2 CP-ABE (Security Component)

Ciphertext policy attribute-based encryption (CP-ABE) is another mechanism to preserve data confidentiality against unauthorized accesses.

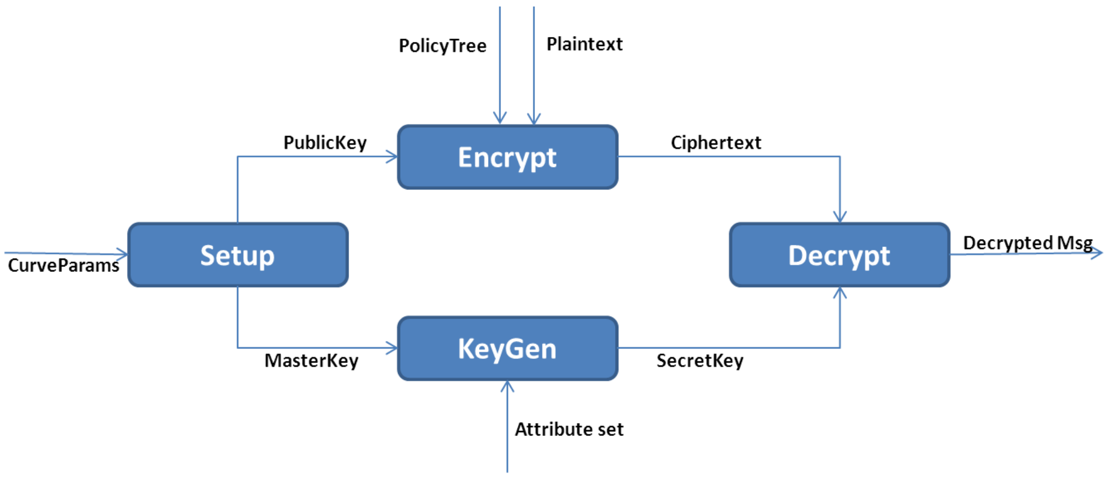


Figure 8 : CP-ABE data flow

The CP-ABE implementation consists of four modules: setup, encryption, keygen and decryption, which are implemented by CPABEEngine, CPABEKeyPairGenerator and CPABESecretKeyGenerator. CPABEEngine is responsible of preparing the initial environment for encryption and decryption, while CPABEEngine and CPABEKeyPairGenerator manage the handle setup and the secret key generation.

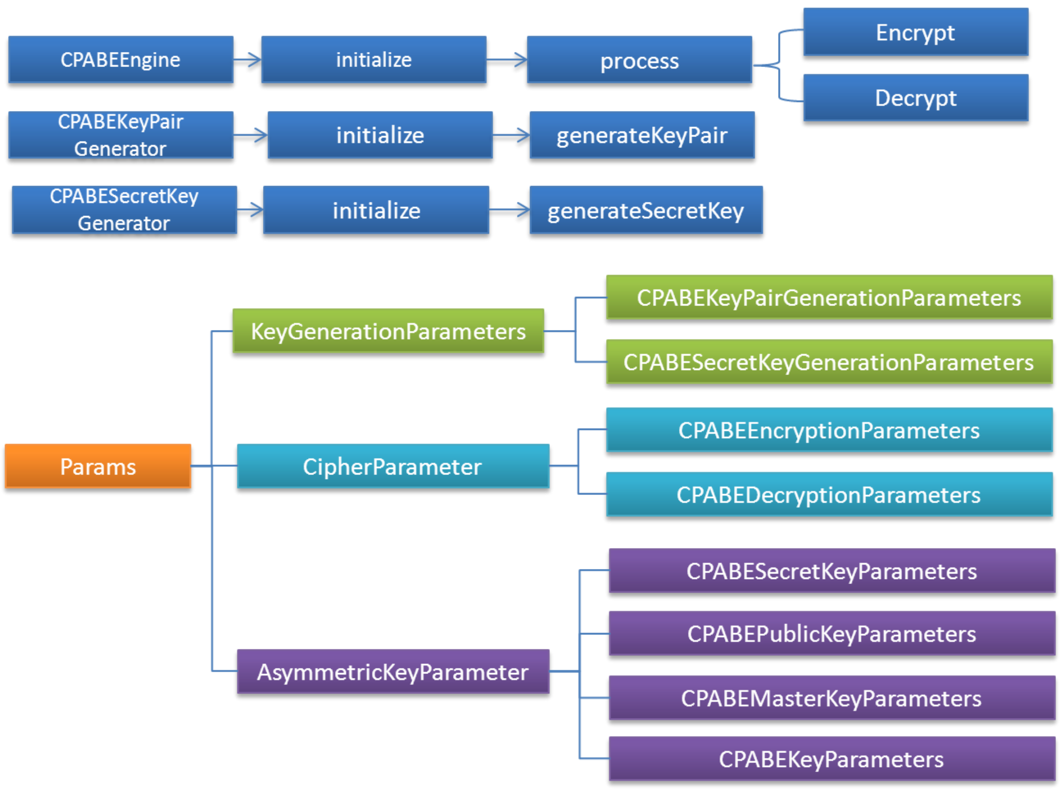


Figure 9 : Function modules and parameter models

2.4.3 Secure logging and auditing (Security Component)

The problem of certifying adherence to privacy regulations is an increasingly important problem for critical systems. IT security and privacy audits are currently the only attested way in which adherence to regulations can be certified. However, such an audit is not possible without the support of a logging system that can reliably record and safely store all relevant events that occur in the system.

2.4.3.1 Secure Logging

When a user or an attacker performs a malicious activity, his actions are recorded by the logging system. As a consequence, attackers usually tamper with the recorded logs to erase any trace of the attack. To overcome such undetectable log tampering actions, Schneier and Kelsey have proposed a novel logging scheme that is able to guarantee log integrity, privacy and access control of log entries (Schneier & Kelsey, 1999).

This secure logging scheme was improved by Ma and Tsudik (Ma & Tsudik, 2009), who also presented a truncation and delayed deletion attacks against the Schneier and Kelsey logging scheme.

A logging scheme based on the previous two logging schemes was tailored and implemented by Smiraglia (POL) as a component part of the TClouds project. The implementation consists of the “**libsklog**” library written in the C programming language (Smiraglia).

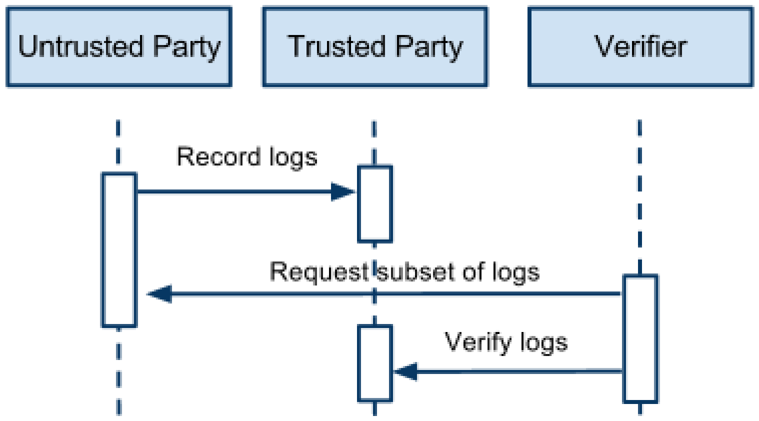


Figure 10 : Interaction between parties in secure logging scheme

This logging library can record general purpose events. However, the development at POL in Y1 was mainly aimed at logging activities at infrastructure level (e.g. start/stop VM, migrate VM, etc.). In Y2, POL has created a "**Log Service**" (SaaS) on top of libsklog, with a RESTful API, to offer remote logging services for A3 applications. However, this Log Service only plays the role of Trusted Party (T) in the secure logging scheme. In order to be able to use the Log Service, A3 applications (e.g. Healthcare TPaaS) have to play the role of Untrusted Party (U) and/or Verified Party (V). In the following section we describe how the integration of the libsklog library was done in the Healthcare TPaaS developed by PHI and FSR.

2.4.3.2 Integration of Log Service (A2) with Healthcare TPaaS (A3)

To take advantage of the security properties offered by the Log Service, PHI has implemented a logging module as part of the Healthcare TPaaS. The logging module plays the role of the untrusted party (U) in the secure logging scheme. Figure 11 shows a high-level architecture of the logging and auditing modules built into the Healthcare TPaaS (right-side) and the Log Service (left-side). Both systems are based on the libsklog C library and communicate with each other via a RESTful interface. Note that the two systems (Log Service and Healthcare TPaaS), should run on separate VMs that both have the libsklog library installed and properly configured.

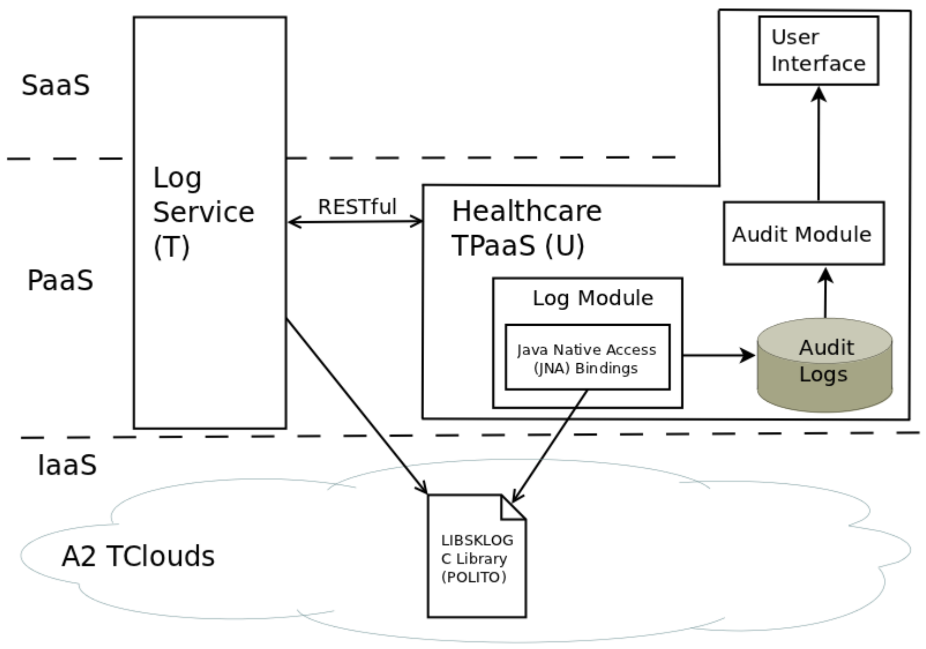


Figure 11 : Log & audit module architecture in the Healthcare TPaaS

2.4.3.3 Healthcare TPaaS Implementation

Due to the myriad of advantages of the secure logging library offered by POL, it was decided that libsklog had to be integrated into the Healthcare TPaaS. Since the Healthcare TPaaS was written in Java, the functions of the libsklog C library provided by POL cannot be directly called. Therefore Java Native Access bindings were implemented by PHI in order to be able to use the libsklog library functions.

2.4.3.4 Log Module

The Healthcare TPaaS initiates a secure logging session with the Log Service through the Log Module by using TCP/IP communication. The Log Module offers developers an object oriented Java interface to the SKLOG_U class of the libsklog C library, shown in the UML class diagram from Figure 12.

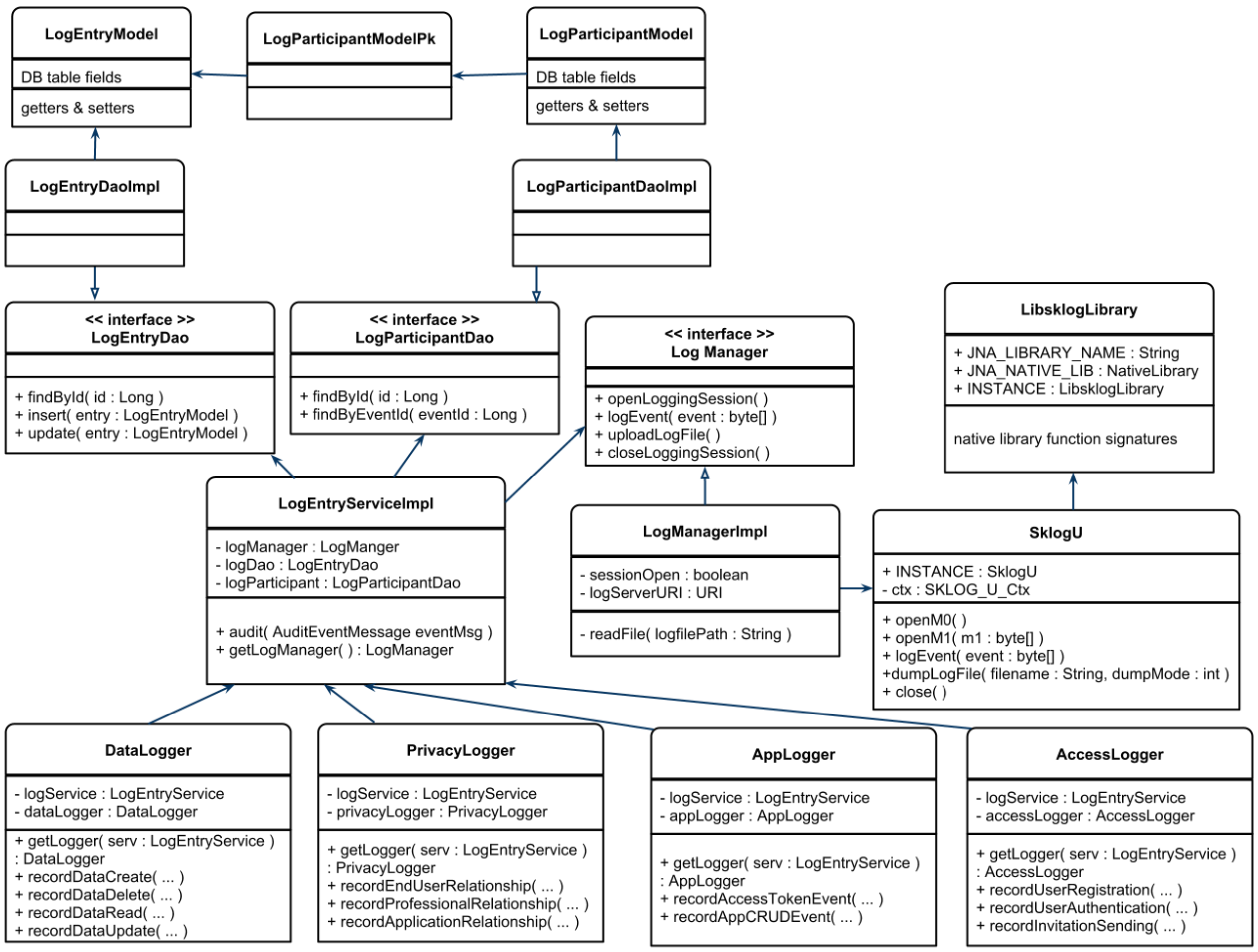

LogEntryModel
DB table fields
getters & setters
LogParticipantModelPk
LogParticipantModel
DB table fields
getters & setters
LogEntryDaoImpl
LogParticipantDaoImpl
<< interface >> LogEntryDao
+ findById(id : Long)
+ insert(entry : LogEntryModel)
+ update(entry : LogEntryModel)
<< interface >> LogParticipantDao
+ findById(id : Long)
+ findByEventId(eventId : Long)
<< interface >> Log Manager
+ openLoggingSession( )
+ logEvent(event : byte[])
+ uploadLogFile( )
+ closeLoggingSession( )
LibsklogLibrary
+ JNA_LIBRARY_NAME : String
+ JNA_NATIVE_LIB : NativeLibrary
+ INSTANCE : LibsklogLibrary
native library function signatures
LogEntryServiceImpl
- logManager : LogManger
- logDao : LogEntryDao
- logParticipant : LogParticipantDao
+ audit(AuditEventMessage eventMsg)
+ getLogManager( ) : LogManager
LogManagerImpl
- sessionOpen : boolean
- logServerURI : URI
- readFile(logfilePath : String)
SklogU
+ INSTANCE : SklogU
- ctx : SKLOG_U_Ctx
+ openM0( )
+ openM1(m1 : byte[])
+ logEvent(event : byte[])
+dumpLogFile(filename : String, dumpMode : int)
+ close( )
DataLogger
- logService : LogEntryService
- dataLogger : DataLogger
+ getLogger(serv : LogEntryService) : DataLogger
+ recordDataCreate(...)
+ recordDataDelete(...)
+ recordDataRead(...)
+ recordDataUpdate(...)
PrivacyLogger
- logService : LogEntryService
- privacyLogger : PrivacyLogger
+ getLogger(serv : LogEntryService) : PrivacyLogger
+ recordEndUserRelationship(...)
+ recordProfessionalRelationship(...)
+ recordApplicationRelationship(...)
AppLogger
- logService : LogEntryService
- appLogger : AppLogger
+ getLogger(serv : LogEntryService) : AppLogger
+ recordAccessTokenEvent(...)
+ recordAppCRUDEvent(...)
AccessLogger
- logService : LogEntryService
- accessLogger : AccessLogger
+ getLogger(serv : LogEntryService) : AccessLogger
+ recordUserRegistration(...)
+ recordUserAuthentication(...)
+ recordInvitationSending(...)

Figure 12 : Class diagram for Log Module

Figure 12 contains classes implementing the model and controller components of the Model-View-Controller software design pattern. The view for the Log Module is part of the Audit Module presented in the next section.

- At the top to the figure the classes LogEntryModel and LogParticipantModel represent database table entities. The LogParticipantPk class represents the composite primary key of the LogParticipantModel. These classes offer an object oriented interface to the database table log_events, respectively log_event_participants, whose structures are shown in Figure 13. The structure of log entries was inspired by the "Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications" described in RFC3881 (6), which is also used by the "Integration the Healthcare Enterprise" (IHE) initiative (7) in their "Audit Trail and Node Authentication" (ATNA) integration profile (8). PHI has implemented a code-value system similar to the one used in ATNA to encode the details of actions. This implementation is not shown in Figure 12, but it can be found in the Healthcare TPaaS source code[11].
- The model classes are used by the Data Access Object (DAO) classes LogEntryDaoImpl and LogParticipantDaoImpl. These two classes implement the

[11] com.tclouds.tpaas.components.security.audit.{codes,events}

LogEntryDao, respectively LogParticipantDao interfaces. These classes are used to perform search, insert and update operations on the database tables mentioned previously. Note that the delete operation is not implemented because log entries shouldn't be deleted.

- The LibsklogLibrary interface contains the JNA bindings for the libsklog C library and several constant fields that specify configuration options.
- The SklogU class is an abstraction of the U party in the Schneier-Kelsey logging scheme that is built using the libsklog library. This class is built according to the Singleton software design pattern.
- The LogManager interface and its implementing class, LogManagerImpl, use the object oriented interface of SklogU to follow the communication protocol with the Log Service from POL.
- The LogEntryService interface and its implementing class, LogEntryServiceImpl, offer a high level object oriented interface to the logger classes (shown at the bottom of Figure 12). The LogEntryService is used to open and close secure logging sessions, but also to record log entries, both locally and via the Log Service.
- The logger classes (shown at the bottom of Figure 12) offer developers an object oriented interface to record customized log messages:
    - DataLogger is used to record any CRUD operation on user data;
    - PrivacyLogger is used to record modifications to the privacy policy or user relationship operations;
    - AppLogger is used to record any CRUD operations on 3rd party applications and access token operations (generate, issue, refresh, revoke, verity)
    - AccessLogger is used to record user actions such as: de-/registration, log-in/out and invitation sending.

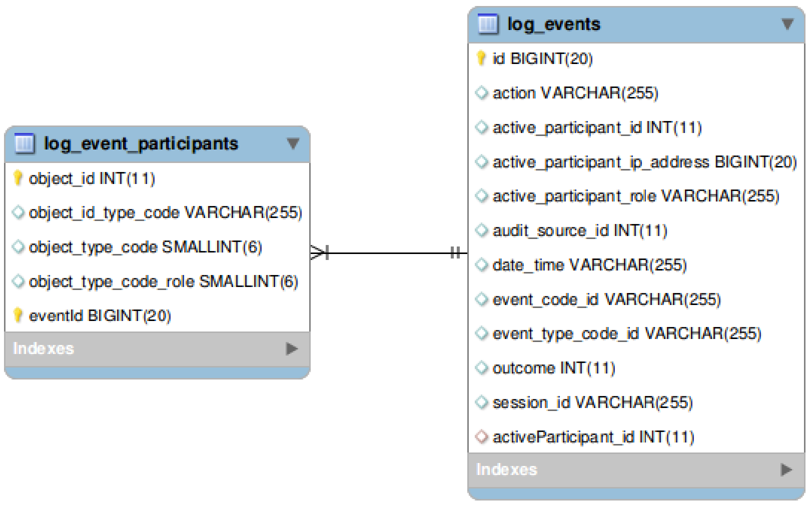

log_event_participants
object_id INT(11)
object_id_type_code VARCHAR(255)
object_type_code SMALLINT(6)
object_type_code_role SMALLINT(6)
eventId BIGINT(20)
Indexes
log_events
id BIGINT(20)
action VARCHAR(255)
active_participant_id INT(11)
active_participant_ip_address BIGINT(20)
active_participant_role VARCHAR(255)
audit_source_id INT(11)
date_time VARCHAR(255)
event_code_id VARCHAR(255)
event_type_code_id VARCHAR(255)
outcome INT(11)
session_id VARCHAR(255)
activeParticipant_id INT(11)
Indexes

Figure 13 : Database table structure

2.4.3.4.1 Open Logging Session

The following steps must be executed in order to open a secure logging session (as shown in the first part of sequence diagram from Figure 14):

1. The Healthcare TPaaS creates a new message M0, which contains: the current timestamp, the shared secret A0 (randomly chosen) and the cryptographic certificate of the Healthcare TPaaS; then it signs and encrypts M0;
2. The Healthcare TPaaS forms the first log entry containing message M0 and a unique identifier of the log file and protects this log entry by a hash-chain field and MAC field that are computed using A0; finally it sends the signed and encrypted M0 to the Log Service.
3. When the Log Service receives M0 it verifies if the Healthcare TPaaS has a valid certificate and if so, it forms an acknowledgement message M1 containing a hash of M0; after signing and encrypting M1 it sends the resulting message to the Healthcare TPaaS.
4. When the Healthcare TPaaS receives the encrypted message from the Log Service it decrypts it and checks the signature. If the signature is valid, it forms a new log entry containing message M1 and it protects this message with a hash-chain and MAC fields that are computed using a hash of A0.

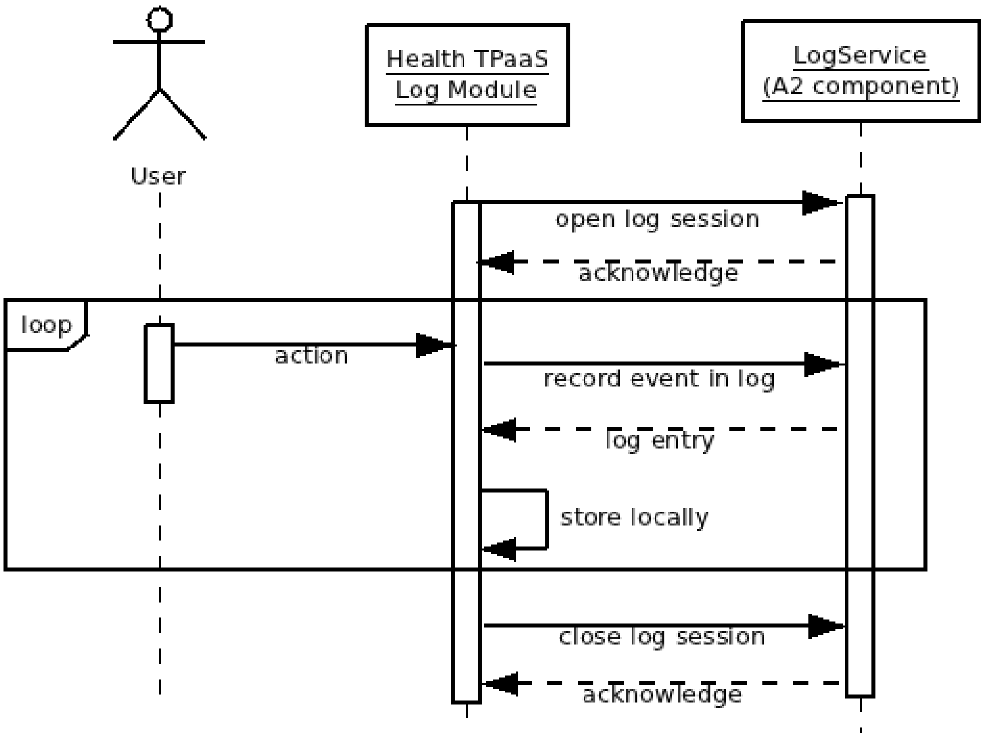


Figure 14 : Sequence diagram for recording log events

2.4.3.4.2 Record Log Entries

After the logging session has been successfully opened, the Healthcare TPaaS can record a log entry via the Log Service. As shown in the loop from Figure 14, whenever a user performs a certain action the following steps are executed by the Log Module of the Healthcare TPaaS:

1. The log entry is formed by the Log Module and a signed message with the log entry is sent to the Log Service.
2. The Log Service checks the signature of the sender; it computes the hash-chain value and the message authentication code for this entry and returns the log entry with the session ID value corresponding to it.
3. The Log Module receives the message containing the log entry from the Log Service, and it records the log entry locally in the database.

Figure 15 show a high-level view of the format of a log entry and how new field are computed before it is added to the log. Following list of notations are used in the figure.

- W_j represents the type of the log entry (e.g. data access, authentication, registration, etc.);
- $E_K(D_j)$ represents the log entry information symmetrically encrypted with the symmetric key K;
- Y_j represents a link from the hash chain that binds the log entry sequence and protects their order; note that this value is a hash of the corresponding field of the previous log entry and the previous two values form the current log entry;
- Z_j represents a Message Authentication Code of Y_j that protects the entire log entry, because the computation of Y_j also involves the values W_j and $E_K(D_j)$.

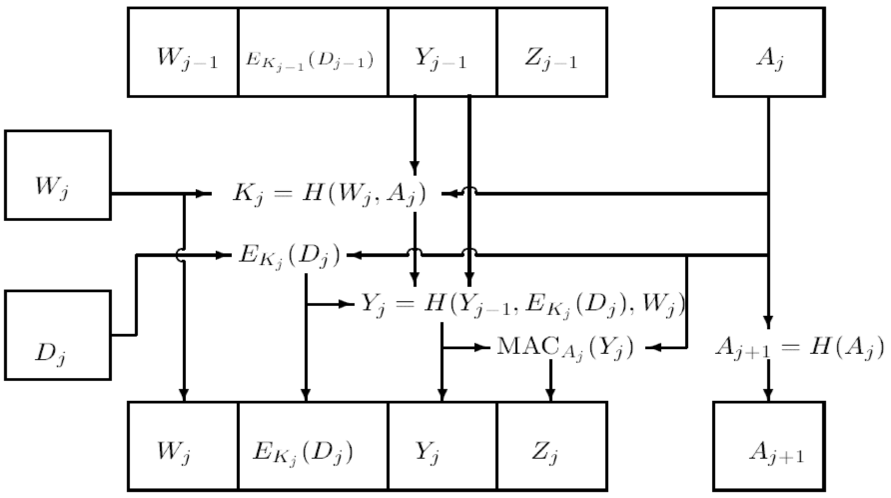


Figure 15 : Adding a log entry to the log (3)

A_j represents an evolving cryptographic key, which is used to compute the values of the last three fields from one log entry. After the fields of the log entry are computed, a hash of A_j is computed and stored, while A_j is irretrievably discarded. In case the untrusted party is compromised by an attacker, s/he will not be able to tamper with previously recorded log entries without being detected. This is due to the fact that s/he cannot compute the reverse of the cryptographically secure hash function and obtain the key needed to modify the fields of the previously recorded log entries.

2.4.3.4.3 Close Logging Session

A logging session is limited to a number of entries fixed upon setup. After this number of log entries is recorded, the session is closed and a new session is reopened. If the logging session would remain forever open, the truncation and delayed deletion attacks could be executed on the log file as presented in (4). However, once a log session is closed a special log entry is written at the end of the log file and the evolving cryptographic key used to compute subsequent log entries for that session is discarded. Thereafter, these two attacks cannot be executed anymore.

2.4.3.5 Audit Module

The purpose of the Audit Module is to offer users of the Healthcare TPaaS the possibility to perform auditing regarding the actual usage of their personal data. The Audit Module uses several classes from the Log Module presented in the previous section, and its structure is shown using a gray background in Figure 16.

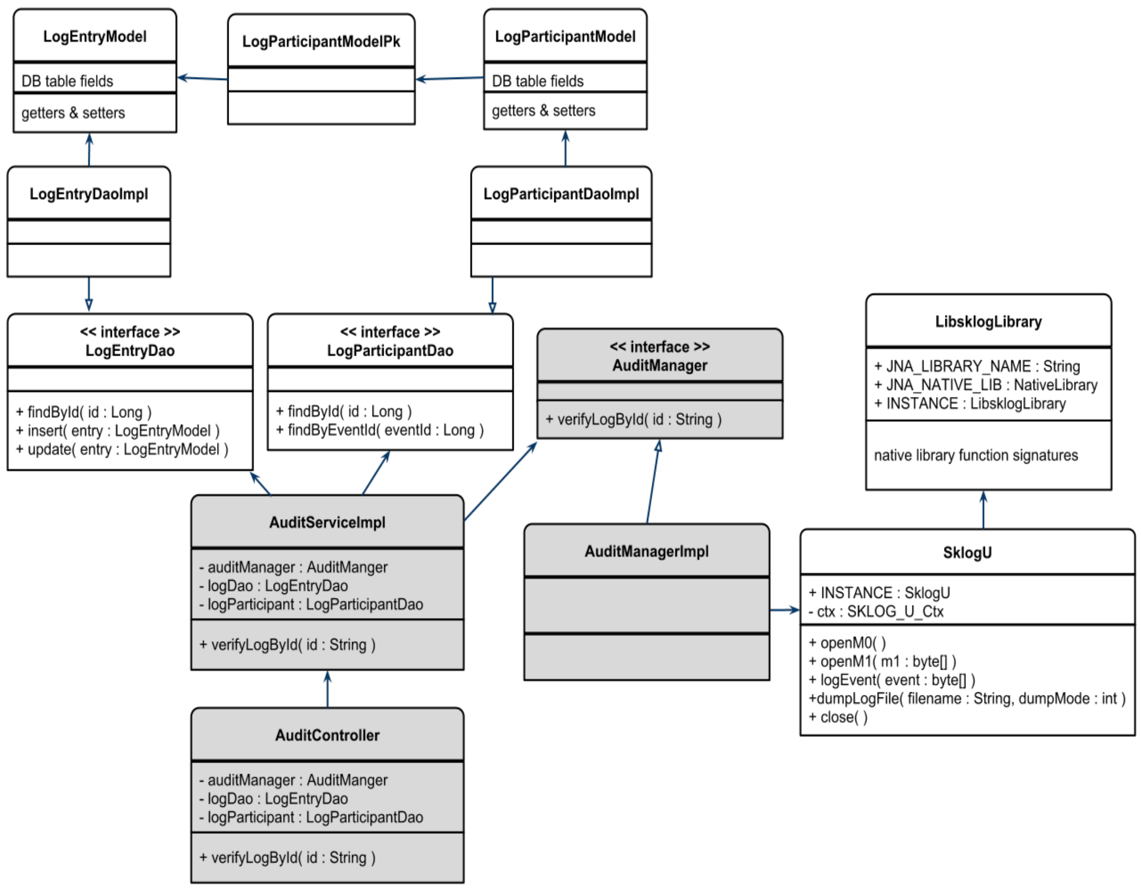

LogEntryModel
DB table fields
getters & setters
LogParticipantModelPk
LogParticipantModel
DB table fields
getters & setters
LogEntryDaoImpl
LogParticipantDaoImpl
<< interface >> LogEntryDao
+ findById(id : Long)
+ insert(entry : LogEntryModel)
+ update(entry : LogEntryModel)
<< interface >> LogParticipantDao
+ findById(id : Long)
+ findByEventId(eventId : Long)
<< interface >> AuditManager
+ verifyLogById(id : String)
LibsklogLibrary
+ JNA_LIBRARY_NAME : String
+ JNA_NATIVE_LIB : NativeLibrary
+ INSTANCE : LibsklogLibrary
native library function signatures
AuditServiceImpl
- auditManager : AuditManger
- logDao : LogEntryDao
- logParticipant : LogParticipantDao
+ verifyLogById(id : String)
AuditManagerImpl
SklogU
+ INSTANCE : SklogU
- ctx : SKLOG_U_Ctx
+ openM0( )
+ openM1(m1 : byte[])
+ logEvent(event : byte[])
+dumpLogFile(filename : String, dumpMode : int)
+ close( )
AuditController
- auditManager : AuditManager
- logDao : LogEntryDao
- logParticipant : LogParticipantDao
+ verifyLogById(id : String)

Figure 16 : Class diagram for Audit Module

Figure 16 shows classes implementing the controller component of the Model-View-Controller software design pattern. The model component was presented previously as part of the Log Module and the view component is implemented as JSP files, not as Java classes, therefore they are not shown in the class diagram.

2.4.3.5.1 Audit Dashboard

The Audit Dashboard in Figure 18 is a web based GUI, which offers end-users information about the events recorded by the Log Module. Access control is used in order to prevent vertical or horizontal privilege escalation.

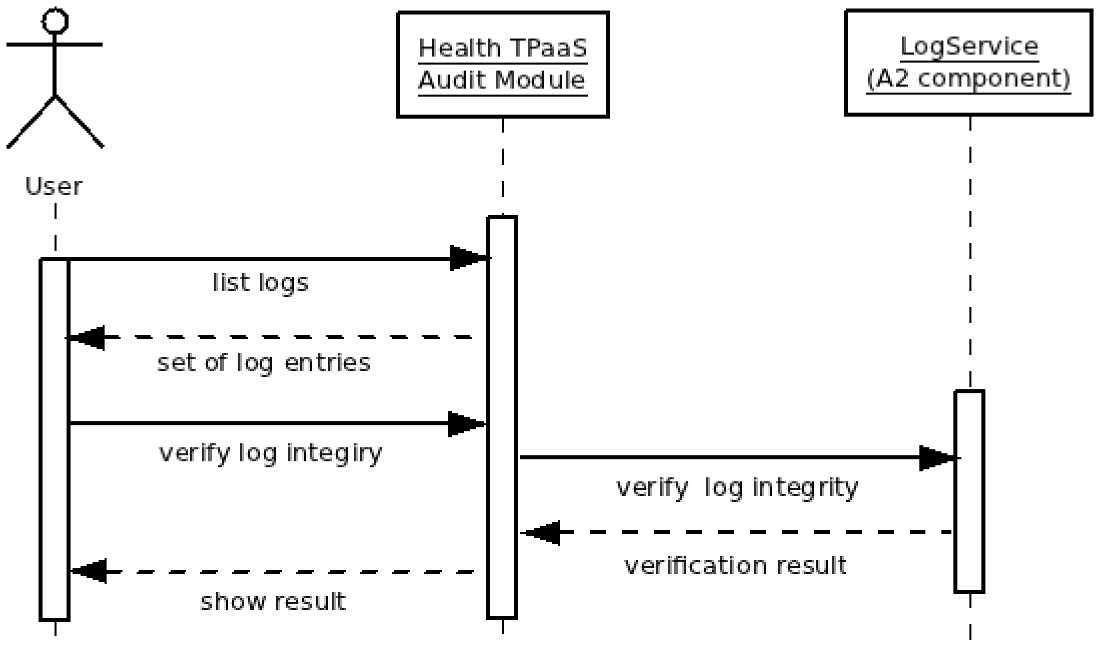


Figure 17 : Sequence diagram for verifying log files

Figure 17 shows the sequence diagram for verifying log files. Verification involves the interaction between 3 parties: the end user of the Healthcare TPaaS, the Audit Module and the Log Service. The steps from the figure are:

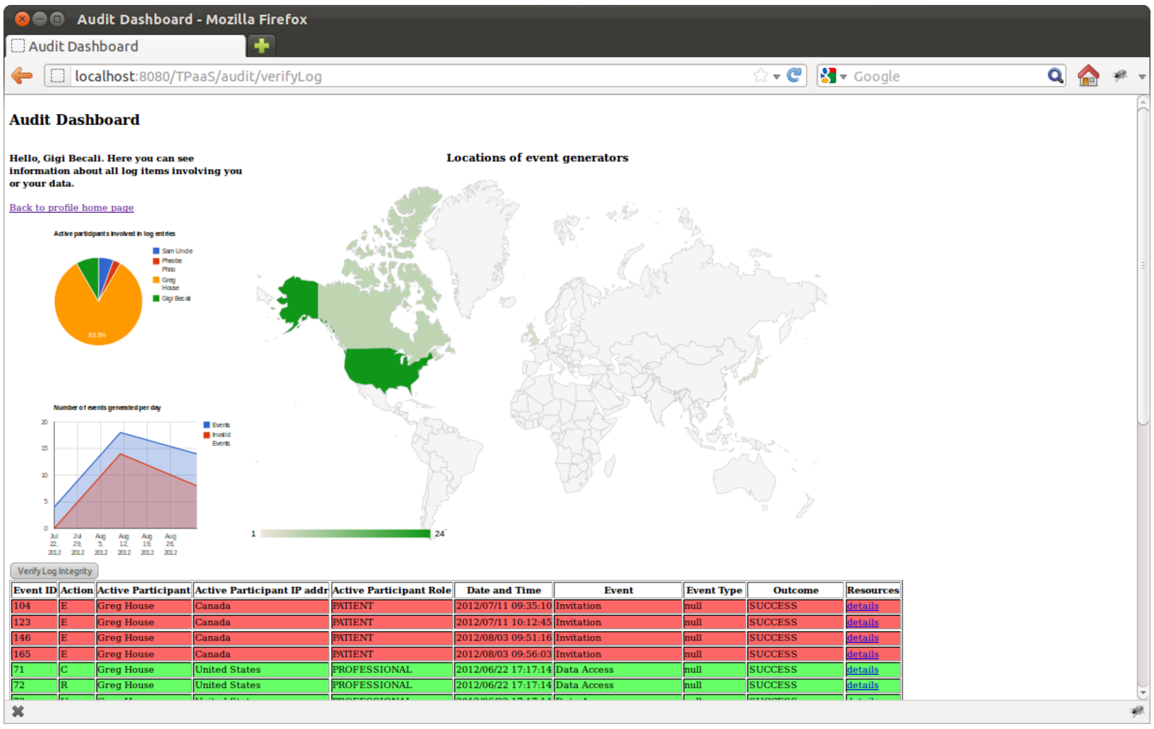
1. After the user has logged into the Healthcare TPaaS with a valid user account he can choose to view the list of log entries involving his/her personal data.

2. The Audit Module of the Healthcare TPaaS retrieves the log entries corresponding to the current user from the database, and then displays the Audit Dashboard depicted in Figure 18.

    a. A pie-chart indicating the number of events generated by different users that accessed the current user’s data;
    b. A geo-chart showing the locations of the users who generated the events; the heat-map indicates the number of events for each country;
    c. A line-chart indicating the number of events generated on each date;
    d. A table containing detailed information of every log entry corresponding to the end-user.
3. The end-user can then choose to verify the integrity of the log entries by a simple click on a button.
4. The Audit Module forwards the verification request to the Log Service, by sending it the log file containing the corresponding log entries. Note that the end-

user is not shown the entire log file since this may contain information that is not related to him/her.

5. The Log Service checks the integrity of the log file and returns the verification result to the Audit Module. Log integrity is verified by checking the message authentication code of each log entry and the integrity of the hash-chain between the log entries.
6. The Audit Module then updates the GUI of the end-user showing the events that passed the verification with a green background and the ones that did not pass verification with a red background in Figure 18. The line-chart is also updated and shows both the total number of log entries and the number of entries that did not pass the integrity check of the Log Service.



Audit Dashboard

Hello, Gigi Becali. Here you can see information about all log items involving you or your data.

Back to profile home page

Event ID	Action	Active Participant	Active Participant IP addr	Active Participant Role	Date and Time	Event	Event Type	Outcome	Resources
104	E	Greg House	Canada	PATIENT	2012/07/11 09:35:10	Invitation	null	SUCCESS	details
123	E	Greg House	Canada	PATIENT	2012/07/11 10:12:45	Invitation	null	SUCCESS	details
146	E	Greg House	Canada	PATIENT	2012/08/03 09:51:16	Invitation	null	SUCCESS	details
165	E	Greg House	Canada	PATIENT	2012/08/03 09:56:03	Invitation	null	SUCCESS	details
71	C	Greg House	United States	PROFESSIONAL	2012/06/22 17:17:14	Data Access	null	SUCCESS	details
72	R	Greg House	United States	PROFESSIONAL	2012/06/22 17:17:14	Data Access	null	SUCCESS	details

Figure 18 : Audit Dashboard

The charts from the Audit Dashboard were developed using the Chart Tools offered as a free service by Google (9). The chart types that were used are: “Pie Chart” and “Area Chart” from the “`corechart`” package and “Geo Chart” from the “`geochart`” package of Google Chart Tools.

The mapping of the IP address to the country of origin was implemented using the Free IP to Country Database for IPv4 (10) distributed under the GNU Public License v3 (11) by the Webcast77 company. The database is freely available for download in the form of a comma separated value (CSV) file.

2.4.4 User interface

The UI module implements all necessary web interfaces to allow each user to interact with the platform and manage policies for data access and friendship. The selected platform for implementing the UI was Liferay[12]. Liferay is an enterprise web platform for building business solutions which is easy to customize and use, and already provides broad product capabilities such as:

- Content & Document Management
- Web Publishing and Shared Workspaces
- Enterprise Collaboration
- Social Networking and Mash-ups
- Enterprise Portals and Identity Management

Some screenshots of the web portal implemented are available in Chapter 4, with the user manual.

2.4.5 DB Management

The diagram in Figure 19 illustrates how policies and relationships user-user and user-app are managed into data models.

Users of the platform are divided into:

- Platform's owner
- Third party
    - App managers
    - App developer
- End users
    - Generic end users (patients)
    - Professional

Relationships among users are allowed only between end-users and more specifically among patient-patient and patient-professional. Please note the double connection between entity Patient and entity user2user, while the entity professional and user2prof has only one connection. This to underline that the relation user1→user2 is different between user2→user1 and that between professional and patents can exists only the relation patient-professional (in which the professional can see patent's data but not vice-versa).

Resources correspond to a type of data/action that can be used under a specific policy. Example of data resource can be light data, sleep data while examples of actions can be: Blood analysis, psychical activity analysis, etc.

Application managers can use application. In that case applications are called instances that are applications that have a specific workspace.

[12] http://www.liferay.com

2.4.6 DB records

The following database schema in Figure 20 represents the current ER design for the TPaaS. The idea is to build a data model that reflects, partially, the distinction from user type and the differences among accounts.

The idea to implement such data model is driven by the idea that having a data-model that reflects the system functionalities requires less code to implement the logic to maintain the database coherent and it leads to a better and robust system.

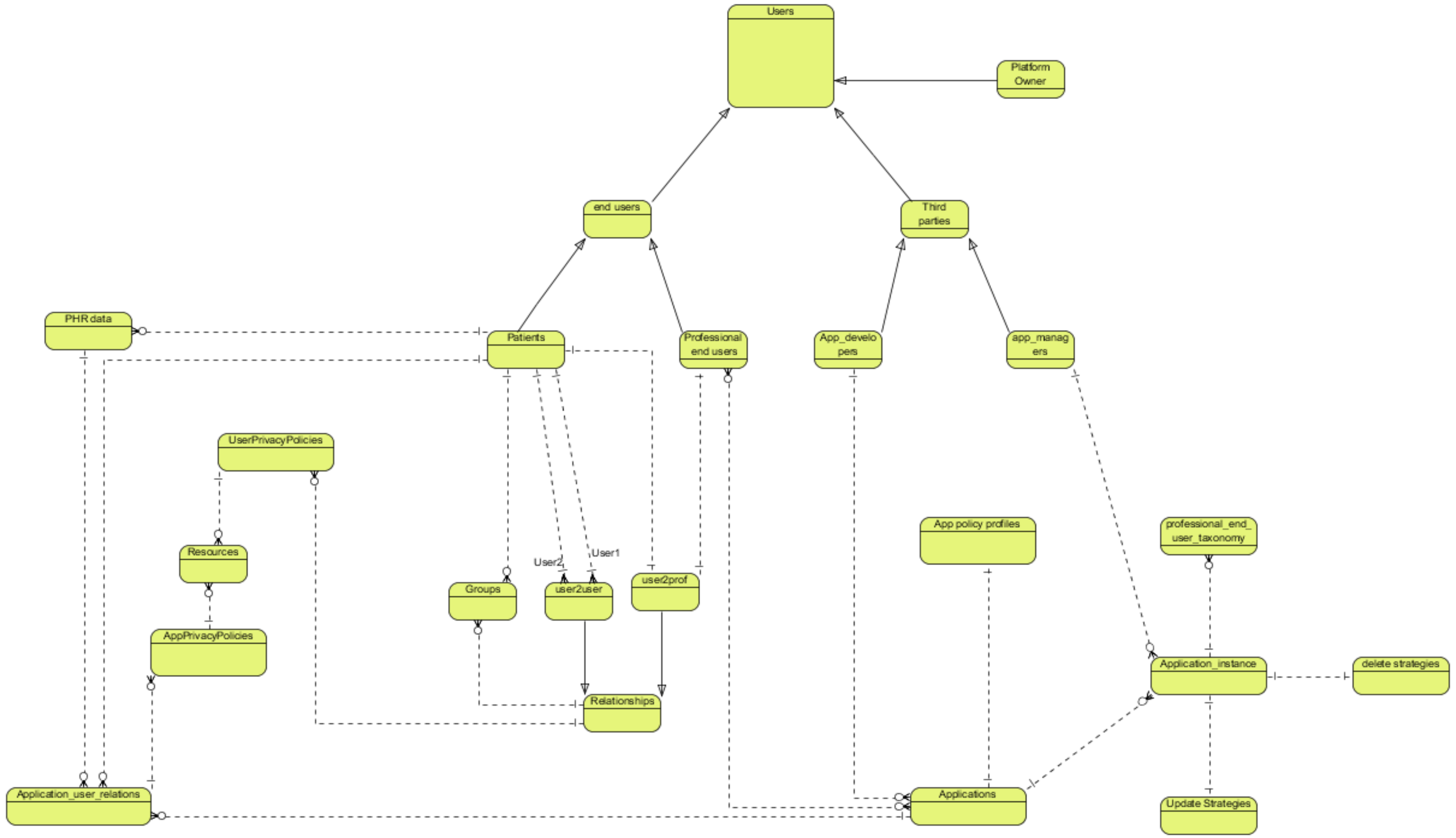


Figure 19 : ER diagram that highlights the policies management and user/ app relationships

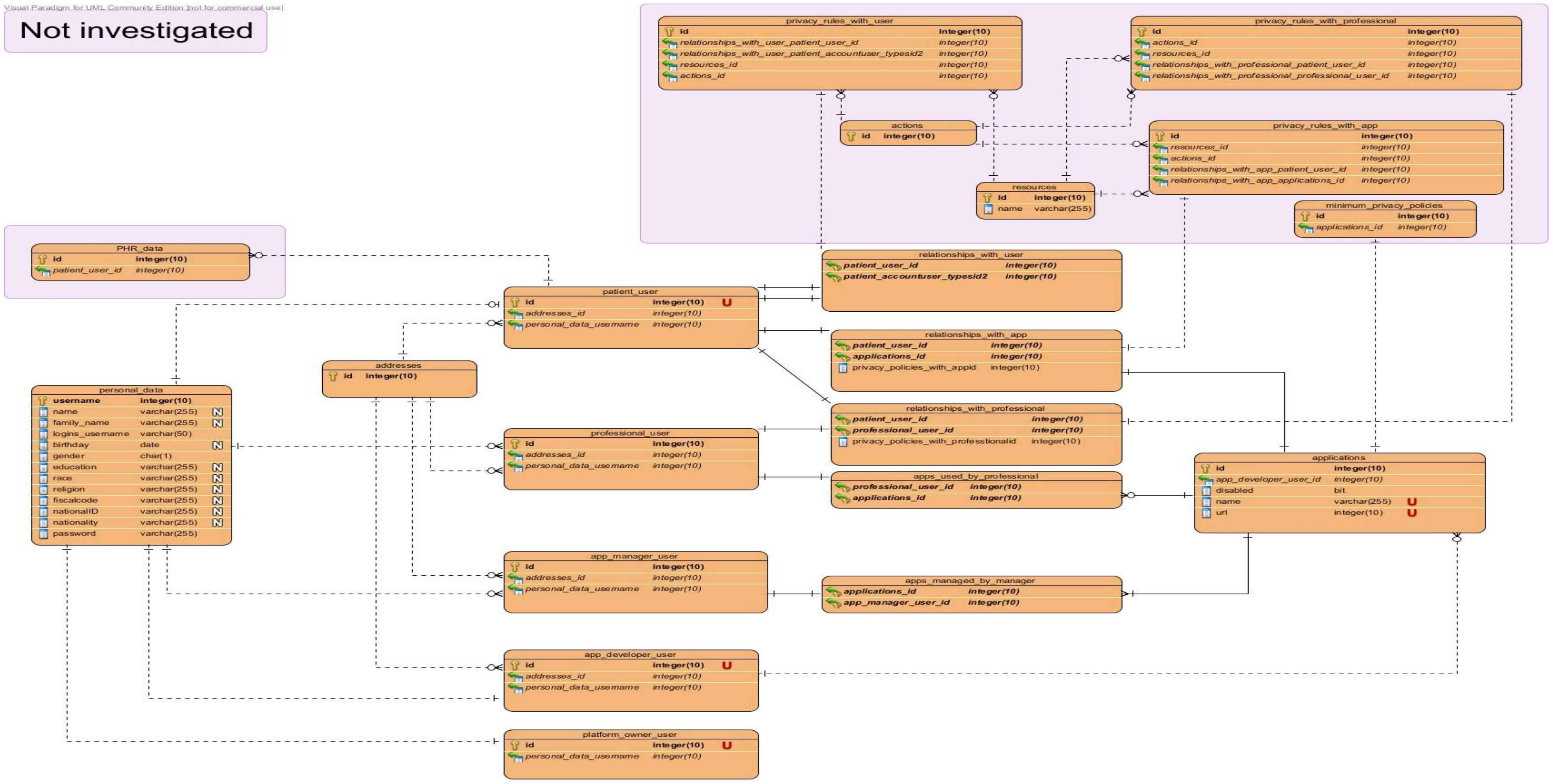

Visual Paradigm for UML Community Edition [not for commercial use]
Not investigated
privacy_rules_with_user
id integer(10)
relationships_with_user_patient_user_id integer(10)
relationships_with_user_patient_accountuser_typesid2 integer(10)
resources_id integer(10)
actions_id integer(10)
privacy_rules_with_professional
id integer(10)
actions_id integer(10)
resources_id integer(10)
relationships_with_professional_patient_user_id integer(10)
relationships_with_professional_professional_user_id integer(10)
actions
id integer(10)
privacy_rules_with_app
id integer(10)
resources_id integer(10)
actions_id integer(10)
relationships_with_app_patient_user_id integer(10)
relationships_with_app_applications_id integer(10)
resources
id integer(10)
name varchar(255)
minimum_privacy_policies
id integer(10)
applications_id integer(10)
PHR_data
id integer(10)
patient_user_id integer(10)
relationships_with_user
patient_user_id integer(10)
patient_accountuser_typesid2 integer(10)
patient_user
id integer(10) U
addresses_id integer(10)
personal_data_username integer(10)
relationships_with_app
patient_user_id integer(10)
applications_id integer(10)
privacy_policies_with_appid integer(10)
addresses
id integer(10)
personal_data
username integer(10)
name varchar(255) N
family_name varchar(255) N
logins_username varchar(50)
birthday date N
gender char(1)
education varchar(255) N
race varchar(255) N
religion varchar(255) N
fiscalcode varchar(255) N
nationalID varchar(255) N
nationality varchar(255) N
password varchar(255)
relationships_with_professional
patient_user_id integer(10)
professional_user_id integer(10)
privacy_policies_with_professionalid integer(10)
professional_user
id integer(10)
addresses_id integer(10)
personal_data_username integer(10)
apps_used_by_professional
professional_user_id integer(10)
applications_id integer(10)
applications
id integer(10)
app_developer_user_id integer(10)
disabled bit
name varchar(255) U
url integer(10) U
app_manager_user
id integer(10)
addresses_id integer(10)
personal_data_username integer(10)
apps_managed_by_manager
applications_id integer(10)
app_manager_user_id integer(10)
app_developer_user
id integer(10) U
addresses_id integer(10)
personal_data_username integer(10)
platform_owner_user
id integer(10) U
personal_data_username integer(10)

Figure 20 : Database tables for user taxonomy and accounts. (Relationships, privacy policies and other aspects are omitted for clarity purposes)

2.4.7 DSL box

The DSL component plays a core action inside TPaaS. Basically its function is to receive all the activities that have to be performed (either from the UI component or from the RESTful API component). It processes every request by using a specific language and executes it by extracting its semantics.

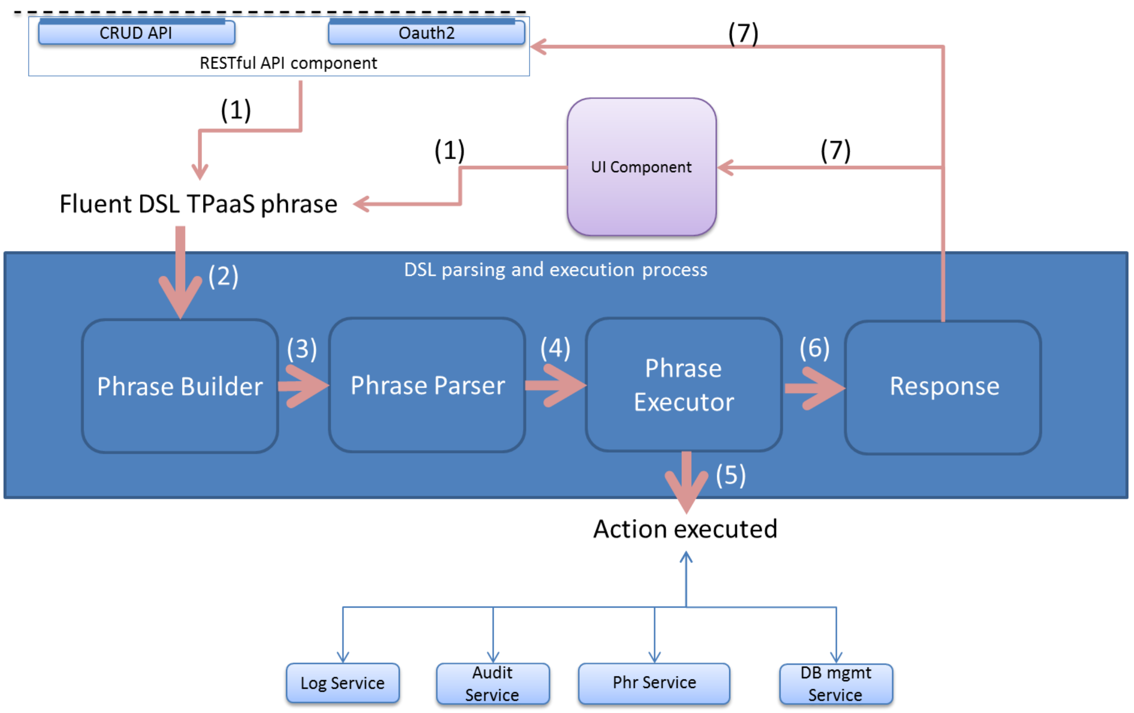


Figure 21 : DSL component interacts with other components in the Healthcare TPaaS

2.4.7.1 DSL implementation

Figure 21 explains how the DSL component interacts with other components into Healthcare TPaaS. We now provide the flow of a single action.

(1) The UI component or the RESTful API component receives a command from a user or a specific app.

(2) The Command is then translated into a TPaaS-DSL phrase through a fluent interface

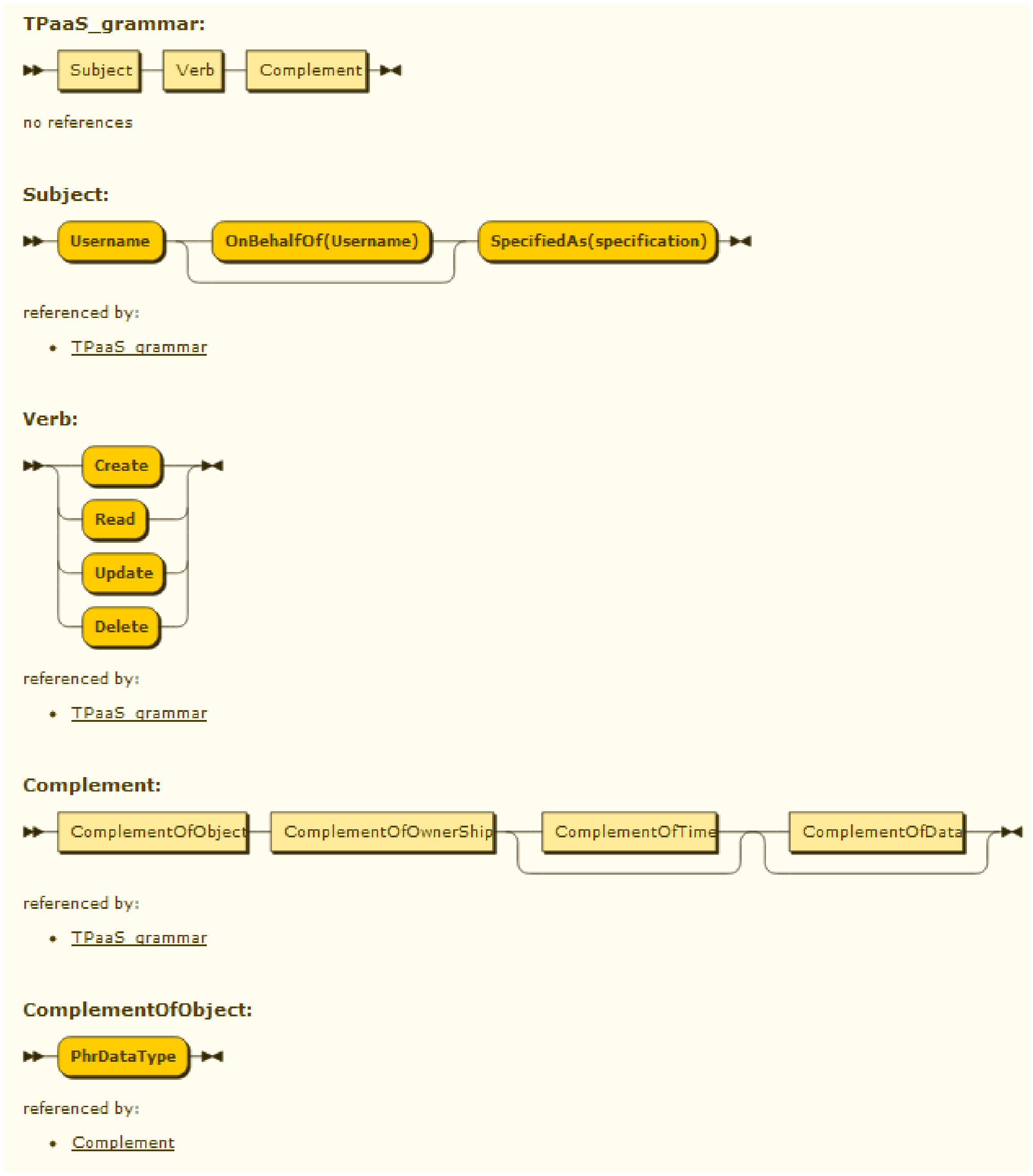
(3) The phrase (syntactically correct but not semantically) is then feed to the parser which checks if sematic meaning of the phrase.

(4) The phrase is ready to be executed if it is syntactically and semantically correct, If so, the phrase is executed.

(5) The execution uses services of lower level using the help of specific adapters (Gang of four (E. Gamma))

(6) Subsequently, the execution output of the phrase is packaged in order to be Sent back to the caller (either the UI component or the RESTful API component).

The Domain Specific Language that has been created in TPaaS respects a specific grammar described in figure below (for sake of understanding it has been simplified):



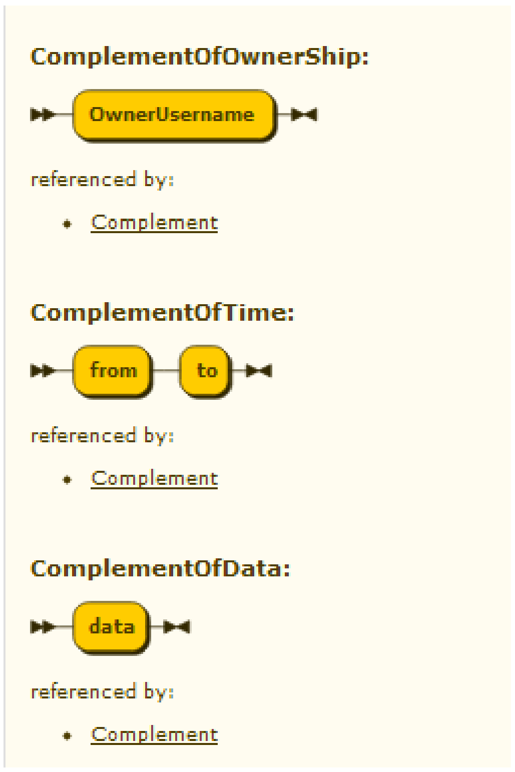


Figure 22 : DSL TPaaS grammar

DSL can be invoked by any high level layer (the UI or the REST Interface API) to perform any action into the platform.

A DSL phrase can be easily built by using its specific builder with an easy and straightforward fluent notation:

```
PhraseBuilder.currentUser()
     .read()
     .resource("bloodPressure")
     .of("OtherUsername")
     .from("12/02/2011).to("12/12/2012)
     .build().execute();
```

That expresses that the current logged user wants to read the blood pressure data of OtherUsername from 12/02/20111 to 12/12/2012

The PhraseBuilder object builds a well-written (syntax wise) phrase that is ready to be processed by the parser. This happens because the PhraseBuilder defines a sub-grammar based on the TPaaS grammar of Figure 22.

The Parser takes the phrase object and parses the phrase checking its semantics: checks that the users exist and that there is a proper friendship among them with a proper policy for the action requested (read, in our example). If everything is ok, the phrase is than passed to the executor that performs the requested action.

2.5 Software abstractions

The Healthcare TPaaS Platform has been layered in order to abstract the lower activities with high level activities. Figure 23 explains how TPaaS components are categorized into different logical layers.

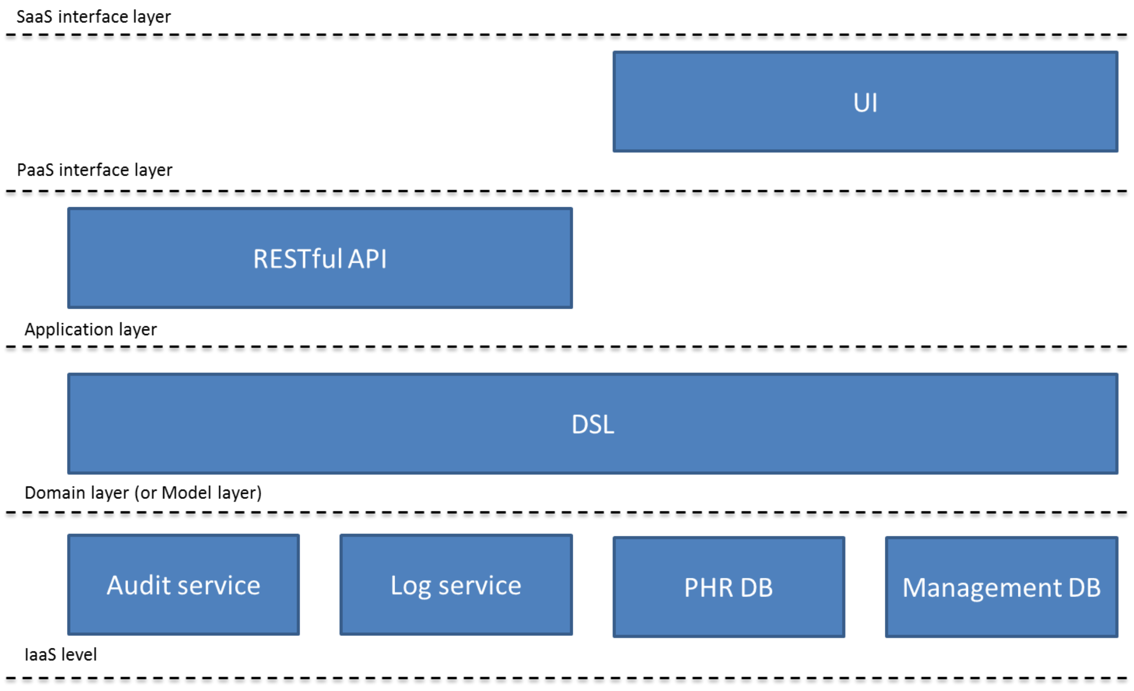


Figure 23 : Healthcare TPaaS layering

The Layering described above shows how all the modules are connected among them. Layering has been respected by considering (Evans, 2003)'s best practices in terms of application's layers.

Table 2 : Healthcare TPaaS layer description. As written in (Evans, 2003)

Layer	Description
SaaS/PaaS Interface	Responsible for showing information to the user and interpreting the user's commands. The external actor might sometimes be another computer system rather than a human user.
Application Layer	Defines the jobs the software is supposed to do and directs the expressive domain objects to work out problems. The tasks this layer is responsible for are meaningful to the business or necessary for interaction with the application layers of other systems. This layer is kept thin. It does not contain business rules or knowledge, but only coordinates tasks and delegates work to collaborations of domain objects in the next layer down. It does not have state reflecting the business situation, but it can have state that reflects the progress of a task for the user or the program.
Domain Layer (or Model Layer)	Responsible for representing concepts of the business, information about the business situation, and business rules. State that reflects

Layer	Description
	the business situation is controlled and used here, even though the technical details of storing it are delegated to the infrastructure. This layer is the heart of business software.
Infrastructure Layer	Provides generic technical capabilities that support the higher layers: message sending for the application, persistence for the domain, drawing widgets for the UI, and so on. The infrastructure layer may also support the pattern of interactions between the four layers through an architectural framework.

2.6 Conclusion

In this chapter is described the implementation of the Healthcare TPaaS. The architecture of the TPaaS is service-oriented, including three services: Management DB service, UI service, and Log and Audit service. Besides these services, a Domain Specific Language was implemented to receive all the operation instructions that have to be performed either from the UI component or from the RESTful API component.

The Healthcare TPaaS offers a set of on-board software security modules, including RESTful APIs, the encryption module, and the secure logging and auditing module. The RESTful APIs implement OAuth2.0 protocol allowing users to share their personal health records stored in the TPaaS without having to disclose their private credentials to any third party applications that are deployed on the TPaaS. Besides OAuth2.0, these APIs implement the OpenID protocol to allow users to log on to third party applications with their TPaaS ID. The data files and communications within the TPaaS are encrypted by the encryption module by using, for instance, CP-ABE cipher. The secure logging and auditing module logs every access from third party applications, ensures the integrity of the log files, and analyses whether activities have been performed as promised. The databases used in the TPaaS comprise the Management Database, which stores all the data necessary to maintain and perform basic social functionalities of the platform, and the Data Database, which contains Personal Health Records of users.

Chapter 3

Addressing home healthcare requirements using TClouds

Chapter Author:

Mina Deng, Zheyi Rong, Sebastian Banescu (PHI)

3.1 Security and privacy requirements integrated view in the cloud services

Table 3 identifies the relevant security and privacy requirements, alongside an estimate of the provision of these requirements in different layers of the cloud services, namely IaaS, PaaS and SaaS. The first two columns give an indication whether these requirements can be provided by standard commodity clouds such as Amazon and OpenStack.

1. The requirements for the TClouds home healthcare use case discussed in this document are based on the service-logic driven and the architecture-driven requirements, from the technical perspective with a focus on security and privacy. Relevant legislation shall be analyzed as a future step to gain a thorough understanding of security and privacy requirements.
2. Table 3 shows that both Amazon as well as OpenStack scores high on the protection of communication channels. This is due to the fact that both cloud computing structures provides virtual private networks per project. This shields the virtual machines reasonably well from other projects and eavesdroppers.
3. We analyzed both privacy and security requirements together as part of the Security Development Lifecycle (Lipner, 2006). Security is a necessary means to achieve privacy. In addition, in spite of the coexistence of security and privacy properties in one system, some security objectives might conflict with some privacy objectives. Therefore, it is important to consider the service-logic driven requirements to identify the desirable objectives of the system (e.g. repudiation and plausible deniability as privacy properties are not desired in the TClouds healthcare system). It is thus useful to consider requirements both for security and privacy together.
4. There are some tradeoffs between security and privacy with system performance, e.g. in terms of cost and efficiency (Deng, 2010). To facilitate this, it is important to find a proper balance between requirements and system performance, while taking the system's practical constraints into consideration. One tradeoff is between privacy and efficiency; the other is between privacy and cost. These two tradeoffs are interactive. Generally speaking, building security privacy in is usually at the price of increasing the implementation budget or lowering the performance efficiency of the system. Therefore, it is important to identify what are the relevant requirements and to apply risk assessment (ENISA, 2009) to prioritize these requirements.

Table 3 : Security requirements satisfaction w.r.t. Amazon and OpenStack

	IaaS			PaaS	SaaS
	Amazon	**Open Stack**	**TClouds infrastructure (T-IaaS)**	**TClouds platform (T-PaaS)**	**TClouds application (T-SaaS)**
Security requirements					
1. User (device at client side) authentication					√ Secure use, SSL/ TLS authentication (application/protocol level authentication with device key/secret)
2. Credentials that authenticate the users are well protected			√ ▪ IaaS level, authentication for VM image installation / instance instantiation ▪ Inside data store, secure, VM		
3. Log as a Service (and Integrity of the log files), Data/VM modification and deletion events are securely logged			√ Log as a Service at IaaS and PaaS level		√ Provide API interface (a list of events that need to be logged)
4. Proof of trustworthiness of environment (Communicating devices must be able to assess trustworthiness of the process they are communicating with), Integrity of VM			√ ▪ Option 1: no verification states of VM, but own Trustworthy IaaS cloud, e.g. Trusted Virtual Domain (TVD) ▪ Option 2: verification trustworthiness of OpenStack environment		
5. Access control mechanisms ensure administrative parts of VMs are inaccessible to unauthorized parties (outsider) or infrastructure cloud			√ (At VM level, but not at appl. level for users to access VMs) ▪ OpenStack, monitoring malicious insiders for secure cloud maintenance	√ Access Control as a Service	

	IaaS			PaaS	SaaS
	Amazon	**Open Stack**	**TClouds infrastructure (T-IaaS)**	**TClouds platform (T-PaaS)**	**TClouds application (T-SaaS)**
administrators (insider)			▪ Secure key management and secure image		
6. Isolation between VMs (separation of VMs in different physical machines, Integrity and Confidentiality of VM)	√ (Open issue: covert channel in the cloud communication layer)		√ ▪ Amazon extended interface (choice) ▪ Secure hypervisor (choice) ▪ Trusted Virtual Domain (security cluster of VMs)		
7. Data confidentiality (data at rest)			√ Physical layer, secure data store (protected by internal security policies) ▪ Trusted Virtual Domain (security cluster of VMs) ▪ Key management secure data storage[13]		√ Application layer encryption
8. Extra-monitor access is impossible, firewall, access control,	√				
9. Overcapacity failures are handled properly	√				

[13] No encryption for VM at rest, but data at rest are encrypted (e.g. to use common hypervisor prevent from picking information from the memory)

	IaaS			PaaS	SaaS
	Amazon	Open Stack	TClouds infrastructure (T-IaaS)	TClouds platform (T-PaaS)	TClouds application (T-SaaS)
10. Only private networks/secure channels used for communication	High				
11. Fine-grained control mechanisms are required to access data storage				√ Access Control as a Service	
12. Invalid input/queries are filtered out	Limited		√ Isolation between VMs		
13. Accessing communication channel is protected by encryption or access control or VPN	High				
14. Channel integrity is protected at application level					√ SSL/ TLS or WS-Security
15. Channel integrity is protected at middleware/OS					
16. The confidentiality of communication channel is protected using security of middleware/OS			√ e.g. with Trusted Virtual Domain with management component ensures only server can access trusted management component, can't access VMs in other TVD	√ SSL/ TLS or WS-Security	

	IaaS			PaaS	SaaS
	Amazon	Open Stack	TClouds infrastructure (T-IaaS)	TClouds platform (T-PaaS)	TClouds application (T-SaaS)
Privacy requirements					
17. Consent and policy compliance of the whole system					√ Contract and legal enforcements
18. Patient's content awareness					
19. Patient-centric protection: patient should be able to specify his/her privacy policy					√
20. Pseudonymize users IDs					√
21. Anonymity of patients of healthcare professionals					(Not required in the current use case)
22. Unlinkability of data flow	(Same as the security requirement of confidentiality and integrity of communication channel)				(Assume there is no need to deploy anonymous communication channels based on the current use case.)
23. Anonymization of data flow					
24. Use data anonymization / pseudonymization techniques to anonymize / pseudonimize the documents stored in the data store	(Same as the security requirement to enforce data protection by means of access control, while taking patient's privacy policy and consent into consideration.)				√
25. Enforce process (e.g. the state, the memory and administrative interfaces of the process) confidentiality by means of strong / secure access control	(Same as the security requirement of access control for VMs)				

	IaaS			PaaS	SaaS
	Amazon	Open Stack	TClouds infrastructure (T-IaaS)	TClouds platform (T-PaaS)	TClouds application (T-SaaS)
26. (Revocable) anonymity of privacy sensitive users (patients or healthcare professional) such that the entity will not be identified from the application process memories, by unauthorized parties					(Not required in the current use case)

3.2 The need of integration with A2 security components

Building Healthcare TPaaS on existing commodity clouds may suffer from security risks. However, these risks may be covered by A2 security components[14], as detailed below.

3.2.1 BFT-SMART/DivDB

Component Name	**BFT-SMaRt/DivDB**	
Partners involved	FFCUL	
Short Description	BFT-SMaRt is an Intrusion Tolerant State Machine Replication protocol. It can be used to replicate data across different replicas and tolerate up to *f* byzantine or crash faults having at least *3f+1* replica. To demonstrate the use of BFT-SMaRt we are developing DivDB, a Database replication protocol that replicates queries and transactions over the replicas managed by BFT-SMaRt. Our goal is to offer several methods defined by the JPA Specification requested by Efacec.	
Possible threat/attacks/Intrusions	BFT-SMaRt/DivDB can tolerate up to f faults, having at least 3f+1 replica. These faults can be replicas that are offline or too slow to reply, or even malicious replicas that can send any sort of requests.	
Possible Attacker/Intruder	Authorized and non-authorized application users, server administrators and users that gets access to servers.	
Security Threat that is covered by the component	*Spoofing*	Yes
	Tampering	Yes
	Repudiation	Yes
	Information Disclosure	No
	Denial of Service	No
	Elevation of privileges	Yes
Privacy Threat that is covered by the component	*Linkability*	No
	Identifiability	No
	Non-repudiation	No
	Detectability	No
	Information Disclosure	Yes
	Content unawareness	Yes
	Tampering	Yes

[14] See also TC-D2.4.2, Initial Component Integration, Final API Specification, and First Reference Platform, https://svn.tclouds-project.eu/trunk/ActivityA2-TrustedCloudPlatform/Documents/D-EU-Deliverables/TC-D2.4.2/TC-D2_4_2.pdf

Benefits with respect to existing solutions	*Vendor provided replication*. Database vendors such as Oracle and MySQL already have database replication mechanisms that can be used to replicate data. The advantage of using BFT-SMaRt/DivDB is vendor independent, providing diversity. This leverages security against bugs on a specific database. It also prevents malicious attacks on database replicas. Read only operations can perform faster compared with a single database approach. Also, when using the multi-leader protocol of DivDB, operations can be delegated to the closest replica, balancing the load against multiple replicas.
How to demonstrate the advantages of the component	It can be demonstrated by changing an existing application that makes use of JPA to store data and replace the current database with BFT-SMaRt/DivDB. If well configured, the data can be replicated in the databases defined. It can also be demonstrated that the protocol can tolerate attacks to *f* replicas, as long as *n-f* replicas continue to work properly, *n* being the number of existing replicas.

Security/privacy threats of the Healthcare TPaaS covered by BFT-SMART/DivDB:

- *Datacenter and cloud outages/failures*. The BFT-SMART component is able to tolerate f crash faults having at least 2f+1 replicas of storage (e.g. DBMS) in at least 3 different clouds/datacenters.
- *Data corruption tolerance from bugs, malicious insider attacks and intrusions*. The BFT-SMART component is able to tolerate f byzantine faults having at least 3f+1 replicas of storage (e.g. DBMS) in at least 4 different clouds/datacenters. Byzantine faults may be caused by malicious insiders (e.g. cloud provider).
- *Vulnerabilities of specific DBMS implementations*. The DivDB replication protocol allows creating replicas of a database schema using different DBMS vendor implementations (e.g. MySQL, PostgresSQL, HSQL, Firebird). If vulnerability is discovered in one of the DBMS implementations, this can be tolerated if at least 4 different DBMS vendors are used.
- *Vendor lock-in*. Because DivDB is vendor independent it facilitates migration from one DBMS vendor to another without significant costs.

Security properties offered by BFT-SMART/DivDB:

- Availability of data
- Integrity of data

3.2.2 Cloud of Clouds file systems: ICStore, DepSky

Component Name	**Cloud of clouds storage**		
Partners involved	IBM, FFCUL		
Short Description	Cloud of clouds storage is a middleware for storing data on many clouds. The main idea behind this component is to leverage many different clouds in order to increase robustness and security of a remote storage system solution. Our component tolerates minority of corrupted cloud storage providers, while still retaining liveness and correctness. Cloud of Clouds storage implementation is built such that it can work in two different modes, each tailored toward specific use cases: crash-tolerant mode that focuses on improving write performance and a Byzantine-tolerant mode, that trades off write performance for increased resilience. Modes are not interchangeable, in the sense that the system administrator must choose the operating mode during system's deployment. In order to tolerate f faults, the crash-tolerant mode requires 2f+1 replica, while the Byzantine-tolerant mode requires 3f+1 replica. Both variants assume that writers are correct.		
Possible threat/attacks/Intrusions	Crash-tolerant mode: - some (minority) of cloud storage providers becomes inaccessible - some (minority) of cloud storage providers introduce new data in the cloud		
	Byzantine-tolerant mode: - some (minority) of cloud storage providers corrupt existing data in the cloud - some (minority) of cloud storage providers becomes inaccessible - some (minority) of cloud storage providers introduce new data in the cloud		
Possible Attacker/Intruder	Cloud provider, attacker gaining access to the storage system of the cloud provider		
Security Threat (put Yes or No for each line)	Crash-tolerant	Spoofing	No
		Tampering	No
		Repudiation	No
		Information Disclosure	Yes
		Denial of Service	Yes
		Elevation of privileges	No
	Byzantine-tolerant	Spoofing	Yes
		Tampering	Yes
		Repudiation	Yes
		Information Disclosure	No
		Denial of Service	Yes
		Elevation of privileges	No
Privacy Threat (put Yes or No for each	Crash-tolerant	Linkability	No

line)			
		Identifiability	No
		Non-repudiation	No
		Detectability	No
		Information Disclosure	No
		Content unawareness	No
		Consent/policy non-compliance	No
	Byzantine-tolerant	Linkability	No
		Identifiability	No
		Non-repudiation	No
		Detectability	No
		Information Disclosure	No
		Content unawareness	No
		Consent/policy non-compliance	No
Benefits with respect to existing solutions	There is currently no existing middleware that leverages several clouds in order to increase security and robustness. The main advantage with our system is that it avoids single cloud lock-in, as data is replicated over multiple clouds. Moreover, our system improves read performance (compared to a single cloud), and could reduce cost of read operations, as it can always first read from the fastest or least expensive cloud.		
How to demonstrate the advantages of the component	Crash-tolerant: store data on the cloud crash, or prevent access to some clouds show that it is still possible to access data		
	Byzantine-tolerant: store data on the cloud corrupt some data on one (or minority) clouds verify that clients still fetch correct data		

Security/privacy threats of the Healthcare TPaaS covered by the crash-tolerant mode:

- *Datacenter and cloud outages/failures*. The crash-tolerant mode of the file system is able to tolerate f crash faults having at least 2f+1 replicas (for f=1 fault, this assumes the use of at least 3 different clouds/datacenters).
- *Data corruption due to concurrent writes*. The crash-tolerant mode is able to tolerate an unlimited number of concurrent writes to the same data file without any problems.
- *Information disclosure*. The crash-tolerant mode may use several encryption schemes (e.g. based on block ciphers) to protect the confidentiality of data stored in the file system. The key management unit of this mode allows storing keys both locally or on multiple clouds (i.e., secret sharing).

- *Vendor lock-in*. This mode is independent of any individual cloud provider. Therefore the user is technically unaffected if one cloud provider closes its business.

Security/privacy threats of the Healthcare TPaaS covered by DepSky:

- *Datacenter and cloud outages/failures*. The Byzantine-tolerant mode of the file system is able to tolerate f crash faults having at least 2f+1 replicas (for f=1 fault, this assumes the use of at least 3 different clouds/datacenters). This system nominally runs with 3f+1 replicas (at least 4 clouds for f=1 fault).
- *Data corruption tolerance from bugs, malicious insider attacks and intrusions*. The Byzantine-tolerant mode of the file system is able to tolerate f byzantine faults having at least 3f+1 replicas (at least 4 different clouds/datacenters, for f=1 fault). Byzantine faults are caused by malicious insiders (e.g. cloud provider).
- *Information disclosure*. This mode protects the confidentiality of data stored in the file system. Collusion attacks between cloud providers are not possible. At least two correct (honest) cloud providers have to be online for the data to be correctly retrieved to the client.
- *Vendor lock-in*. This mode is independent of any individual cloud provider. Therefore the user is technically unaffected if one cloud provider closes its business.

Security properties offered by CoC file systems:

- Confidentiality of data
- Availability of data
- Integrity of data
  - Against concurrent writes (the crash-tolerant mode)
  - Against Byzantine faults (the Byzantine-tolerant mode)

3.2.3 Log service

Component Name	**Log Service**	
Partners involved	POL	
Short Description	The main focus of the Log Service is to log and track events generated at multiple cloud layers (infrastructure, platform, and software). The Log Service is mainly based on the scheme for secure logging proposed by Schneier and Kelsey in *"Secure Audit Log to Support Computer Forensics"*. Such a scheme ensures the presence of the *forward integrity* property: if an attacker succeeds in compromising the log system, he cannot delete or modify log entries collected before his attack without being noticed.	
Possible threat/attacks/Intrusions	Log tampering, unprivileged access to the logs	
Possible Attacker/Intruder	Entity who abuses of its privileged position (e.g. sysadmin, cloud service provider) to delete and/or modify log entries.	
Security Threat	*Spoofing*	No

	Tampering	Yes
	Repudiation	Yes
	Information Disclosure	Yes
	Denial of Service	No
	Elevation of privileges	No
Privacy Threat	*Linkability*	No
	Identifiability	No
	Non-repudiation	Yes
	Detectability	Yes
	Information Disclosure	Yes
	Content unawareness	No
	Consent/policy non-compliance	Yes
Benefits with respect to existing solutions	There are no commercial or open source solutions that implement the Log Service features	
How to demonstrate the advantages of the component	Cloud Provider or more in general, a privileged entity, removes a log entry and the log reviewer can verify that the log is not integer anymore	

Security/privacy threats of the Healthcare TPaaS covered by the Log Service:

- *Tampering with log entries collected before an attack without being detected.* The logging scheme implemented by the Log Service ensures integrity of log entries and their order through cryptographic hash chains and message authentication codes. Users of the Healthcare TPaaS can use the Log Service to verify that the logs shown to them have not been tampered with by a malicious internal or external entity.
- *Log entry information disclosure to unauthorized parties.* The Log Service employs an access control system, based on log entry encryption, which regulates access to logs and prevents information disclosure to unauthorized entities. This also implies that a compromise of the Log Service storage unit does not lead to information disclosure since the log entries are encrypted and the attacker does not have the decryption key.
- *Repudiation of actions executed by an entity (e.g. user of third party application).* The entries recorded by the log service serve as evidence of actions executed by all entities using the Healthcare TPaaS. Non-repudiation is ensured only if the client of the Log Service is guaranteed to log all events that occur.
- *Detectability of an item of interest by an attacker.* The Log Service stores log entries in encrypted form. Therefore, an attacker cannot discriminate IOIs from the random noise.
- *Consent/policy non-compliance of third party applications and the Healthcare TPaaS.* The auditor may use the log files stored by the Log Service to verify whether the system actually complies with the advertised policies.

Security properties offered by the Log Service

- Tamper evident audit logs

- Confidentiality and access control of audit logs
- Non-repudiation of actions
- Un-detectability of items of interest
- Consent/policy compliance

Integration with A3 Logging Module

The A3 logging module, developed for the Healthcare TPaaS at PHI, is the untrusted party in the Schneier and Kelsey logging scheme. It initiates a secure logging session with the Log Service provided by A2, where a symmetric key is established to protect the integrity of the logging session. The A3 logging module offers developers an object-oriented interface to create customized log entries for the Healthcare TPaaS, that is:

- User log-in and log-out;
- CRUD operations on data;
- CRUD operations on third party applications;
- Modifications to privacy policies;
- Access token management events.

The Healthcare TPaaS needs to use the Log Service in order to provide a guarantee to its users (e.g. patients, third party application vendors, etc.) that the log has not been tampered with. A logging session in Healthcare TPaaS includes a number of entries fixed upon setup. The entries involve actions performed by any user or third party application. Users of the Healthcare TPaaS are offered the possibility to perform auditing. More specifically:

- Patients should only see all accesses to their personal healthcare records or authentication actions;
- Third party application managers should only see all CRUD operations involving their applications;
- And so on for other types of users.

This implies, searching in the log for entries that match a specified criteria. Since search capabilities are not currently offered by the Log Service, we store a copy of the log entries in a local MySQL database. This allows us to efficiently search in the log.

3.2.4 Remote Attestation

Component Name	**Remote Attestation Service**
Partners involved	POL
Short Description	The main objective of the Remote Attestation Service subsystem is to assess the integrity of physical hosts that are part of the Cloud infrastructure. This subsystem will determine the integrity level of a platform by verifying that the code of executed software is known (i.e. its digest is present in a database of reference values) and that installed packages are up to date.
Possible threat/attacks/Intrusions	- Exploitation of vulnerabilities on the hypervisor and access to the physical host - Compromised physical host that tries to attack other hosts in the Cloud network
Possible Attacker/Intruder	- Malicious insiders, meaning malicious employees of the Cloud Provider (while Cloud Provider itself is instead considered

	trusted) - External attackers	
Security Threat	*Spoofing*	Yes
	Tampering	Yes
	Repudiation	Yes
	Information Disclosure	Yes
	Denial of Service	Yes
	Elevation of privileges	Yes
Privacy Threat	*Linkability*	Yes
	Identifiability	Yes
	Non-repudiation	Yes
	Detectability	Yes
	Information Disclosure	Yes
	Content unawareness	Yes
	Consent/policy non-compliance	Yes
Benefits with respect to existing solutions	1) Better management of Cloud infrastructure, especially early detection of changes/alterations in physical nodes. 2) Actually there are no services which allow verifying the integrity of commonly used Linux distributions, such as Fedora and Ubuntu. 3) Certification that no unknown or malicious programs are running inside the VM	
How to demonstrate the advantages of the component	For the Cloud Provider: 1) A user should specify the desired integrity level for the physical host where he wants to deploy a new virtual machine 2) The OpenStack Scheduler searches a Cloud node that meets the user requirements by asking to the Remote Attestation Service the integrity level of each node 3) The Remote Attestation Service verifies the integrity level of Cloud nodes by comparing digests measured by the Operating System to values in a database of known measurements 4) If a physical host is compromised, its integrity status is corrupt and the virtual machine will not be scheduled in this host 5) The OpenStack Scheduler deploys a new virtual machine in the first node that meets the user requirements For the Cloud Customer: 1) The Cloud customer specifies which software libraries and executable should be running with the VM. 2) The Cloud Infrastructure provides an interface to the Cloud customer to remotely verify that all the VMs are running proper software. For the End User (e.g. of the Healthcare TPaaS): 1) The Cloud Customer provides his software to a certified third party (e.g. IT Auditor), trusted by all End Users. The software will be installed on a standard VM image, which has been previously measured by the Cloud Provider.	

	2) The trusted third party performs a review/audit of the software to check that it is compliant with legal regulations and SLAs advertised by the Cloud Customer (e.g. the application logs all necessary events). 3) The Cloud Provider records all necessary measurements of the software (binaries and libraries) reviewed by the Auditor. 4) The Cloud Customer deploys a new virtual machine by using a standard image and installs his software. 5) The End User is protected against a malicious employee of the Cloud Customer who may try to alter the VM image by uploading or modifying binaries or libraries. Because the Cloud Provider will see whether the VM image has been modified, he could alert End Users or the proper authorities.

Security/privacy threats of the Healthcare TPaaS covered by the Remote Attestation:

- *All kind of attacks:* we listed all kind of possible attacks, as the remote attestation can detect any modification of the software running inside physical nodes and virtual machines.
- *Modification of any library or binary running on the VM*

Security properties offered by Remote Attestation

- Remote attestation of physical nodes (limited to libraries and executables)
- Remote attestation of software within VMs (limited to libraries and executables) without integrity guarantee of the measurements list (an attacker could modify the list of measurements by deleting those records that would reveal the execution of a malicious program)

3.2.5 Secure Key Management (Cryptography-as-a-Service)

Component Name	**Secure Key Management ("Secret-less VMs") and secure block storage as a use-case of this**
Partners involved	TUDA, PHI, FSR, IBM
Short Description	Our key management component allows it to separate sensitive operations, such as crypto operations, and secrets, i.e. long-term secrets, to be separated from the workload VM. The separate crypto execution environment (currently denoted as "CryptoProxy") can be accessed explicitly from the VM (e.g., TPM, HSM and SmartCard) or implicitly (i.e., all I/O data traffic of certain streams such as writing/reading to an attached block storage, is automatically en-/decrypted)
Possible threat/ attacks/ Intrusions	External attacks (e.g., clients of the service implemented by A3) which intrude the service VM and extract secret keys or passwords from memory; Internal attacks (e.g., cloud administrators) which aim at dumping the memory and extracting secret keys or passwords from it.
Possible Attacker/Intruder	Cloud administrators operating in the privileged domain of the cloud; Cloud storage administrators; Other cloud customers to some extent (side-channels currently only

	partially prevented)	
Security Threat	*Spoofing*	No
	Tampering	Yes
	Repudiation	No
	Information Disclosure	Yes
	Denial of Service	No
	Elevation of privileges	No
Privacy Threat	*Linkability*	No
	Identifiability	No
	Non-repudiation	No
	Detectability	No
	Information Disclosure	Yes
	Content unawareness	No
	Consent/policy non-compliance	No
Benefits with respect to existing solutions	Having a trustworthy solution to store and use secrets such as crypto keys is currently not possible in clouds, while the usage of TPM, HSM, SmartCard in conventional datacenters for the protection of such credentials is standard; Leveraging virtualization technology for use-cases such as implicit encryption of storage in such a convenient way is currently only possible in manually configured datacenter setups or dedicated infrastructures such as TVDs	
How to demonstrate the advantages of the component	Several simple attacks can be demoed that work w/o our component (i.e., hypervisor) such as: 1) Dumping memory from the privileged domains and extracting secrets; 2) Intruding the service VM and stealing long term secrets; 3) Inspecting the image of the service VM for secrets.	

Security/privacy threats of the Healthcare TPaaS covered by Secure Key Management:

- *Tampering with data*. The Secure Key Management VM cannot be accessed by malicious users or cloud administrators. Therefore, they cannot obtain the keys needed to tamper with data from the storage of the customer VM, since this would require encrypting and decrypting the stored data. Furthermore if the encrypted data is tampered with, this will cause different (probably corrupt) results upon decryption by the Secure Key Management VM.
- *Information disclosure*. Since the Secure Key Management VM cannot be accessed by malicious users or cloud administrators, they cannot decrypt any of the encrypted data from the storage of the customer VM.

Security properties offered by Secure Key Management:

- Confidentiality of information stored in main memory of VM;
- Integrity of information stored in main memory of VM.

3.3 Conclusion

In this chapter, the security and privacy requirements of home healthcare applications are analyzed in different could layers (viz. IaaS, PaaS, and SaaS), and the fulfillments of these requirements are then discussed considering A2 security components.

We listed 16 security and 10 privacy requirements for a cloud platform; however, neither Amazon nor OpenStack can fulfill as many requirements as TClouds does. With the help of A2's security components, TClouds can:

- replicates data across different replicas and tolerate byzantine or crash faults (BFT-SMART/DivDB)
- leverages many different clouds in order to increase robustness and security of a remote storages system solution (Cloud of clouds storage)
- log and track events generated at multiple cloud layers (Log Service)
- assesses the integrity of physical hosts that are parts of the Cloud infrastructure (Remote Attestation Service).
- separates sensitive operations and secrets (Secure Key Management).

Therefore A3 is happy about the security and privacy requirements that could be supported by TClouds.

Chapter 4

User manual

Chapter Author:

Marco Abitabile, Marco Nalin (FSR), Sebastian Banescu (PHI)

In this chapter we describe the main functionalities of the home healthcare platform and application from a user perspective.

4.1 General Description

4.1.1 Installation

TPaaS is divided in different modules written in Java. Installation in not straight-forward and requires many tasks to install all the different modules.

For sake of simplicity we decided to make it available online. It is enough to surf at: http://tclouds-dev.eservices4life.org:8080

Here you have the chance to login/ create new accounts and play with TPaaS prototype.

At the time of writing auto-creation of users experiences some problems due to FSR's mailing system. In case the error persists one can anyway login by using the default users that has been inserted:

Table 4 : TPaaS demo user credential

Name	*email*	*password*
Alessandro	*test1@liferay.com*	*12345*
Bruno	test2@liferay.com	*12345*
Carmelo	test3@liferay.com	*12345*
Demetrio	test4@liferay.com	*12345*
Emanuele	test5@liferay.com	*12345*
Fabio	test6@liferay.com	*12345*

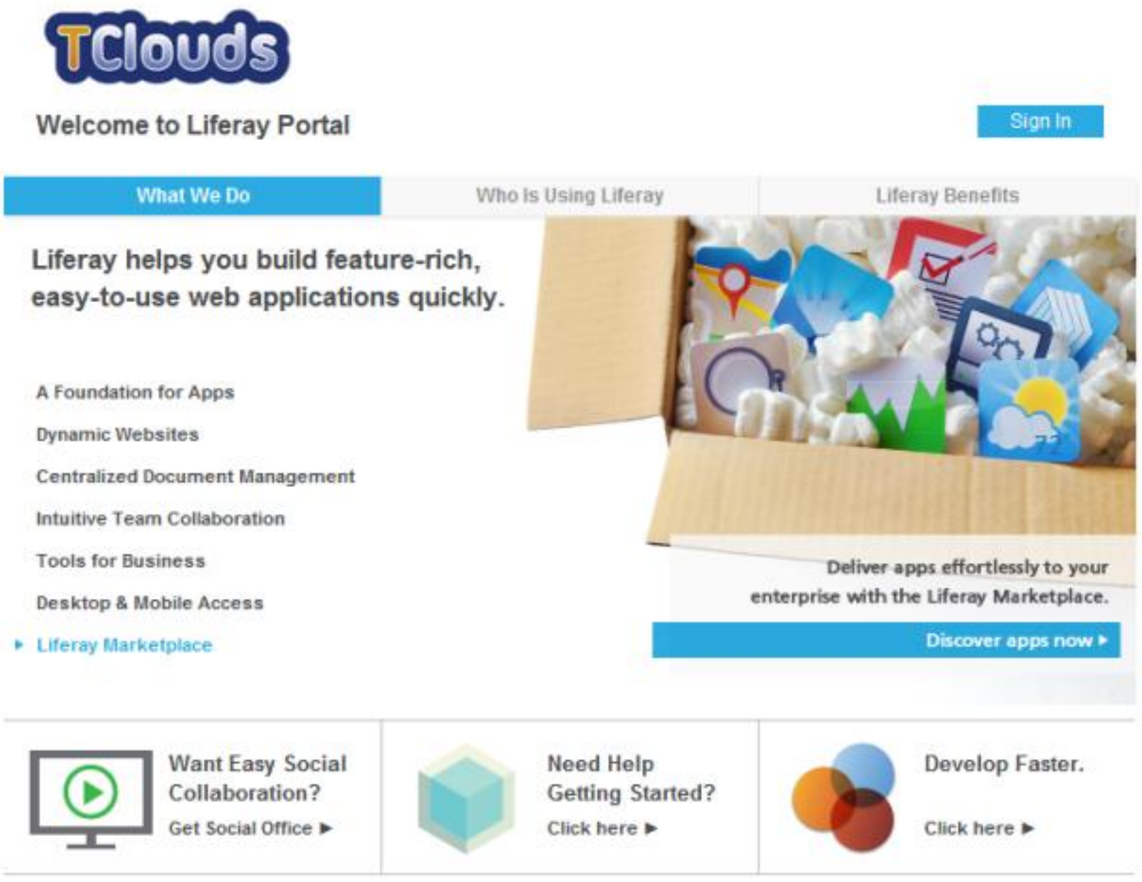

TClouds
Welcome to Liferay Portal
Sign In
What We Do
Who Is Using Liferay
Liferay Benefits
Liferay helps you build feature-rich, easy-to-use web applications quickly.
A Foundation for Apps
Dynamic Websites
Centralized Document Management
Intuitive Team Collaboration
Tools for Business
Desktop & Mobile Access
Liferay Marketplace
Deliver apps effortlessly to your enterprise with the Liferay Marketplace.
Discover apps now
Want Easy Social Collaboration?
Get Social Office
Need Help Getting Started?
Click here
Develop Faster.
Click here

Figure 24 : TPaaS demo homepage

4.1.2 Login / Sig-in

In order to Login into the platform you need a valid account. You can freely decide to create a new one by clicking on the “Sign in” link on top-right corner

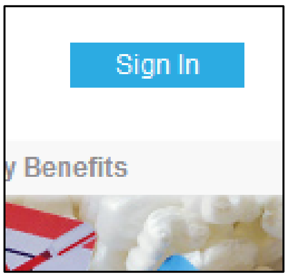


Figure 25 : Sign In link

And select the link “Create new account” on the bottom area of the Sign In page.

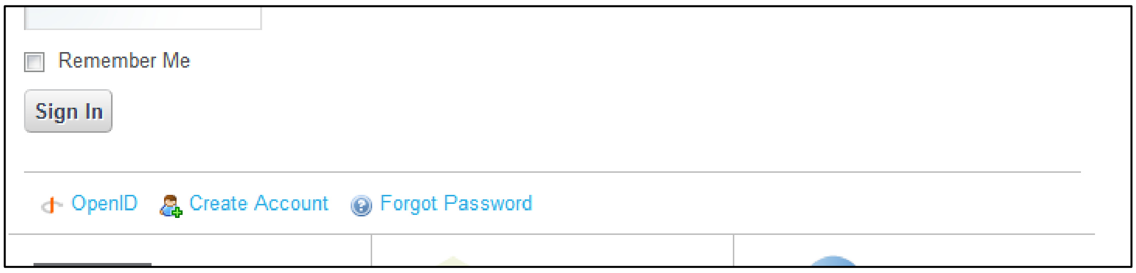


Figure 26 : Create new account link

Otherwise you can login with one of the pre-defined demo users already registered for you (refer to Table 4 for login data).

4.1.3 Adding new friends

As you login, a new sidebar appears on top of the page. If you click on “Go to” you can select “My Private Pages” to see the private welcome page of the user.

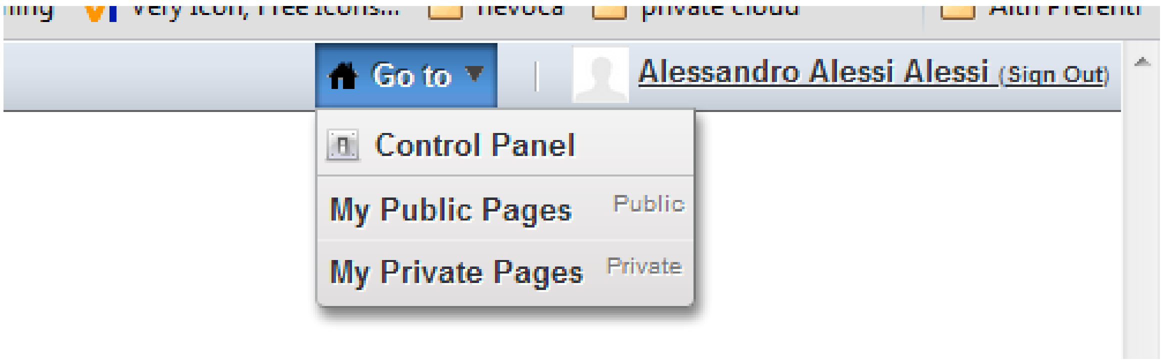
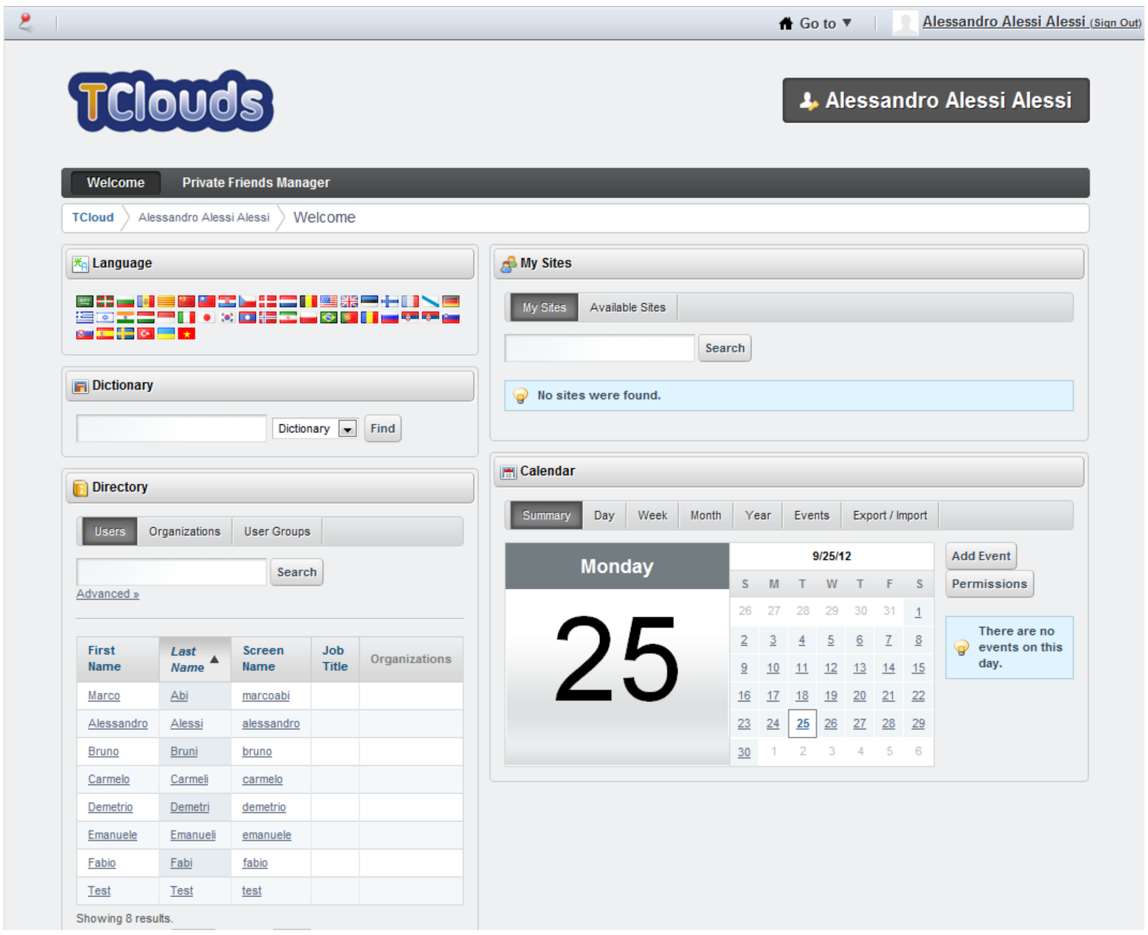


Figure 27 : TPaaS main menu bar



No sites were found.

There are no events on this day.

First Name	Last Name	Screen Name	Job Title	Organizations
Marco	Abi	marcoabi		
Alessandro	Alessi	alessandro		
Bruno	Bruni	bruno		
Carmelo	Carmeli	carmelo		
Demetrio	Demetri	demetrio		
Emanuele	Emanueli	emanuele		
Fabio	Fabi	fabio		
Test	Test	test		

Showing 8 results.

Figure 28 : User private welcome page

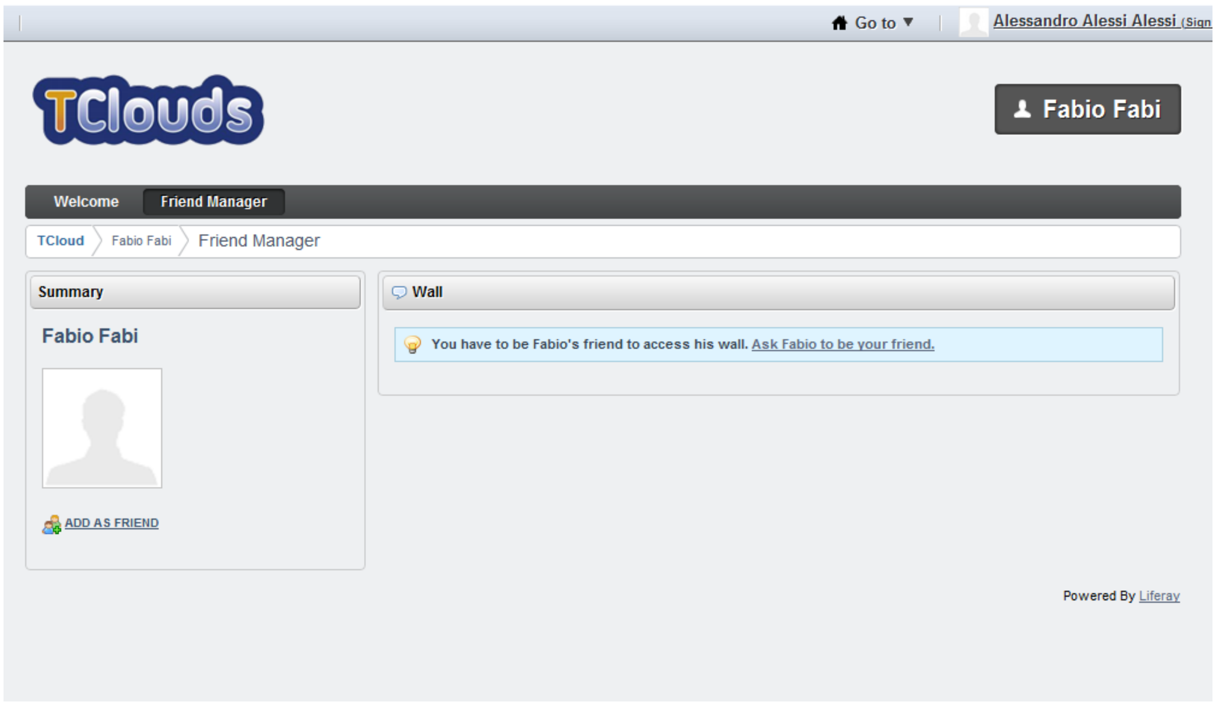
In the User private Welcome Page (see figure above) you can see all the users already registered into the platform under the “Directly” box on bottom left of the page.

You can see the public pages of the other users by hitting the following url:

http://tclouds-dev.eservices4life.org:8080/web/{user_screen_name}

where {user_screen_name} corresponds to the screen name appearing into the “directory” box on the welcome page.

Once you surf on a user public page it is possible to select to add his/her as a friend.



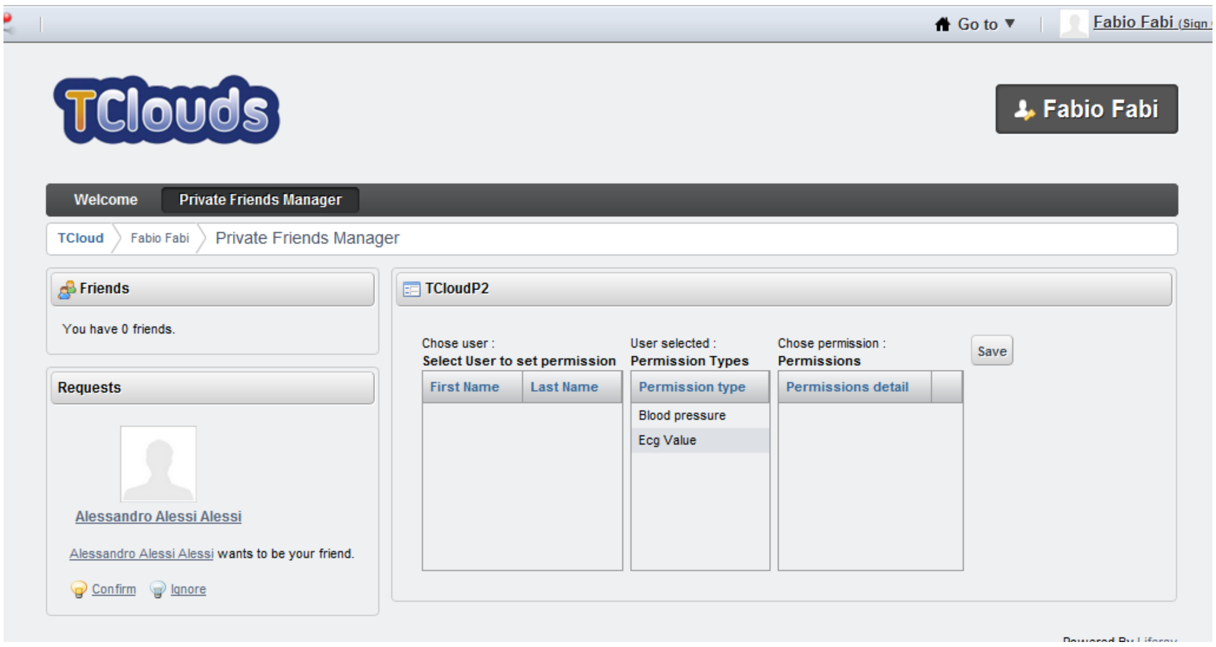
Fabio Fabi

You have to be Fabio's friend to access his wall. Ask Fabio to be your friend.

Figure 29 : Add as a friend of the user

4.1.4 Privacy Policy specification

Once you accept a new friend it is possible to define new privacy policies to allow/deny actions on your data.



You have 0 friends.

Alessandro Alessi Alessi

Alessandro Alessi Alessi wants to be your friend.

Chose user :
Select User to set permission

First Name	Last Name

User selected :
Permission Types

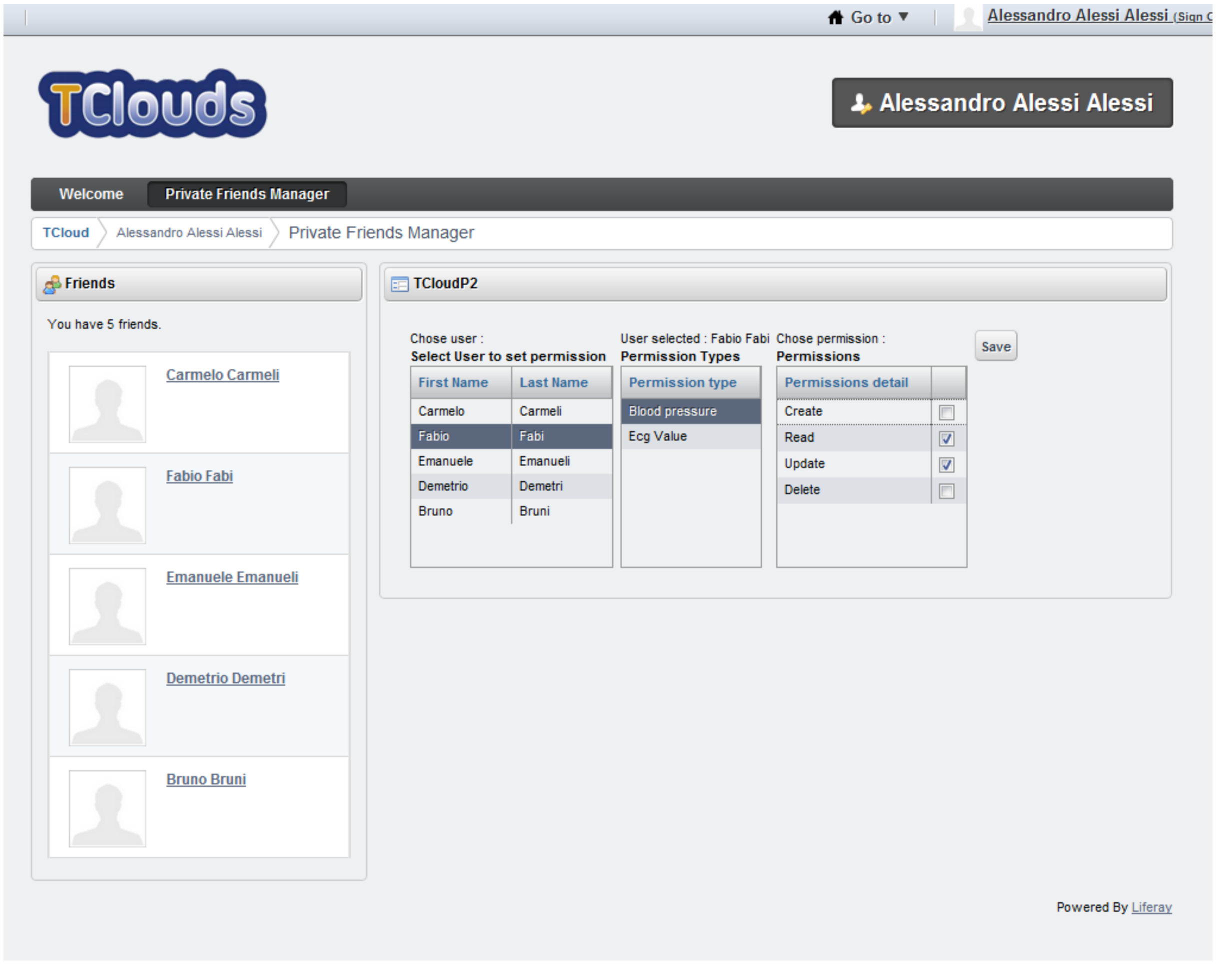
Permission type
Blood pressure
Ecg Value

Chose permission :
Permissions

Permissions detail

Figure 30 : Pending friendship request

Once you login, by going to “private friends Manager” you can select your friend, than the data type and the permissions on the data.


Go to ▼ | Alessandro Alessi Alessi (Sign O

Alessandro Alessi Alessi

Welcome | Private Friends Manager

TCloud › Alessandro Alessi Alessi › Private Friends Manager

Friends

You have 5 friends.

- Carmelo Carmeli
- Fabio Fabi
- Emanuele Emanueli
- Demetrio Demetri
- Bruno Bruni

TCloudP2

Chose user :
Select User to set permission

First Name	Last Name
Carmelo	Carmeli
Fabio	Fabi
Emanuele	Emanueli
Demetrio	Demetri
Bruno	Bruni

User selected : Fabio Fabi
Permission Types

Permission type
Blood pressure
Ecg Value

Chose permission :
Permissions

Permissions detail	
Create	☐
Read	☑
Update	☑
Delete	☐

Save

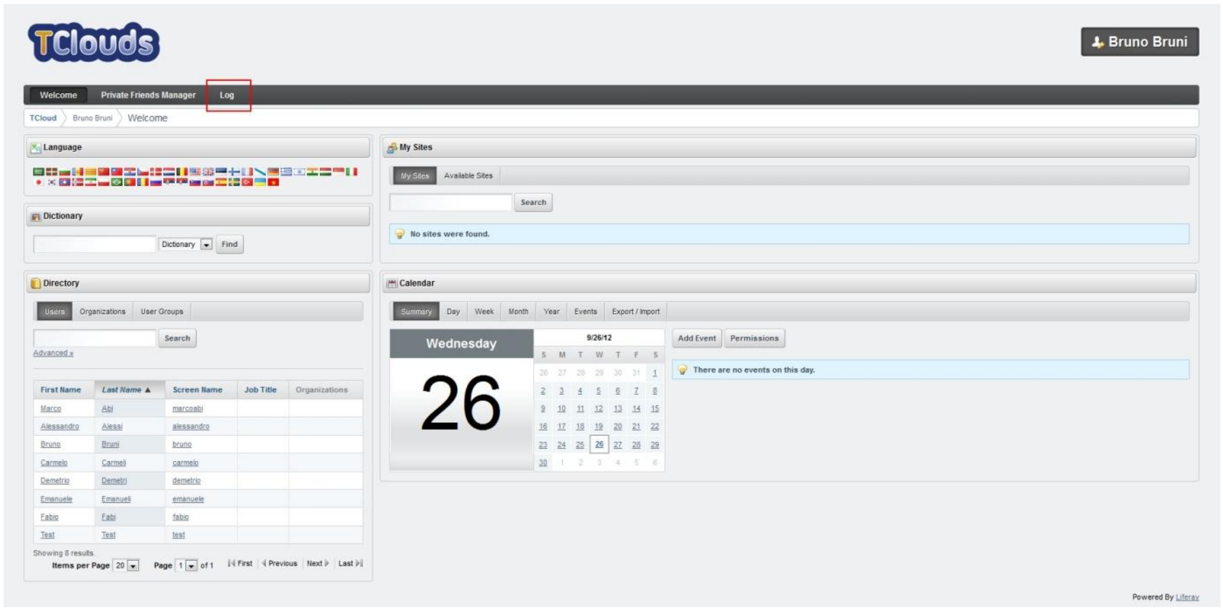
Powered By Liferay

Figure 31 : Privacy policy specification

4.1.5 Auditing

When apps registered into the platform access to users data (on behalf of some other user), the activity is constantly monitored and logged to allow the data owner to check who has accessed to his data and when.

You can access to the dashboard by clicking on the Log link on the control bar, as shown in the next picture:

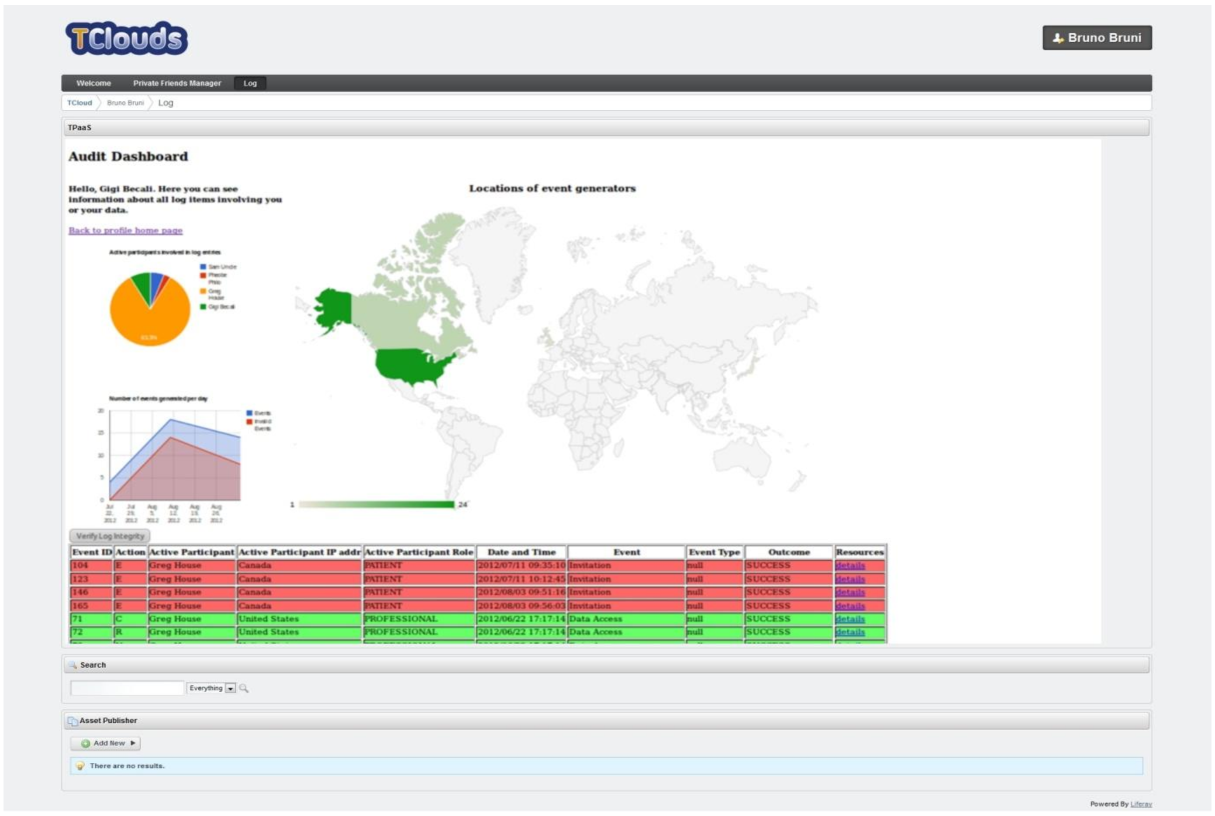


First Name	Last Name ▲	Screen Name	Job Title	Organizations
Marco	Abi	marcoabi		
Alessandro	Alessi	alessandro		
Bruno	Bruni	bruno		
Carmelo	Carmeli	carmelo		
Demetrio	Demetri	demetrio		
Emanuele	Emanueli	emanuele		
Fabio	Fabi	fabio		
Test	Test	test		

Showing 8 results.
Items per Page 20 Page 1 of 1 First Previous Next Last

Figure 32 : Log link from the welcome page

And the page with the log information will open, as shown in the next figure:



Audit Dashboard

Hello, Gigi Becali. Here you can see information about all log items involving you or your data.

Back to profile home page

Event ID	Action	Active Participant	Active Participant IP addr	Active Participant Role	Date and Time	Event	Event Type	Outcome	Resources
104	E	Greg House	Canada	PATIENT	2012/07/11 09:35:10	Invitation	null	SUCCESS	details
123	E	Greg House	Canada	PATIENT	2012/07/11 10:12:45	Invitation	null	SUCCESS	details
146	E	Greg House	Canada	PATIENT	2012/08/03 09:51:16	Invitation	null	SUCCESS	details
165	E	Greg House	Canada	PATIENT	2012/08/03 09:56:03	Invitation	null	SUCCESS	details
71	C	Greg House	United States	PROFESSIONAL	2012/06/22 17:17:14	Data Access	null	SUCCESS	details
72	R	Greg House	United States	PROFESSIONAL	2012/06/22 17:17:14	Data Access	null	SUCCESS	details

There are no results.

Figure 33 : Audit dashboard

4.2 Waiting room

The development of TPaaS is an ongoing process and it is still not complete at the time of writing.

The following activities need to be performed in order to benefit from the platform capabilities:

- Management of other users (developers, professionals, etc.)
- Management of apps from the UI
- Extensive use of DSL grammar in all aspects of the platform (now it is confined into the RESTful API)
- Integration of the main UI with the Auditing UI

4.3 Conclusion

This chapter illustrates the usages of the Healthcare TPaaS. To avoid complex installation by a user using binaries, this platform is deployed online. With a graphical user interface, it provides functionalities of login/sign-in, adding new friends, privacy policy specification, and auditing. In the third project year, there are improvements to be done, such as extending the user interface and the use of DSL.

Chapter 5

Conclusion

Chapter Author:

Mina Deng, Zheyi Rong (PHI)

In TClouds, Activity 3 specifies cloud applications, functionalities and requirements, defines application prototypes to be implemented on the cloud platform, and validates the application prototypes and the TClouds platform. In this document we describe the draft proof of concept for home Healthcare Trusted Platform as a Service (TPaaS), which is the use case discussed in work package 3.1.

The Healthcare TPaaS is a trusted platform that is deployed in the cloud in a safe and trustworthy fashion, allowing third party applications to be deployed and benefit from the TPaaS offered services. For implementation, we designed a service-oriented architecture, including three main services: UI service, Management DB service, Log and Audit service. Besides these services, a Domain Specific Language (DSL) was implemented to receive all the operation instructions that have to be performed either from the UI component or from the RESTful API component.

The modules in the Healthcare TPaaS provides a set of security features on board that are specified as follows.The RESTful APIs implement OAuth2.0 protocol that allows users to share their personal health records stored in TClouds without having to hand out their credentials to third party applications. Besides OAuth2.0, these APIs implement OpenID protocol to allow users log on to third party applications with their TPaaS accounts. The communications among all procedures could be encrypted by the module CP-ABE. The secure logging and auditing module logs every access from third party applications, and analyzes whether they have performed as promised. The databases used in the TPaaS comprise the Management Database, which stores all the side-data necessary to maintain and perform basic social functionalities of the platform, and the Data Database, which contains Personal Health Records of users.

We provided a list of security and privacy requirements to the underlying cloud environment. These requirements are to be satisfied in different layers of the cloud ecosystem, namely IssS, PaaS and SaaS. Current implementation of the draft proof of concept is deployed on a commodity cloud, which is OpenStack provided by SRX.

Furthermore, we showed that current market available commodity clouds such as OpenStack or Amazon cannot fully satisfy the aforementioned security and resilience requirements. Therefore, we are motivated for the TClouds security components, along with the Healthcare TPaaS on-board security features, to fulfill the requirements, and to ensure security, privacy, and resilience of the entire TClouds supported healthcare ecosystem. Such security solutions shall be provided by security components offered by TClouds Activities 2, among which are:

- replicate data across different replicas and tolerate byzantine or crash faults (BFT-SMART/DivDB)

- leverage many different clouds in order to increase robustness and security of a remote storage system solution (Cloud of clouds storage)
- log and track events generated at multiple cloud layers (Log Service)
- assess the integrity of physical hosts that are parts of the Cloud infrastructure (Remote Attestation Service).
- separate sensitive operations and secrets (Secure Key Management).

The distribution of the draft proof of concept of the Heathcare TPaaS is not provided in the form of binary files due to the complex installation. Instead, this platform is accessible online. With the graphical user interface of this platform, a user can create his profile, log in this platform, add his friends and audit the behaviors of third party applications.

In the third project year, the Healthcare TPaaS would be deployed on TClouds with the integration of A2, rather than on commodity clouds such as OpenStack. The logging/auditing service shall be as well deployed on TClouds. Furthermore, the user interface shall be improved, allowing a user to manage the third party applications. Finally, at least one third party application of medical use cases would be developed, which can access users data under their privacy control.

List of Abbreviations

Table 5 : List of Abbreviations

ACaaS	Access Control as a Service
API	Application Programming Interface
BPEL	Business Process Execution Language
CRUD	Create, Retrieve, Update, Delete
DAO	Data Access Object
CoC	Cloud of Clouds
CP-ABE	Ciphertext Policy Attribute-based Encryption
DBMS	Database Management System
DSL	Domain Specific Language
HSM	Hardware Security Module
IaaS	Infrastructure as a Service
JDBC	Java Database Connectivity
JPA	Java Persistence API
MVC	Model-View-Controller
PaaS	Platform as a Service
REST	Representational State Transfer
SaaS	Software as a Service
SBS	Secure Block Storage
SLA	Service-level Agreement
TPM	Trusted Platform Module
TVD	Trusted Virtual Domain
UI	User Interface
VM	Virtual Machine

Appendix A Legal boundaries for healthcare use case

Chapter Author: Ninja Marnau (ULD)

A.1 *Legal boundaries for the further product development*

These considerations are meant to extend the legal requirements stated in D3.1.1. Due to health data being highly sensitive personal data that is governed by Art. 8 Data Protection Directive 95/46/EC, the Member States are ultimately competent for legislation regarding the processing of health data. Therefore, the restrictions for a health care cloud application like this highly vary on national level. There are two key boundary areas which are described by showing exemplary national legislation of a number of member states.

A.1.1 Professional secrecy

The health care cloud application requires the patient and the medical professional to share and access information via the TPaaS platform. In order to enable this interaction, information is disclosed to the cloud provider. Such disclosure of medical information may be hindered by national laws of professional secrecy.

Medical professionals are bound by the pledge of professional secrecy. This pledge dates back to the Hippocratic Oath, which can be translated to: "What I may see or hear in the course of the treatment or even outside of the treatment in regard to the life of men, which on no account one must spread abroad, I will keep to myself holding such things shameful to be spoken about."[15]

The ethical principle of discretion regarding all information associated medical practice may include various different types of information. Professional secrecy does not cover only the immediate treatment but also all the information that the medical professional acquires during practicing, such as:

- Existence of doctor/patient relationship
- Reason for this relationship
- Patient's master data
- Visits
- Medical findings
- Diagnosis
- Anamnesis
- Clinical records
- Personal circumstances of the patient

[15] The Hippocratic Oath: Text, Translation and Interpretation by Ludwig Edelstein, Supplements to the Bulletin of the History of Medicine, no. 1 (Baltimore: Johns Hopkins University Press, 1943).

- Treatment
- Medication
- Insurance information
- Relationship with other medical professionals
- Etc

In all member countries that are subject to the investigation, the breach of professional secrecy is a matter of criminal law, except for the United Kingdom, which does not have a Criminal Code. See the wording of the criminal laws below, translated into English (whenever the translation is possible).

Austria	Strafgesetzbuch (Austrian Penal Code – abbreviated to StGB) Section 121 Paragraph 1 Verletzung von Berufsgeheimnissen Wer ein Geheimnis offenbart oder verwertet, das den Gesundheitszustand einer Person betrifft und das ihm bei berufsmäßiger Ausübung eines gesetzlich geregelten Gesundheitsberufes oder bei berufsmäßiger Beschäftigung mit Aufgaben der Verwaltung einer Krankenanstalt oder mit Aufgaben der Kranken-, der Unfall-, der Lebens- oder der Sozialversicherung ausschließlich kraft seines Berufes anvertraut worden oder zugänglich geworden ist und dessen Offenbarung oder Verwertung geeignet ist, ein berechtigtes Interesse der Person zu verletzen, die seine Tätigkeit in Anspruch genommen hat oder für die sie in Anspruch genommen worden ist, ist mit Freiheitsstrafe bis zu sechs Monaten oder mit Geldstrafe bis zu 360 Tagessätzen zu bestrafen.[16]
Estonia	Karistusseadustik (Estonian Penal Code) Sections 157 and 157a 157. Violation of obligation to maintain confidentiality of secrets which have become known in course of professional activities Disclosure of information obtained in the course of professional activities and relating to the health, private life or commercial activities of another person by a person who is required by law to maintain the confidentiality of such information is punishable by a pecuniary punishment. 157a Illegal disclosure of sensitive personal data Illegal disclosure of sensitive personal data, enabling access to such data or transfer of such data for personal gain or if significant damage is caused thereby to the rights or interests of another person that are protected by law shall be punished by a pecuniary punishment or up to one year of imprisonment.[17]
France	Code pénal (French Penal Code) LIVRE II : Des crimes et délits contre les personnes Paragraphe 1 : De l'atteinte au secret professionnel Article 226-13 La révélation d'une information à caractère secret par une personne qui en est dépositaire soit par état ou par profession, soit en raison d'une fonction ou d'une

[16] http://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10002296

[17] http://www.legislationline.org/download/action/download/id/1280/file/4d16963509db70c09d23e52cb8df.htm/preview

	mission temporaire, est punie d'un an d'emprisonnement et de 15000 euros d'amende.[18] Article 226-14 states the exception for medical professionals in case of moral conflicts or when public security is endangered.
Germany	Strafgesetzbuch (German Penal Code, abbreviated to StGB) Section 203 Paragraph 1 Violation of private secrets Whosoever unlawfully discloses a secret of another, in particular, a secret which belongs to the sphere of personal privacy or a business or trade secret, which was confided to or otherwise made known to him in his capacity as a 1. physician, dentist, veterinarian, pharmacist or member of another healthcare profession which requires state-regulated education for engaging in the profession or to use the professional title; 2. professional psychologist with a final scientific examination recognised by the State; … 6. member of a private health, accident or life insurance company or a private medical, tax consultant or attorney invoicing service, shall be liable to imprisonment not exceeding one year or a fine.[19]
Spain	Código Penal (Spanish Penal Code) Article 197 Paragraph 5 Igualmente, cuando los hechos descritos en los apartados anteriores afecten a datos de carácter personal que revelen la ideología, religión, creencias, salud, origen racial o vida sexual, o la víctima fuere un menor de edad o un incapaz, se impondrán las penas previstas en su mitad superior Article 199 Paragraph 2 El profesional que, con incumplimiento de su obligación de sigilo o reserva, divulgue los secretos de otra persona, será castigado con la pena de prisión de uno a cuatro años, multa de doce a veinticuatro meses e inhabilitación especial para dicha profesión por tiempo de dos a seis años.[20]

[18]http://www.legifrance.gouv.fr/affichCode.do;jsessionid=21A54A995580CC07BB1CE9244BEDB016.tpdjo12v_3?idSectionTA=LEGISCTA000006181756&cidTexte=LEGITEXT000006070719&dateTexte=20120917

[19] http://www.gesetze-im-internet.de/englisch_stgb/englisch_stgb.html#p1721

[20] http://www.boe.es/buscar/doc.php?coleccion=iberlex&id=1995/25444

What is considered subject to the medical professional–patient privilege differs among the member states.[21] Some include all medical professionals, some only physicians. The scope of protected information also varies; some states protect only knowledge acquired during the course of providing medical consultation or service, others extending the scope to “side”-information like e.g. the existence of a patient relationship and insurance information.

Consequently, what is considered a breach of professional secrecy also differs. Some states are more restrictive than others when it comes to information transfer from medical professionals. Outsourcing of information technology is basically information transfer.

The TPaaS health care application requires the involved medical professionals to register as professional users and use the platform to get access to the patient’s information. Therefore, dependent on the usage of the application the medical professional at least reveals the existence of the medical professional/patient-relationship to the cloud provider. According to the criminal laws of some member states, this can be considered a breach of professional secrecy.

Especially restrictive is the criminal law of Germany, which does forbid any sort of outsourcing of processing of medical data by medical professionals.[22] Exceptions are only possible if there is a legal basis for the transfer of the explicit consent of the patient.

The more information the medical professional reveals via the health care application, e.g. messages to the patient, the more national criminal laws would be breached. Therefore, for the lawful usage of the health care application in several member states it is necessary that the patient gives her consent by giving a detailed and informed written release from the pledge of secrecy, including recipient, data and purpose of the release.

To which extent patients suffering from depression will be mentally able to freely give their consent and give a release from the pledge of secrecy have to be assessed on a case by case basis.

A.1.2 Cross-border data transfer

The second key area of boundaries for Healthcare TPaaS concerns national restrictions for cross-border transfer of medical data.

This begins with differing national requirements for the registration of controllers and processors of personal data. Several member states demand that data controllers and processors to register in order to be allowed to process personal data. The requirements range from none to extensive authorization process.

	National registration requirements
France	➢ four different sets of formalities, ranging from simple notification through to authorization ➢ depending on the type of data processing involved

[21] Due to language restrictions the following considerations are based on English translations of national laws and legal documents and might therefore be incomplete or not interpreted correctly.

[22] https://www.datenschutzzentrum.de/material/themen/gesund/patdvia.htm

	➢ health data requires highest level of formalities
Germany	➢ no registration requirements
Italy	➢ registration is required for certain categories of data, including health and genetic data, credit data, location data, market research data and of customer profiling
Spain	➢ extensive registration requirements and establishment of a public personal data register
United Kingdom	➢ data controllers must register with the Information Commissioner's Office (ICO) to notify their intention to process personal data before beginning ➢ fees and an annual renewal requirement ➢ small number of exemptions

Member states may also have installed additional or separate requirements for registering for cross-border data transfer. All examined states do not require registration for transfer to another EU member state. The transfer outside of the EU is treated differently. All member states have installed additional requirements (though not all a registration or authorization) including the data subject's consent or standard contractual clauses. Please refer to D1.2.2 for more information on cross-border transfer.

	National registration requirements for cross-border transfer
France	➢ transfer within the EU does not require registration ➢ transfer to countries formally declared as 'adequate' by the EU requires notification ➢ transfer to all other countries requires authorisation
Germany	➢ no registration required ➢ transfer to a non-EU country possible if that country ensures an adequate level of protection. ➢ extensive list of exceptions, including consent and contractual arrangements
Italy	➢ no additional registration for European and non-EU transfers
Spain	➢ prior authorization from the Director of the Data Protection Agency, 'who may grant it only if adequate guarantees are obtained'
United Kingdom	➢ no additional registration for non-European transfers ➢ unusual in European legislation: data can be transferred based on an internal risk assessment by the organization

Data from the health care system and eHealth records (EHR) face stricter boundaries. Most states have regulations to prohibit EHR to be transferred outside of the national borders (Germany[23] and England[24], most probably Italy, Spain, Denmark, France). The

[23] http://www.aerzteblatt.de/archiv/31932

reason for this restriction is that health care system data and EHR are considered highly sensitive. A breach of this information might have a high impact on the concerned data subject. The member states want to protect their citizens from unpredictable (lawful) access of public and private foreign bodies.

Healthcare TPaaS does not necessarily include a complete EHR. Whether the information uploaded by the patient qualifies as EHR depends on the interaction with the medical professional. The protection of EHR does apply to health data collected and controlled by a medical professional for providing medical treatment.[25] Therefore, the patient herself is free to transfer her data based on her consent. A cross-border transfer of data uploaded by the medical professional would have to be treated differently. A release from the pledge of secrecy may be a solution. Although weighing the data subject's interest may lead to a restriction on cross-border transfer of the data uploaded by the medical professional. An out-of-the-EU/EEA transfer of EHR seems not feasible. The risks seem hardly comprehensible and understandable, which makes the appropriate information of the data subject hardly manageable. The Article 29 Working Party underlines that "considering the elevated risk to personal data in an EHR system in an environment without adequate protection […] any processing – especially the storage – of EHR data should take place within jurisdictions applying the EU Data Protection Directive or an adequate data protection legal framework."[26] This boundary is highly relevant for any future expansion of Healthcare TPaaS.

One possible approach would be to grant providers from outside of the EU/EEA only access to anonymized data.

[24] Phil Walker, Department of Health (England), Comments on the draft of WP131 of the Article 29 Working Party http://ec.europa.eu/justice/policies/privacy/docs/health_records/member_states/dept_health_uk_en.pdf

[25] Article 29 Working Party, Working Document on the processing of personal data relating to health in electronic health records (EHR), 00323/07/EN, WP 131, Adopted on February 15, 2007, page 4 defines EHR as: "A comprehensive medical record or similar documentation of the past and present physical and mental state of health of an individual in electronic form and providing for ready availability of these data for medical treatment and other closely related purposes. Therefore, the more comprehensive the Healthcare TPaaS health care application becomes, the closer it is to an EHR.

[26] Article 29 Working Party, Working Document on the processing of personal data relating to health in electronic health records (EHR), 00323/07/EN, WP 131, Adopted on February 15, 2007, page 19.

Appendix B Healthcare TPaaS RASD (Requirements Analysis Specification Document)

Chapter Author: Marco Abitabile, Marco Nalin (FSR), Mina Deng, Zheyi Rong, Sebastian Banescu, Ya Liu (PHI)

B.1 *Non-functional requirements*

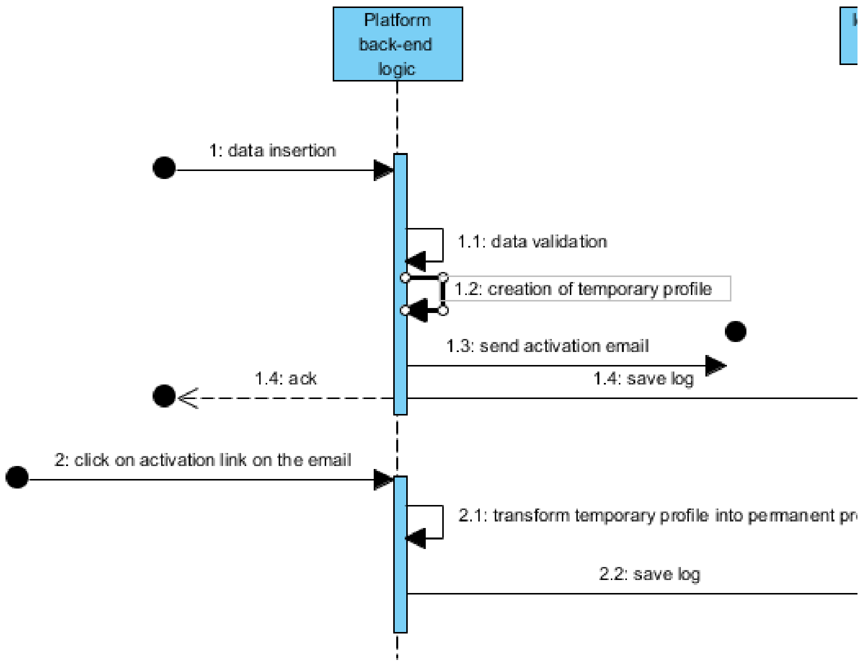
Req#:	100 – Perceived security is high
Description:	End users should have high confidence in the platform's ability to keep their data safe, or they won't use the platform.
Rationale:	End users need justification to fully trust the platform and save their data in.

B.2 *Functional and data requirements*

B.2.1 Functional requirements

Req#:	1 – User creation
Description:	The platform is able to recognize different types of users. Some of them can perform self-registration, others needs to be invited first. When a user gives her personal details (like name, surname, gender, birthday…) the system has to: - Log the user creation - Show to the user the terms of use and let the user to accept them (see req#23) - Create the related account - An user is identified by an unique email address Any new account will not be fully active (this will be treated as a temporary account) until the user confirms the activation email that the system has sent
Req. composed of:	#3.18, #3.4, #3.5, #3.2, #3.10

Sequence diagram:


Platform back-end logic
1: data insertion
1.1: data validation
1.2: creation of temporary profile
1.3: send activation email
1.4: ack
1.4: save log
2: click on activation link on the email
2.1: transform temporary profile into permanent pr
2.2: save log

Req#:	1.1 – generic end user creation
Description:	End users are data ownersand they need a proper login to the platform. They can be self-created (see req#3.5.6) or the creation starts from a previous email invite (see req#3.5.7). Once created they will have a proper account for Patients While creating the end user some personal info are requested like: - Name, surname, gender, age… Moreover: - She can store personal data - She has relationship with other user, professionals or applications - She can see data of the users she is related with (those with an incoming relationship) - She has only outgoing relationship with apps
Rationale:	End users are owner’s data and they need a proper login to the platform
Req. composed of:	#3.5.7, #3.5.6

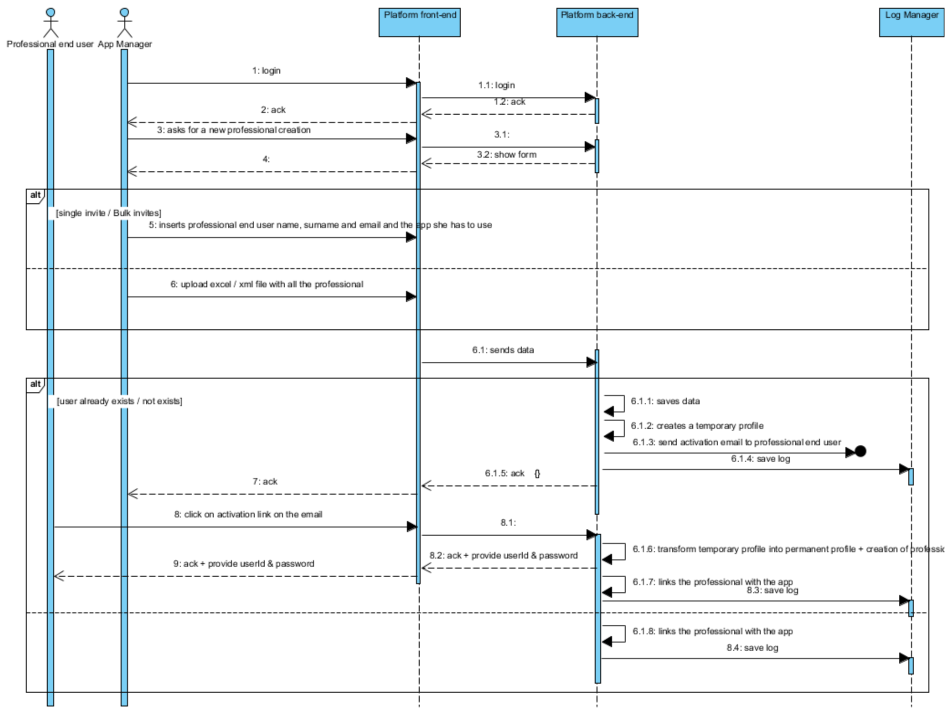
Req#:	1.3 – creation of developer account
Description:	Developer users are users recognized by the platform and have a specific account. - Account for developer can be created autonomously directly by the user
Req. composed of:	#3.18.8

Req#:	1.4 – app manager creation
Description:	App manager users are recognized by the platform as specific user with an app manager account. The meaning of the app manger is basically to, somehow, certificate the “professionality” of professional users. It is expected to have professional applications that have specific functionalities for professionals. NOTE: at this stage app managers can be created only if a previous invitation email has been sent to the potential app manager. (see req#3.4.1)
Rationale:	App managers can perform specific actions and have specific rights, for this reason the platform has to recognize them
Req. composed of:	#3.4.1

Req#:	1.2 – professional end user creation
Description:	Professional end users are proper users recognized by the platform and have a professional account. While creating the professional end user some personal info are asked like: - Name, surname, gender, age… Moreover: - She cannot store her personal data (in that case the individual needs another “common User” account); - She sees the personal data of the common user she is related to (always in respect with the privacy policies that the common user has chosen) - She has only incoming relationship - It might happen that the Professional is only subscribed to an App, and this will grant her rights to access patient’s data, even if no explicit relationship exists in the platform between the Professional and the patient. NOTE: at this stage app managers can be created only if a previous invitation email has been sent to the potential app manager. (see req#3.2.1)

Req#:	1.5 – Platform owner creation with TPaaS installation setup
Description:	The platform owner has a specific account as well. It can be created only at TPaaS installation phase and a specific account is created as well - There is only one owner per platform and - she cannot have relationships with other users - she has no rights to see users data

Req#:	1.1.2 – Generic end user creation by invite
Description:	Account for Generic end user can be created not only with self-creation but also via an invite by any other generic end user, professional end users and app managers. A temporary relationship between the two users will be made as well. In case the user already exists, than will be used the existent account and the relationship will be properly defined.

Req#:	1.2.1 – Professional end user creation by invite
Description:	Account for professional end user can be created via an invite by an app manager. In case the user already exists, than the existing account will be used and the relationship will be properly defined. If the professional end users accepts, she will have the rights to use the particular app that the app manager manages
Rationale:	
Req. composed of:	n.a.
Sequence diagram:	

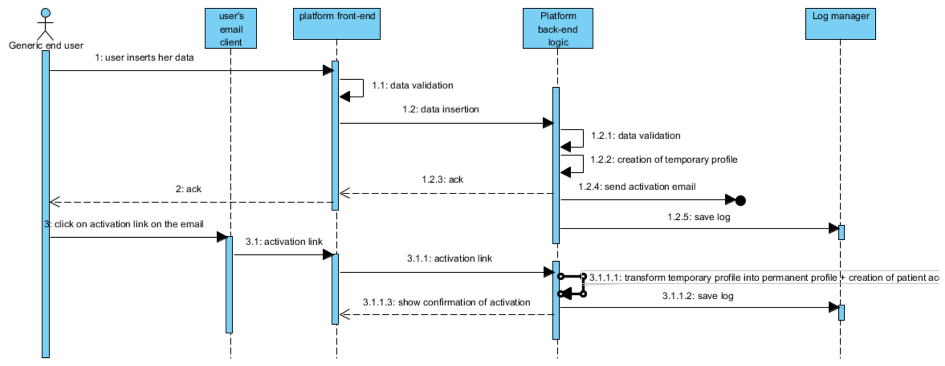
Professional end user
App Manager
Platform front-end
Platform back-end
Log Manager
1: login
1.1: login
1.2: ack
2: ack
3: asks for a new professional creation
3.1:
3.2: show form
4:
alt
[single invite / Bulk invites]
5: inserts professional end user name, surname and email and the app she has to use
6: upload excel / xml file with all the professional
6.1: sends data
alt
[user already exists / not exists]
6.1.1: saves data
6.1.2: creates a temporary profile
6.1.3: send activation email to professional end user
6.1.4: save log
6.1.5: ack {}
7: ack
8: click on activation link on the email
8.1:
6.1.6: transform temporary profile into permanent profile + creation of professi
8.2: ack + provide userId & password
9: ack + provide userId & password
6.1.7: links the professional with the app
8.3: save log
6.1.8: links the professional with the app
8.4: save log

Req#:	1.4.1 – App manager creation by invite
Description:	Account for app manager can be created via an invite by an app developer. In case the user already exists, than will be used the existent account.
Rationale:	The app manager is needed in the case of professional applications that require the creation of new professional users. The invite has the objective to allow an app manager to manage a certain app.

Req#:	1.3.1 – App developer self-creation
Description:	App developers can create an account by themselves. They have to insert their personal data and, if needed, company's personal data.

Req#:	1.1.1 – Generic end user self-creation
Description:	App developers can create an account by themselves. They have to insert their personal data autonomously and confirm the verification email sent.

Sequence diagram:



Req#: 4 – professional end user, app subscription

Description: Professional end users can have rights to use and login into a particular app. These rights are given from the App Manager that sends an invitation to the professional. (see #3.2.1)

Rationale: Professional end users can use particular apps that requires professional users to be logged on (generally they are professional apps for hospitals, clinics, etc…)

Req#: 9 – User content management

Description: The generic end user has the ability to manage its own data. In particular she can perform CRUD operation via the DATA API and manage her privacy policies and her relationships

Req#: 7.1 – Manage someone else's data

Description: A Generic end user or a professional end user have the ability to manage someone else's data under specific conditions expressed by the privacy policies of the data owner

Req#: 7.8 - relationship

Description: A relationship exists between generic end users, professional end users and apps. Relation can be only added (see #28.2.2) or removed (see #28.2.3) and it determines how a related user can manage owner's data

Req#: 7.8.3 – deletion of a relationship

Description: Taking an user (User1), deleting a relationship with User2 means that user2 is no longer able to see User1's data, while user1 is still able to see User2's data.

Req#: 7.8.1 – Update relationship

Description: An user that has a relationship with someone else, can modify the policies that define how the other user can access the owner's data.

The same can be done with a relationship with an application

Req#:	7.8.2 – Create relationship
Description:	An user can create a new relationship with another user (generic or professional) or an application

Req#:	7.8.2.1 – Creation of a generic end user

Req#:	7.8.2.2 – Creation of a professional end user

Req#:	7.8.2.3 – Creation of an application

Req#:	7.8.2.4 – Temporary relationship
Description:	If the end user has been created by invitation and it has not been confirmed, than the user account will be a temporary account and the relationship will be a temporary relationship.

B.2.2 Data requirements

The diagram below wants to illustrate how policies and relationships user-user and user-app are managed into data models.

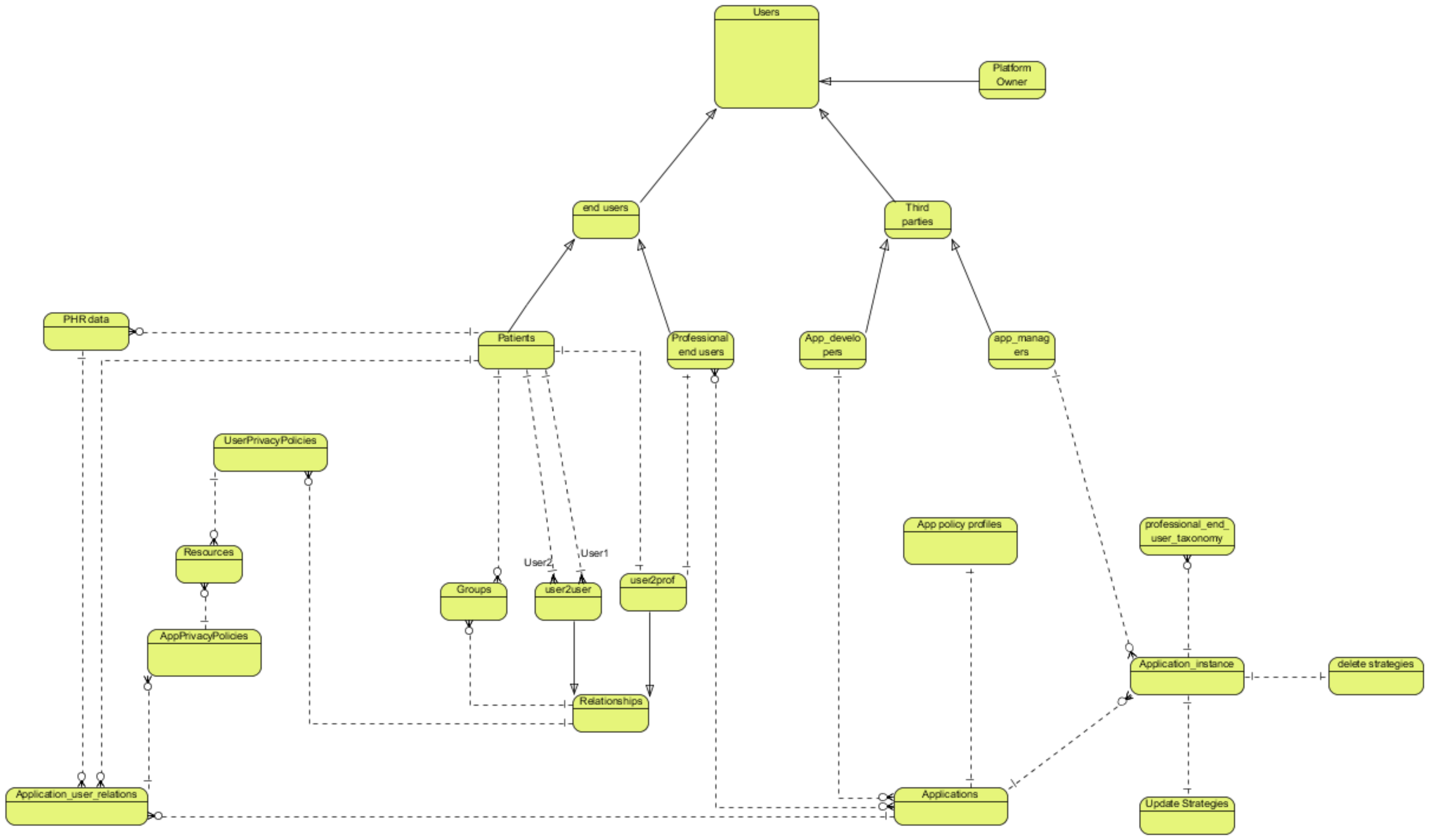


Figure 34 : ER diagram (draft) to highlight the policies management and user/app relationships

Users of the platform are divided into:

- Platform's owner
- Third party
    - App managers
    - App developer
- End users
    - Generic end users (patients)
    - Professional

Things to highlight:

Relationships among users are allowed only between end-users and more specifically among patient-patient and patient-professional. Please note the double connection between entity Patient and entity user2user, while the entity professional and user2prof has only one connection. This underlines that the relation of user1→user2 is different from that of user2→user1, and that between professional and patents can exist only the relation patient-professional (in which the professional can see patent's data but not vice-versa).

Resources correspond to a type of data/action that can be used under a specific policy. Example of data resource can be light data, sleep data while examples of actions can be: Blood analysis, psychical activity analysis, etc.

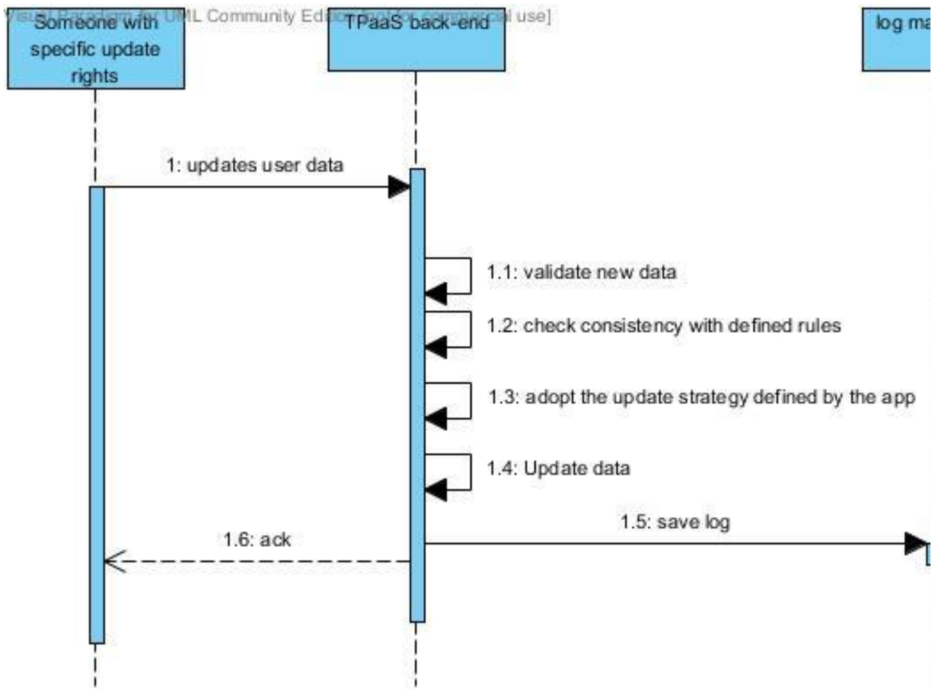
Application managers can use application. In that case applications are called instances that are applications that have a specific workspace.

Req#:	33 – Data ontology definition
Description:	Platform has to provide a Data Ontology. Owner user can insert it from restricted area.
Rationale:	Data ontology is needed to categorize and extract semantics from data. By using data ontology it is possible to force applications to respect it and store data according to the ontology defined.

Req#:	34– Request for Data ontology extension
Description:	Developers need the ability to ask for an extension of the data ontology in case their app has a specific type of data that cannot be mapped into the existing ontology. Request will be processed and dispatched by the platform's owner.
Rationale:	It is important to have the right data definition and categorization. Developers can contribute to create the most complete ontology, which will on one hand fit their needs and on the other will improve the platform completeness.

Req#:	11 – CRUD operation on data
Description:	Platform has to provide CRUD operations for user's data. Unless CRUD operations are made from the Generic end user's owner, all the requests have to pass through the privacy policy filter, removing and normalizing the data that has to be showed
Req. composed of: #46.33, #46.1, #46.32, #46.2	

Req#:	11.3 – Creation
Description:	The platform has to allow creation of new data. Creation of new data can involve some constraint checks see: Consistency checks and Consistency rules

Req#:	11.4 – Update
Description:	The platform has to allow update of existing data. Update of data may involve some consistency checks (see #60) and the Update strategy (see #45) defined by the app developer when registering the app to the platform (see #16.1)
Sequence diagram:	 Visual Paradigm for UML Community Edition [not for commercial use] Someone with specific update rights TPaaS back-end log ma 1: updates user data 1.1: validate new data 1.2: check consistency with defined rules 1.3: adopt the update strategy defined by the app 1.4: Update data 1.5: save log 1.6: ack

Req#:	10 – CRUD API
Description:	Applications have to connect to the platform by using apposite API.

Req#:	16 – CRUD operation on app
Description:	App developer can do CRUD operations on the app they develop. With this we intend the ability of the developer to register a new app (create, see #16.1), modify an app info (Update, see #16.4), delete an app form the platform(delete, see #16.2) and see app details (read, see #16.3).

Req. composed of:	#16.1, #16.2, #16.3, #16.4

Req.#:	16.1 - dev_CRUD_on_app: CREATE
Description:	- Registration of a new app (CREATE): developer can register a new app they have developed in order to allow the app to use the platform's API interfaces. At this stage the developer has to provide some info like: o The app name o The app version o The website in which the app runs o The callback URI for OAuth tokens o The minimum privacy policies that a future related user has to accept o The name of the App manager (if needed) o The list of data type used by the application and its mapping into the data ontology o The users' type that will use the app (may be: end users, professional end users, or both) -
Rationale:	Giving access to developers to register and manage their apps into the platform we allow to increase the "world" that spins around the platform and provide meaningful new services.

Req.#:	16.3 - dev_CRUD_on_app: READ
Description:	Watch app data (READ): developer can see the info about their app - See app name, version, callback url, app website, minimum privacy policies, number of related users

Req.#:	16.4 - dev_CRUD_on_app: UPDATE
Description:	- Modify an app info (UPDATE): developer can modify the info related to a specific app like: version, callback url, app website, minimum privacy policies
Rationale:	Can happen that the developer develops new app's release to add new functionalities and she may want to modify the minimum privacy policies, or the app is moved to a new address and so on…

Req.#:	16.2 - dev_CRUD_on_app: DELETE
Description:	Deleting an app (DELETE): a developer may want to delete her app. When deleting the app the system has to remove all the relationships that it has with all the users

B.3 *Performance requirements*

Req#:	6 – Load balancing and performances
Description:	The platform owner as the possibility to check and manage the performances of the entire platform. In particular he can check how the load balancer is working and how the application server (see #6.1) and DB server (see #6.2) are loaded
Rationale:	The platform has to be optimized to work in cloud environment and support different loads. Computational power and Storage has to be “elastic” enough to guarantee the performances of the platform
Req. composed of:	#6.1, #6.2

B.3.1 Capacity requirements

Req#:	6.2 – DB server mgmt.
Description:	The platform allows to the owner to add or reduce the number of physical machines required for db server in order to fit the actual demand of storage and maintain efficient the overall storage capacity efficient

B.3.2 Scalability or extensibility requirements

Req#:	32 – Scalability & extensibility
Description:	The platform has to scale server clusters to fit with the actual demand of computation / storage
Rationale:	The platform has to be optimized to work in cloud environment and support different loads. Computational power and Storage has to be “elastic” enough to guarantee the performances of the platform

Req#:	6.1 – Application server mgmt.
Description:	The platform allows to the owner to add or reduce the number of physical machines required for application server in order to fit the actual demand of computation and maintain efficient the overall computational power

B.4 *Longevity requirements*

B.4.1 Security requirements

Req#:	20 – Security APIs
Description:	The platform has to implements security API in order to perform identity mgmt. and access control. and OAuth-like credentials. The requirement is composed of - Consumer Authentication (see #20.1.2) - Access token verification (see #20.1.1) - Consumer registration (see #20.1.4) - Consumer un-registration(see #20.1.6) - Access token generation (see #20.1.3) - Access token issuing(see #20.1.5) - Access token refresh(see #20.1.7) - Access token revocation (see #20.1.8)
Req. composed of:	#20.1.1 - #20.1.8

Req#:	20.1.2 – Consumer Authentication
Description:	Platform needs to authenticate each consumer (apps) by verifying provided credential before a trusted communication is established
Rationale:	The authentication should be done before the consumers (apps) ask for RESTful services

Req#:	20.1.1 – Access token verification
Description:	The 3rd party application accesses protected resources by presenting the access token to Healthcare TPaaS. Our platform must validate the access token and ensure it has not expired and that its scope covers the requested resource.
Req. composed of:	#20.1.2

Req#:	20.1.4 Consumer registration
Description:	Consumer (web application, native application and etc.) needs to provide required information to Healthcare TPaaS. After verifying the information, Healthcare TPaaS will generate a consumer ID and consumer secrete (password, or public/private key pairs)
Rationale:	It is the basic for application integration, and consumer obtains its credentials from the registration which can be used to gain an access token.

Req#:	20.1.3 Access token generation
Description:	The platform should be able to generate secure access tokens. The implementation of token generator should be certified as being secure. Specifically, the token length should be enough against brute forcing attack. The output of the token generator should be random enough to resist against guessing attacks.
Rationale:	The platform must use a random number generator with good statistical properties such that an adversary will not be able to predict the correct value

	of an access token for a given application.

Req#:	20.1.5 Access token issuing
Description:	If the access token request is valid and authorized, the platform generates and issues an access token and optional refresh token. If the request is not authenticated or authorized, the platform returns error codes.
Rationale:	Platform must require authentication for any consumer that was issued credentials (or with other authentication requirements) before assigning an access token to the consumer.

Req#:	20.1.6 – Consumer un-registration/deletion
Description:	Our platform needs to allow consumer to unregister. A fully secure revocation of consumer credentials, access token and other related information should be operated by the platform.
Rationale:	It ensures that after the consumer discontinue the integrated service on the platform, consumer cannot access the protected resource anymore.
Req. composed of:	#20.1.8

Req#:	20.1.7 – Access token refresh
Description:	Our platform needs to refresh the access token under a valid refresh request from consumer.
Rationale:	Some access tokens might have expiration as additional attribute. In order to maintain the access, platform should refresh the access token regularly.

Req#:	20.1.8 – Access token revocation
Description:	Our platform should provide access token revocation once the user discontinue the access from certain consumer or the consumer unregister from the platform.

B.4.2 Access requirements

Req#:	5 – User Login
Description:	Once a user logins the platform will present only the functionalities she is allowed to do. Depending on the user, Login can be made directly to the platform (via the platform web page) or through an application (via the APIs) in this latter case the OpenID technique will be used to validate the access. Every login attempt (either failed or successes) has to be logged. User login is composed of: - Platform's owner login (see #5.5) - Developer login (see #5.3) - Generic end user login (see #5.1) - Professional end user login (see #5.2) - App manager login (see #5.4) If the login is successful the application will manage the login type eg: is an application has either professional or patient interface, it will manage to check whether the user has professional capabilities for that application or not. Login from the API will not allow to provide information if the user is an app

manager, app developer or platform owner.

Rationale: Horizontal privilege escalation must be prevented. For example:

- User/App A should not be able to access data belonging to user B (that has not granted access to A) by issuing a maliciously crafted request.

Vertical privilege escalation must be prevented. For example:

- An app manager should not be able to perform actions that can only be performed by the platform owner by issuing a maliciously crafted request.
- A developer cannot perform action that can only be performed by the app manager by issuing a maliciously crafted request.

Req. composed of: #5.1, #5.2, #5.3, #5.4, #5.5

Req#: 5.1 – Generic end user login

Description: Generic end user can login to the platform either via the web interface of the platform or from an application via the APIs. Following the abilities she can do depending on the access

- Login via API
  - She can perform CRUD operations on her data (see #63)
  - She can manage shared data of other friends (see #28)
- Platform’s web interface login
  - She can perform CRUD operations on her data (see #63)
  - She can see shared data of other friends (see #28)
  - She can perform audits on her data (see #50.58)
  - She can manage her relationships (with apps and users) & privacy policies (see #28.2)

Req#: 5.3 – Developer login

Description: Only platform’s web interface is allowed. When an app developers logins to the system, she can perform certain actions like:

- Invite an app manager to join for a specific app (see # 3.4.1)
- Perform CRUD operations on the app index like: creating a new app, modifying the privacy policies, deleting an app and so on… (see # 16)
- She can login only through the platform web access and she can’t perform any action via APIs

Req#: 5.5 – Owner login

Description: Only platform’s web interface is allowed. Platform's owner may be able to login and be recognized by the platform.

Once she is logged in, the owner can perform several activities like:

- Check the load balancing and check the performances (see # 51)
- Perform audits (see # 50.27)

Req#: 5.2 – Professional end user login

Description: Professional end users can login either via APIs or via the platform’s web interface. Professional users that login into the platform can:

	a- Manage the Applications to which they have a specific access to (see #45) b- Manage relationships with other users (recall the fact that professional users can have only incoming relationships) (see #28) c- Manage their own data and related data (but just basic visualization, maybe some charts or a lost with all the values – apps will process and aggregate meaningfully data) according with privacy policies that the related patient has set. (see #28) And accessing via API they can: 1. Manage related data (see #28)
Rationale:	Unlike generic users, Professional user can access their personal area within Healthcare TPaaS to access shared data from those generic users that have relationships with her.
Req. composed of:	5.5

Req#:	5.4 – App manager login
Description:	Only platform's web interface is allowed. Once app managers login will have this features enabled. App managers can login either from the platform's web interface or through the APIs interface (by using an app). Depending on their access point they have the ability to perform different actions as below: - Via API: o Ability to invite new professional user to use the application she is related with (see #3.2.1) - Via platform's web interface o Ability to invite new professional user to use the application she is related with (see #3.2.1) o Perform audits about what app users did on the data of who (see #55.59)

B.4.3 Integrity requirements

Req#:	15 – Consistency rules
Description:	The system has to keep track of the consistency rules that a certain data type may have. Rules are inserted by app developer once he registers the application in the platform (see #16.1). Platform has to use the rules when performing CRUD operations.(see #46.23, #46.2, #46.33)
Rationale:	In case of CRUD operations on user data, some consistency issues may arise. This may happen when an application save a certain data and another application will modify/delete it, without caring of data consistency that the first application has defined on that data.

Req#:	14– Consistency checks
Description:	Consistency checks use consistency rules active on a certain type of data and verify that new data is compliant. In case it is not, the action will be discarded.
Rationale:	Consistency checks are activated when someone performs CRUD operations on user content data. More specifically in case of Creation, Deletion and Update of existing data.

Req#:	14.2.3 – Data integrity in transit
Description:	Platform should establish a secure channel (TLS/SSL) where data integrity should be preserved against malicious attacks.
Rationale:	When applications and users need to receive or send data from/to the platform, it is likely that the data could be modified by malicious users. A secure channel at application level should be provided by specific schemes, e.g. SSL and TLS

B.4.4 Privacy requirements

Req#:	2 – Terms of Use
Description:	The product shall explicitly specify the purposes for which their personal data is being collected, no later than the time of collection. Subsequent use of this data will be limited to those purposes or other purposes that are subsequently authorized.
Rationale:	Article 10 of the EU Directive similarly requires the data collector to advise the individual of its identity, the purposes for which the data are intended, and any information such as the recipients of the data.

Req#:	19 – User Notification
Description:	The product shall notify users in case a related application changes its privacy policy requests
Rationale:	Article 6(1)(a) and (b) of EU Directive require that personal data be collected only for specified, explicit, and legitimate purposes and not further processed in a way incompatible with these purposes.

Req#:	7 – User Content Sharing
Description:	The product shall reveal private information only in compliance with the privacy policies specified by the user, except as subsequently permitted by the individual user or by the authority of law. These privacy policies are stored within the relationship info (see #7.8). Once policies are chosen they will allow a certain freedom to the related user / app to manage owner’s data (see #11)
Rationale:	Article 7 of the EU Directive provides that data collectors may only process personal information with the individual’s unambiguous consent, to perform a contractual or legal obligation, to protect the interests of the individual or the public, or are necessary for the legitimate interests of the data processor, but do not violate the rights of the individual.
Req. composed of:	#7.8, #11

Req#:	21 – Identity Protection (Anonymous Credential)
Description:	The product shall be able to provide anonymous credentials to a user, which she may subsequently use to authenticate to any third party application. The third party application will recognize her as a valid user, however it will not be able to infer her true identity, nor will the application be able to link between different anonymous credentials used by the same user.
Rationale:	This allows users to benefit from healthcare services offered by third party applications without needing to reveal their true identity to professional users.

Req#:	17 – Minimum privacy policy requirements (for app)
Description:	At creation stage of the app, the app developer has to define the minimum policy requirements that users must accept in order to use the app

Req#:	18 – Privacy policy specification
Description:	Every generic end-user has the ability to specify privacy policies for their relations with apps and other users.

Req#:	40 – Restrictions due to privacy policy
Description:	Privacy policies can allow or deny access and actions on generic end-user resources.
Rationale:	Privacy policies act as access control lists on the user's personal data

B.4.5 Logging requirements

Req#:	20.3.2 – Data Logging
Description:	The platform has the ability to detect and record all the activities performed on user data. The activities will be recorded in compliance with RFC 3881 and the users will be able to perform auditing activities on the recorded logs (see #20.2.3).
Req. composed of:	#11

Req#:	20.3.2.1 – Save Data Log
Description:	The platform will record the following information related to used data access in the audit log: - Date of action (ISO 8601 compliant) - Type of action (Create, Read, Update, Delete) - The outcome of the action (Success, {Minor, Serious, Major} Failure) - The function of the audit source that requires the data (app specific vocabulary) - The unique identifier of each user that the application is making the request on behalf of. - (optional) The IP address of the device used by the user - (optional) The role held by the user when the event was generated - A unique identifier of each audit source requesting the action (e.g. app ID) - (optional) The type of source requesting the action (e.g. Application, TPaaS User interface) - The unique identifier of each user data that is being accessed - (optional) The type of the object that corresponds to user data, i.e. 2=System object with respect to RFC 3881 defaults - (optional) The life-cycle of each user data (e.g. creation, access, aggregation, amendment, disclosure, etc.) - (optional) The sensitivity level of the data item (defined by Healthcare TPaaS according to user preferences) - The type of the user data (e.g. medical record, SSN, account, report,

	etc.)
Rationale:	This log format is compliant with the specification given in RFC 3881.
Req. composed of:	#11

Req#:	20.3.3 – App Logging
Description:	The platform has the ability to detect and record all application management and use activities (e.g. create/delete/update/use apps). The platform owner and app managers can perform audits on it (see #20.2.1, #20.2.2).
Req. composed of:	#16

Req#:	20.3.3.1 – Save App Log
Description:	The platform will record the following information related to application management and use in the audit log: - Date of action (ISO 8601 compliant) - Type of action (Create, Read, Update, Delete, Execute) - The outcome of the action (Success, {Minor, Serious, Major} Failure) - The function of the audit source that requires the data (Healthcare TPaaS specific vocabulary) - The unique identifier of each user that is performing the activity. - (optional) The IP address of the device used by the user - (optional) The role held by the user when the event was generated - A unique identifier of each audit source requesting the action (e.g. App management interface ID) - (optional) The type of source requesting the action (e.g. Application, TPaaS User interface)
Rationale:	This log format is compliant with the specification given in RFC 3881.
Req. composed of:	#16

Req#:	20.3.4 – Access Logging
Description:	The platform shall be able to detect and record any access to the platform, since user creation. App manages, platform owners, generic and users can benefit from it by performing audits. (see #20.2.1, #20.2.2, #20.2.3).
Req. composed of:	#1, #4, #5

Req#:	20.3.4.1 – Save Access Log
Description:	The platform will record the following information related to platform management and use in the audit log: - Date of action (ISO 8601 compliant) - Type of action (Create, Read, Update, Delete, Execute) - The outcome of the action (Success, {Minor, Serious, Major} Failure) - The function of the audit source that requires the data (Healthcare TPaaS specific vocabulary) - The unique identifier of each user that is performing the activity.

	- (optional) The IP address of the device used by the user - (optional) The role held by the user when the event was generated - A unique identifier of each audit source requesting the action (e.g. App management interface ID, User interface ID, etc.) - (optional) The type of source requesting the action (e.g. Application, TPaaS User interface)
Rationale:	This log format is compliant with the specification given in RFC 3881.
Req. composed of:	#1, #4, #5

Req#:	20.3.1 – Relationship & privacy policy logging
Description:	The platform shall be able to detect and record any relationship and privacy policy management operations performed by generic end-users. Generic end-users can benefit from it by performing audits. (see #20.2.3).
Req. composed of:	#7.8, #18

Req#:	20.3.1.1 – Save relationship & privacy log
Description:	The platform will record the following information related to relationship and privacy policy management in the audit log: - Date of action (ISO 8601 compliant) - Type of action (Create, Read, Update, Delete) - The outcome of the action (Success, {Minor, Serious, Major} Failure) - The function of the audit source that requires the data (app specific vocabulary) - The unique identifier of each user that the application is making the request on behalf of. - (optional) The IP address of the device used by the user - (optional) The role held by the user when the event was generated - A unique identifier of the audit source requesting the action (e.g. platform GUI ID) - (optional) The type of source requesting the action (e.g. TPaaS User interface) - The unique identifier of the subject (user or app) that the operation refers to - (optional) The type of the object that corresponds to user data, i.e. 4=Other with respect to RFC 3881 defaults - (optional) The life-cycle of each user data (e.g. creation, access, aggregation, amendment, disclosure, etc.) - The type of the subject (e.g. user identifier for users, URIs for apps)
Rationale:	This log format is compliant with the specification given in RFC 3881.
Req. composed of:	#7.8, #18

B.4.6 Audit requirements

Req#:	20.2.3 – Generic End- user Audits
Description:	Generic end users can perform audits on Access Log and Data Log. The type of audit that the end user can perform are: - Who has accessed her data using which application and which action they performed on her data? - Her own accesses to the platform (date & time and IP Address)
Rationale:	Article 12 of the EU Directive entitles an individual to obtain from the data controller: confirmation as to whether or not data relating to him are being processed and information at least as to the purposes of the processing, the categories of data concerned, and the recipients or categories of recipients to whom the data are disclosed.

Req#:	20.2.1 – Owner Audits
Description:	Owner can perform Audits at Infrastructure and platform level. - IaaS level: Audit has to be tied with audit services offered by the infrastructure level (TClouds A2) - PaaS level: Performed audit will give info about o Given a user: who has accessed her data and when? o Given an app: when did the app use the platform, accessing which data, on behalf of which user? o Accesses to the platform
Req. composed of:	#1, #4, #5, #7.8, #16

Req#:	20.2.2 – App manager audit
Description:	The platform has to provide audit features for App Managers. The audit has to provide the following info regarding the apps of each manager: - Who has logged on to the platform through an app? - Which actions, (related to which end-user) did the authenticated user perform?
Req. composed of:	#5.1, #5.2, #5.4, #16

B.4.7 Confidentiality requirements

Req#:	30 - Data encryption
Description:	Our platform provides data encryption as optional service for patients and also preserve the confidentiality of data in transit which includes: - Data at rest encryption for private use (see #30.1) - Data at rest encryption for data sharing (see # 30.2) - Data confidentiality in transit (see #30.3)
Rationale:	Besides access control, the resource should be protected by secure encryption against unauthorized access.
Req. composed of:	#30.1, #30.2, #30.3

Req#:	30.1– Data at rest encryption for private use
Description:	Our platform allows the data owner to encrypt his own PHR by his own key, only the owner and users who has access to the key can succeed in decryption. Encryption and decryption is operated securely by platform security component.
Rationale:	For highly confidential data, platform should provide basic encryption to maintain the data confidentiality. Symmetric encryption is applied to provide light weight service.

Req#:	30.2– Data at rest encryption for data sharing
Description:	Our platform provides asymmetric key management for data encryption with access control purpose. Data owner can specify access policy and encrypt the data with public key which is generated from the platform. Meanwhile, secrete keys based on the roles (attributes) in the policy are assigned to different users or applications. The data can only be decrypted with specific secrete keys corresponding to the policy. Platform provides: - Key setup - Secrete key generation - Policy verification - (optional) encryption - (optional) decryption
Rationale:	Different from access control, e.g. oAuth, data owner doesn't need to authorize each access but define a single access policy to protect the resource against unauthorized access. Encryption and decryption can be provided by either platform or clients.

Req#:	30.3– Data confidentiality in transit
Description:	Platform should establish a secure channel (TLS/SSL) where data should be protected securely during transition.
Rationale:	When applications and users need to receive or send data from/to the platform, it is likely that the channel is eavesdropped by malicious users. A secure and encrypted channel at application level should be provided by specific schemes, e.g. SSL and TLS

B.4.8 Password Management Requirements

Req#:	31 – Password management
Description:	Our platform provides user registration, during which user (patient, app manager, developer, platform admin, etc.) is required to set a valid username and password as login credentials. The password needs to be secure enough against attacks, including brute forcing attacks: - Password format requirement (see #31.1) - Password renewal/expiration requirement (see #31.2)
Req. composed of:	#31.1, #31.2

Req#:	31.1 – Password format
Description:	All users' password should follow the requirement below: - The minimum length of password should be 10 characters. - Passwords shall not contain any proper noun or the name of any

	person, pet, child, or fictional character. Passwords shall not contain any employee serial number, Social Security number, birth date, phone number, or any information that could be readily guessed about the creator of the password. - The password should include at least 1 non-alphabet character, such as "!, @ ,#" - The password should contain a combination of alphabetic and numeric characters. - The password should contain a nonnumeric in the first and last position.

Req#:	31.2 – Password renewal/expiration
Description:	Password for the platform user needs to be renewed according to the requirement: - Password for platform admin should be renewed per month - Password for user except for platform admin should be renewed every 6 months. - The new password should contain no more than three identical consecutive characters in any position from the previous password.
Rationale:	This requirement limits the window of opportunity for password guessing attackers
Req. composed of:	

B.4.9 Key management Requirements

Req#:	35 –Key management
Description:	Cryptographic keys including symmetric keys and asymmetric keys should follow the following requirements: - Key generation requirement (see #32.1) - Key store requirement (see #32.2)
Rationale:	
Req. composed of:	#32.1, #32.2

Req#:	35.1 –Key generation
Description:	Platform should have a separate key generation module to generate key for AES encryption. The generation function should be operated in a secure, trusted domain where confidentiality and integrity of the key should be maintained. The minimum key Length for AES is 256 bits.
Rationale:	Attackers should not be able to guess the value of the key. It should be computationally un-feasible to perform brute force attacks or other key cracking attacks (e.g. Pollard Rho attack) on the cryptographic keys.

Req#:	35.2 –Key store
Description:	Key for all data encryption should be stored in a separate database in ciphertext (master key is required for encryption). Access to the database and master key should be controlled strictly by privacy policy. The master key should be renewed per month
Rationale:	No attacker should be able to get access to the secret keys

B.5 *Legal requirements*

These requirements are not complete. In Appendix A there is the complete legal analysis with further legal requirements to satisfy.

B.5.1 Compliance requirements

Req#:	13- data deletion Strategy
Description:	Deletion of data has to respect legal aspects relative to the country's law in which the data is stored. - Every country has its own law about data deletion and protection - Despite national law, every company has to follow certain law about privacy policy of its own customers and some companies (like hospitals) has to preserve clinical data for a certain period of time (Law analysis will quantify this dimensions). Deletion: below a list of the type of deletion allowed by the platform, the type of deletion can be chosen by the app developer at registration phase. - Disabled: deletion is not allowed - Deferred: the data is flagged as deleted (thus not available) and then removed permanently from DB after a certain period of time - Immediate: the actual data will be deleted immediately from DB - Fake deletion: deleted data will be flagged as deleted Deletion can be only of the actual data or also of backed-up data.
Rationale:	Deletion of data is a key aspect in order to preserve data's privacy. Due to the fact that the platform can be in any EU country (the prototype will be in Sirrix (Germany) and HSR (Italy)) and the law about privacy policy may vary from time to time, It is necessary to have a feature that allow the Platform owner to insert rules about how the platform has to manage Delete operation in order to be compliant with either national law and data storing of health data for professional users like hospitals.

Req#:	12 – data update Strategry
Description:	Update of data has to respect legal aspects relative to the country's law in which the data is stored. - Every country has its own law about data update and protection - Despite national law, every company has to follow certain law about privacy policy of its own customers Update techniques are different and they can be summarized in the list below: - Effective: old value is removed permanently and the new one takes its place - Fake upload: the old value is flagged as deleted and it is replaced with the new one - Delayed: old value is first flagged as deleted than, after a certain period of time is permanently deleted. Then new value will take the place of the old value.
Rationale:	Update of data is a key aspect in order to preserve data's privacy. Due to the fact that the platform can be in any EU country (the prototype will be in Sirrix (Germany) and HSR (Italy)) and the law about privacy policy may vary from time to time, It is necessary to have a feature that allow the Platform owner to insert rules about how the platform has to manage Update operation in order to be compliant with either national law and data storing of health data for professional users like hospitals.

Bibliography

Bethencourt, J. a. (2007). Ciphertext-Policy Attribute-Based Encryption. *Proceedings of the 2007 IEEE Symposium on Security and Privacy* (pp. 321--334). IEEE Computer Society.

Bloch, J. (2008). *Effective Java, Second Ed.*

Evans, E. (2003). *Domain Driven Desing.* Addison Wesley.

Fowler, M. (2010). *Domain Specific Language.* Addison-Wesley Professional.

Gang of four (E. Gamma, R. H. (n.d.). *Design Patterns.*

Google Inc. (n.d.). *Google Chart Tools*. Retrieved from Google Developers: https://developers.google.com/chart/

Goyal, V. a. (2006). Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 89--98). Alexandria, Virginia, USA: ACM.

Ibraimi, L. a. (2009). Efficient and Provable Secure Ciphertext-Policy Attribute-Based Encryption Schemes. In *Information Security Practice and Experience* (pp. 1-12). Springer Berlin / Heidelberg.

Ibraimi, L. a. (2009). Mediated Ciphertext-Policy Attribute-Based Encryption and Its Application. In H. Y. Youm, *Information Security Applications* (pp. 309--323). Springer-Verlag.

IHE. (n.d.). *Audit Trail and Node Authentication*. Retrieved from IHE Wiki: http://wiki.ihe.net/index.php?title=Audit_Trail_and_Node_Authentication

Integration of the Healthcare Enterprise. (n.d.). *IHE Home*. Retrieved from http://www.ihe.net/

Ma, D., & Tsudik, G. (2009). A new approach to secure. *5*(1).

Marshall, G. (2004). Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications.

Mather, T., Kumaraswamy, S., & Latif, S. (2009). *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance (Theory in Practice).* O'Reilly.

Sahai, A. a. (2005). Fuzzy Identity-Based Encryption. In R. Cramer, *Advances in Cryptology – EUROCRYPT 2005* (pp. 557-557). Springer Berlin / Heidelberg.

Sandhu, R. (1994). Access control: principle and practice . *Communications Magazine, IEEE, 32*(9).

Schneier, B., & Kelsey, J. (1999). Secure audit logs to support computer forensics. *2*(2).

Smiraglia, P. (n.d.). *Libsklog*. Retrieved from Github: https://github.com/psmiraglia/Libsklog

Smith, B. (2007, June 29). *A Quick Guide to GPLv3*. (GNU) Retrieved from GNU Operating System: http://www.gnu.org/licenses/quick-guide-gplv3.html

Waters, B. (2011). Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. *Proceedings of the 14th international conference* *on Practice and theory in public key cryptography conference on Public key cryptography* (pp. 53--70). Taormina, Italy: Springer-Verlag.

Webnet77. (n.d.). *FREE IP to Country Database IPV4*. Retrieved from Software77: http://software77.net/geo-ip/

Yu, S. a. (2010). Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. *2010 Proceedings IEEE INFOCOM*, 1--9.