D3.1.2

Application API and first specification on application side trust protocols

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
Programme:	FP7 IP

Deliverable type:	Report
Deliverable reference number:	ICT-257243 / D3.1.2/ 1.0
Activity and Work package contributing to the deliverable:	Activity 3 / WP 3.1
Due date:	March 2012 – M18
Actual submission date:	2nd April, 2012

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

Abstract:	This deliverable introduces a concept as Healthcare Trustworthy Platform as a Service (Health T-PaaS), by defining and describing its use cases, platform architecture and security components deployed on top of it. The objective of Health T-PaaS is to provide a secure and resilient cloud environment for healthcare service enhancement as well as developer oriented functionalities for 3rd part integration.
Keywords:	Healthcare applications and platform, Trustworthy PaaS, TClouds infrastructure, A2 and A3 integration

Editor

Mina Deng (PHI)

Contributors

Marco Nalin (HSR)

Marco Abitabile (HSR)

Ya Liu (PHI)

Sebastian Banescu (PHI)

Paolo Smiraglia (POL)

Davide Vernizzi (POL)

Klaus Stengel (FAU)

Johannes Behl (FAU)

Stefan Nürnberger (TUDA)

Norbert Schirmer (SRX)

Imad M. Abbadi (OXFD)

Sören Bleikertz (IBM)

Disclaimer

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

Executive Summary

This deliverable presents the work carried out within WP3.1 from M12—M18, and mainly addresses Task 3.1.2 “Definition of application architecture: components, APIs and data structures” and Task 3.1.5 “Integration between Activities A2 and A3”.

The work presented in this deliverable is based the result from WP3.1 in year 1, and it extends the A3 activities from application layer oriented healthcare applications (SaaS) to platform level (PaaS) solutions. Moreover, this work tries to function as a bridge to link results from A2 of trustworthy cloud infrastructure solutions and A3 healthcare use case applications towards end users.

This deliverable introduces a concept as Healthcare Trustworthy Platform as a Service (Health T-PaaS), by defining and describing its use cases, platform architecture and security components deployed on top of it. The objective of Health T-PaaS is to provide a secure and resilient cloud environment for healthcare service enhancement as well as developer oriented functionalities for 3rd part integration.

We define the roles of each involved actor in the proposed Health T-PaaS, and illustrate different functional packages which include user management, security and privacy management, application management, auditing, monitoring and benchmarking.

Then we give a brief overview on existing healthcare PaaS offerings in the market, and analyze selected vendors such as Microsoft HealthVault and TPSC Cloud, to analyze their utility, security, privacy and development aspects.

Based on the exploration of existing healthcare PaaS providers, we introduce the architecture and components of this multilevel healthcare platform, Health T-PaaS, aiming to offer novel services for both users and developers. The business model, prototype and benchmark applications of the platform are illustrated.

Next, a high level API document is provided, focusing on offering RESTful level APIs for developers to implement Health T-PaaS compatible applications, including basic function for authentication, authorization and resource access. Application side trust protocol is described in details with extensive API support.

Last but not least, we discuss the integration between A2 and A3 by proposing a list of A2 security components to be embedded within the Health T-PaaS. These security components include Log Service, Tailored Cloud Services (Tailored memcached), Resilient BPEL (RPEL), Secure Block Storage (SBS), Trusted Virtual Domain (TVD), Resilient Object Storage, Access Control as a Service (ACaaS), and Security Assurance tool for Virtual Environment (SAVE).

Contents

List of Figures

Chapter 1

Introduction

Chapter Authors:

Mina Deng, Ya Liu (PHI)

The main objective of the work package 3.1 “Cloud Applications Data Structures for Home Healthcare Benchmark Scenario” is to define the cloud architecture and specification from the application side of view, i.e., provide technical requirements for cloud computing in the healthcare sector, especially in the area of home healthcare services and leverage this application into an architecture, API, and protocols between application components and clients. The cloud supported home healthcare application architecture will be closely linked and integrated with WP2.1, WP2.2, WP2.3 and WP2.4.

1.1 Outline of the Work Done from M12-M18

This deliverable presents the work carried out for Task 3.1.2 “Definition of application architecture: components, APIs and data structures”, and Task 3.1.5 “Integration between Activities A2 and A3”. In the first year, WP3.1 mainly focuses on application layer (SaaS), defining a number use cases including patient home monitoring and a preliminary patient healthcare record portal. In the second year, we aim to bridge the gap between the results from WP3.1 from year 1 and those from A2, by extending our contribution towards the platform layering by introducing a Healthcare Trustworthy Platform as a Service (Health T-PaaS), where a number of applications can be deployed (i.e. the healthcare applications developed in year 1). We further try to address the integration between A3 and A2 by deploying the Health T-PaaS on top of the TClouds infrastructure which is proposed from A2.

1.2 Structure of this deliverable

This deliverable aims at introducing Healthcare Trustworthy Platform as a Service (Health T-PaaS) by defining and describing its use cases, platform architecture and security components deployed on top of it. The objective of Health T-PaaS is to provide a secure and resilient cloud environment for healthcare service enhancement as well as developer oriented functionalities for 3rd part integration. The document is structured as follows:

In chapter 2 we provide a general overview of Health T-PaaS use cases by defining the role of each involved actor and illustrating different functional packages which includes user management, security and privacy management, application management, auditing, monitoring and benchmarking.

An analysis of existing healthcare PaaS offerings in the market is offered in Chapter 3. Specifically, a case study is conducted on Microsoft HealthVault and TPSC Cloud in utility, security, privacy and development aspects.

In Chapter 4, based on the exploration of existing healthcare PaaS providers, we introduce a multilevel healthcare platform, Health T-PaaS, aiming to offer novel services for both users and developers. The business model, prototype and benchmark applications of the platform are illustrated as well.

In Chapter 5, a high level API document is provided, focusing on offering RESTful level APIs for developers to implement Health T-PaaS compatible applications, including basic function for authentication, authorization and resource access. Application side trust protocol is described in details with extensive API support.

Chapter 6 specifies each security components embedded within the Health T-PaaS in details, including security based infrastructure (e.g. secure block storage) and security based services (e.g. access control as a service).

Chapter 2

Use cases of a health trustworthy PaaS

Chapter Authors:

Marco Nalin (HSR), Marco Abitabile (HSR)

2.1 Relevant facts and assumptions

2.1.1 Facts

Factors that have an effect on the product, but are not mandated requirements constraints. They could be business rules, organizational systems, or any other activities that have an effect on this product. Facts are things you want the reader of the specification to know.

Motivation: Relevant facts provide background information to the specification readers, and might contribute to requirements. They will have an effect on the eventual design of the product.

- T-PaaS can be physically installed in any EU country (in our case we are going to deploy the platform on Sirrix Infrastructure based in Germany.
- Common Users (those who decide to store their data on) of the Platform, thus users of the applications
- Applications:
    - Can have local storage or remote storage, in this latter case we mean the use of T-PaaS Platform to store data
    - Can be desktop application, web application or mobile application

2.1.2 Assumptions

A list of the assumptions is proposed. These assumptions might be about the intended operational environment, but can be about anything that has an effect on the product. As part of managing expectations, assumptions also contain statements about what the product will not do.

Motivation: To make people declare the assumptions that they are making. Also, to make everyone on the project aware of assumptions that has already been made.

Will:
- T-PaaS platform will achieve its legal requirements at Infrastructure level by relying on A2 findings, research and products
- T-PaaS platform will achieve its legal requirements at Platform and Software level by implementing the requirements that the legal analysis will perform, based on this document.
- It may be possible to build a sample application to be connected to the platform to show some of its functionalities (Marvin 2.0 – Y1 HSR results - can be a good candidate)

- Since access to other user data is governed by strict privacy rules and relationships among users, everyone that wants to access to someone else data needs a proper account in the Platform

Will not:

- Be a complete and exhaustive business oriented system (so it may lack of some business/economic aspects)

2.2 Use Case Specification

2.2.1 Definitions

Definition of relation: a connection of type user-user, user-prof or user-app that involves sharing personal data. These relationships are governed by privacy policies.

Relations are Google+ like. The relation depicted in figure1a means that User1 is related with User2 and User2 is able to access to User1 data (according to certain privacy policies that User1 has set) but not vice versa.

In order to have both users to see each other data User2 should instantiate a relation with User1, like in figure1b. Figure1c describes relations allowed that involve privacy policies.

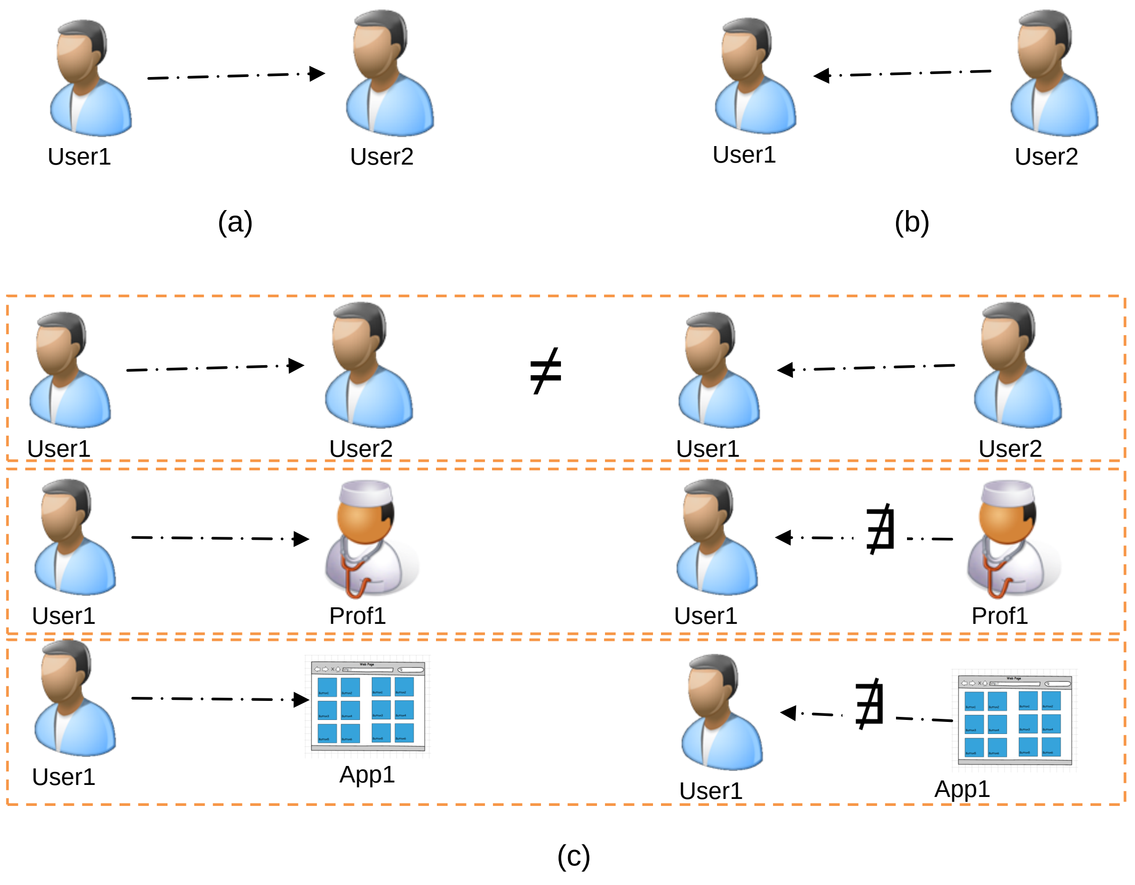
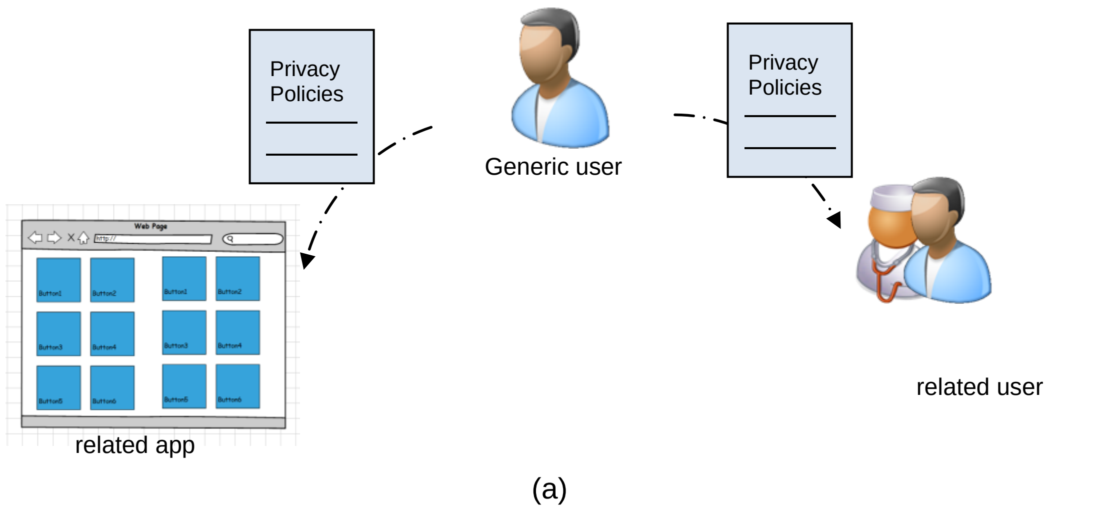


Figure 1 - Relationship between User1->User2 (a). Relationship between User2->User1 (b). Allowed relationships (c)

- A generic user can have a relation with (see figure2a):
  - any other generic user or prof-user
  - any application

- A related user can access to user's data through a related App (see figure2b):



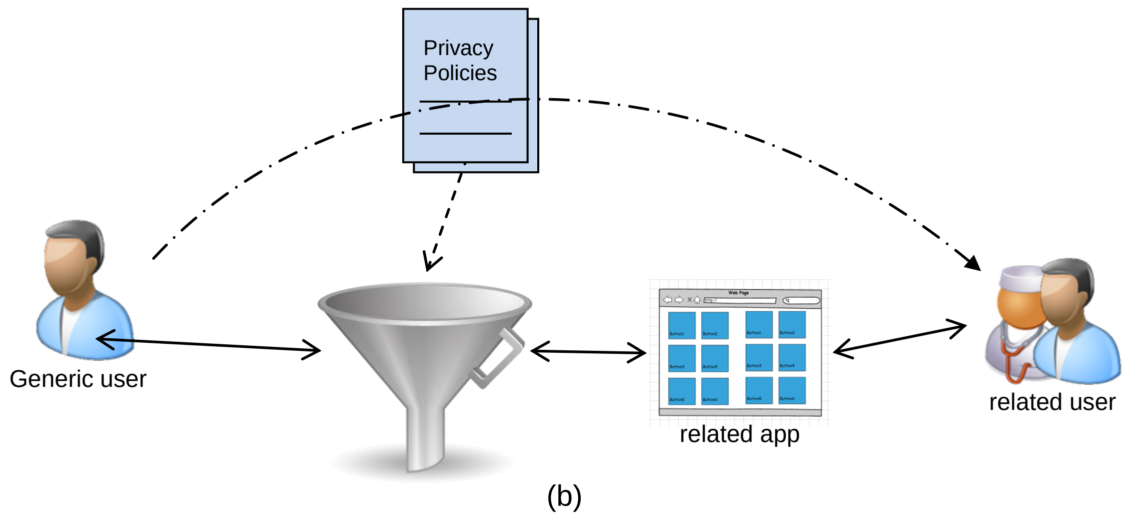


Figure 2 - privacy policies between app, user and related user (a). Use of privacy policy to give a certain access to the app and whose is using it (b)

2.2.2 Actors

See Appendix to have an overview about how users and relations will be managed by the platform

- **End user**
    - **Professional**: it may be a doctor, a psychiatrist, a personal trainer, or any other professional related with health.
        - She has a proper account on the Platform (of type "professional") or a subtype of it (assigned by one App Manager and recognized only by the specific App);

        - 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 relations
        - 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.
    - **Common user**: it is any person that wants to store her personal health data into the T-PaaS.
        - She has a proper account on the platform (of type “user”)
        - She can store personal data
        - She has relations with other user, professionals or applications
        - She can see data of the users she is related with (those with an incoming relation)
        - She has only outgoing relations with apps
- **Administrator**
    - **Platform Administrator:** she can be identified as the platform owner, who is in charge to make it functioning
        - she has a special account on the platform
        - she has no rights to see private data of other user
    - **Applications Administrator**
        - **Application Developer**: she is a person that develops an application that uses T-PaaS as identity, storage, policy, log service
            - She has a proper account on the platform (of type “developer”)
            - She has no ability to have relations with user, professionals or apps
            - She can register and manage her applications in the platform
        - **Application Manager**: she is someone in charge of an application with specific abilities like send request of creation of professional users within an app instance
            - She has a proper account into the platform (of type “app manager”)
            - She cannot have any relation between other users, professional or app to share data.

2.2.3 Use Case Overview

In the following, the identified use cases are discussed grouped according to the following functional packages. Each package represents a service.

- User mgmt. (UC 10, 20, 30, 40)
- Relations/privacy mgmt. (UC 50, 60)

- Auditing (UC 65, 160, 70, 80)
- App mgmt. (UC 110, 120, 60, 130, 140)
- Monitoring and benchmarking (UC 100)

2.2.4 Use Case Model

This section will explain in detail the scenario and the services described in the previous Sections. The Use Cases methodology will be described to highlight the actors involved in the system and their relationships and experience in using the final system, and they will be used for the definition of the reference architecture for the home healthcare scenario.

The use cases in this section are non-extensive for the overall platform. They are written to focus on the legal aspects that the platform can touch. The intention is to find new legal requirements in order to be A1-compliant and integrate properly the platform into the overall TClouds objectives and integration purposes.

2.2.4.1 End Users' activities

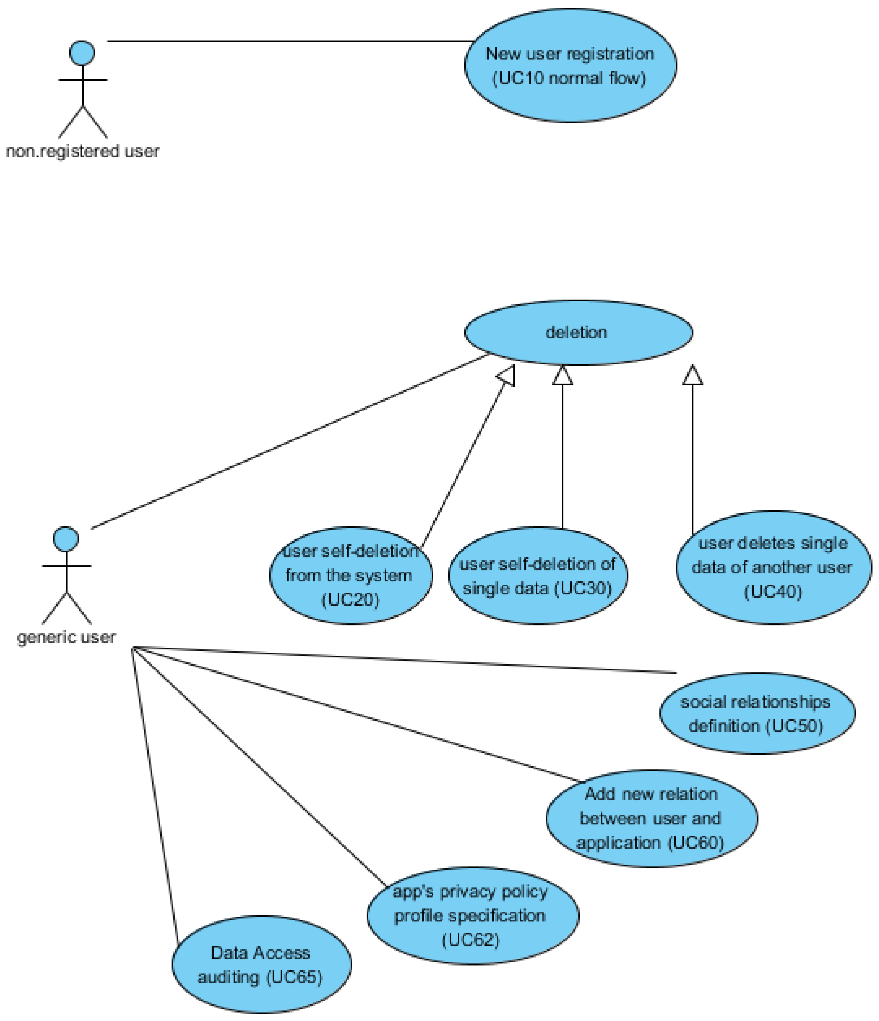

non.registered user
New user registration (UC10 normal flow)
deletion
generic user
user self-deletion from the system (UC20)
user self-deletion of single data (UC30)
user deletes single data of another user (UC40)
social relationships definition (UC50)
Add new relation between user and application (UC60)
app's privacy policy profile specification (UC62)
Data Access auditing (UC65)

Figure 3 - End users use cases overview diagram

Use case unique ID	UC 10 – New User registration
Description	A new user wants to join the Platform, and creates a new profile.
Actors	End User Application Administrator (alternative flow 2)
PreConditions	The user doesn't have an already existing account
PostConditions	The user is registered in the platform
Normal Flow	1) The user connects to the Platform web site 2) The user inputs all her information (including username and password) 3) The user accepts terms and conditions for the platform usage 4) The user confirms the new registration
Alternative Flow 1 **Registration made by the user through a third party application**	1) The entry point for the user is a third party application which needs to save data in our Platform. 2) The application requests all the needed data for the registration (as in the normal flow) to the user and post it to the Platform 3) The Platform sends an email directly to the user, with a password generated by the system, and with a link to confirm the subscription 4) The user can have a look at the term and conditions and must accept them to be subscribed. 5) The new registration is then confirmed.
Alternative Flow 2 **Registration made by an Application Manager through a third party application**	1) The Application Manager of a third party application needs to register the user (or several users) in our Platform. 2) The Manager must input (manually or programmatically) all the needed data for the registration (as in the normal flow) for the user 3) The Platform sends an email directly to the user, with a password generated by the system (and not known by the application), and with a link to confirm the subscription 4) The user can have a look at the term and conditions and must accept them to be subscribed. 5) The new registration is then confirmed.

Use case unique ID	UC 20 – User self-deletion from the system
Description	An existing user wants to delete its account and all its data
Actors	End User of a specific country
PreConditions	The user have an already existing account
PostConditions	The user can no longer login to the system

Use case unique ID	UC 20 – User self-deletion from the system
Normal Flow	1) The user logins to the Platform web site 2) The user goes to her private area and selects the "account deletion" link. 3) The system will logout the user and deletes the user data. Login from the user will not be allowed anymore unless a new registration is made (UC10).

Use case unique ID	UC 30 – User self-deletion/update/write of single data
Description	An existing user wants to delete/update/write some of her data
Actors	End User
PreConditions	The user have an already existing account and some health data are registered
PostConditions	Some user heath data are inserted, are no longer available or has been modified
Normal Flow: deletion	1) The user logins to the Platform (through the platform's web site or through an application) 2) The user selects the personal health data she wants to remove and confirms its removal 3) The system will remove the selected data and it will not be available anymore to the system.
Alternative Flow: update	1) The user logins to the Platform (through the platform's website or through an application) 2) The user selects the personal health data she wants to update and confirms its changes 3) The system will update the selected data
Alternative Flow: write	1) The user logins to the Platform (through the platform's website or through an application) 2) The user inserts personal health data and confirms this insertion 3) The system will write the new data in the database

Use case unique ID	UC 40 – User deletes/updates single data of another user
Description	An existing user wants to delete/update some of a friend's data
Actors	End User: User1, User2
PreConditions	User1: has an existing account and a relation with User2 and has the rights to delete/update User2 data. User2: has an existing account and a relation with User1. User2 gave deletion/update rights on her data to User1

Use case unique ID	**UC 40 – User deletes/updates single data of another user**
PostConditions	Some User2's heath data are no longer available or has been modified
Normal Flow: deletion	1) User1 logins to the Platform (through the platform's web site or through an application) 2) User1 selects the personal health data of User2 she wants to remove and confirms its removal 3) The system will remove the selected data and it will not be available anymore to the system.
Alternative Flow: Update	1) User1 logins to the Platform (through the platform's web site or through an application) 2) User1 selects the personal health data of User2 she wants to update and confirms its changes 3) The system will update the selected data.

Use case unique ID	**UC 45 – User access to data**
Description	A user wants to access her data (or another user's data) through either the platform or an authorized application.
Actors	End User: User1, User2
PreConditions	- User1 and User2 are registered in the system - User1 has an incoming relation with User2 - User1 has the rights to access User2 data.
PostConditions	User1 sees her (or User2's) data
Normal Flow: deletion	1) User1 logins to the Platform (through the platform's web site or through an application) 2) User1 selects the personal health data she wants to see in a given time interval 3) The system (either the Platform or the Application) will display the selected data
Alternative Flow: Update	1) User1 logins to the Platform (through the platform's web site or through an application) 2) User1 selects the personal health data of User2 she wants to see 3) The system (either the Platform or the Application) will display the selected data.

Use case unique ID	**UC 50 – Social relationship definition**
Description	An existing user wants to create a relationship with another existing user *Note: relations between users are Google+ like (for which the relation User1→User2 is different to User2→User1. See Appendix)*

Use case unique ID	**UC 50 – Social relationship definition**
Actors	End User: User1, User2
PreConditions	User1: has an existing account and no relation with User2 (no User1→User2) User2: has an existing account (no matter if the relation User2→User1 exists)
PostConditions	Some User2's heath data are no longer available due to data deletion
Normal Flow	1) User1 logins to the Platform (through the platform's web site or through an application) 2) Step1: User1 searches for User2 and inserts User2 among her friends. 3) Step2: User1 selects the privacy restrictions for the new relation (User1→User2). This privacy affects the experience of User2 to make CRUD operations on User1's data (CRUD = Create, Read, Update, Delete)

Use case unique ID	**UC 60 – Add new relation between user and application**
Description	A User wants to specify new privacy policies for a specific application
Actors	End User
PreConditions	The application registered into the T-PaaS system. The user is logged into the system (via the T-PaaS web site).
PostConditions	The application can access to User data according to the policies defined by the user
Normal Flow	1) Step1: The user goes in the private area and adds/selects a specific application among her app list. 2) Step2: the user selects the privacy policies that limits the application's access to her data

Use case unique ID	**UC 62 – App's privacy policy profile specification**
Description	A User wants a new application to access her data
Actors	End User
PreConditions	The application is registered into the T-PaaS system. The user is logged into the system (via the T-PaaS web site). The application is not yet related with the user (no relation User→App)
PostConditions	The application can access to User data according to the policies defined.

Use case unique ID	UC 62 – App's privacy policy profile specification
Normal Flow	1) The user search in the app index on the system website the application she wants to add. 2) Once selected the app, the system will show the minimum policy requirements needed by the app to run properly 3) The User can choose whether to accept the minimum requirements (and thus create the new relation between the user and the application) or deny the minimum requirements (with the consequence to have the app not related with the user.

Use case unique ID	UC 65 – Data Access Auditing
Description	A User wants to see the history of who has accessed to which data
Actors	End User
PreConditions	The user is registered and logged in T-PaaS via the platform website
PostConditions	The user is able to see the log of her data access
Normal Flow	1) The user selects the page relative to history of data access in her private area on the platform website 2) The platform provides a way to select the window time and all the auditing data appear accordingly

2.2.4.2 App administrators' activities

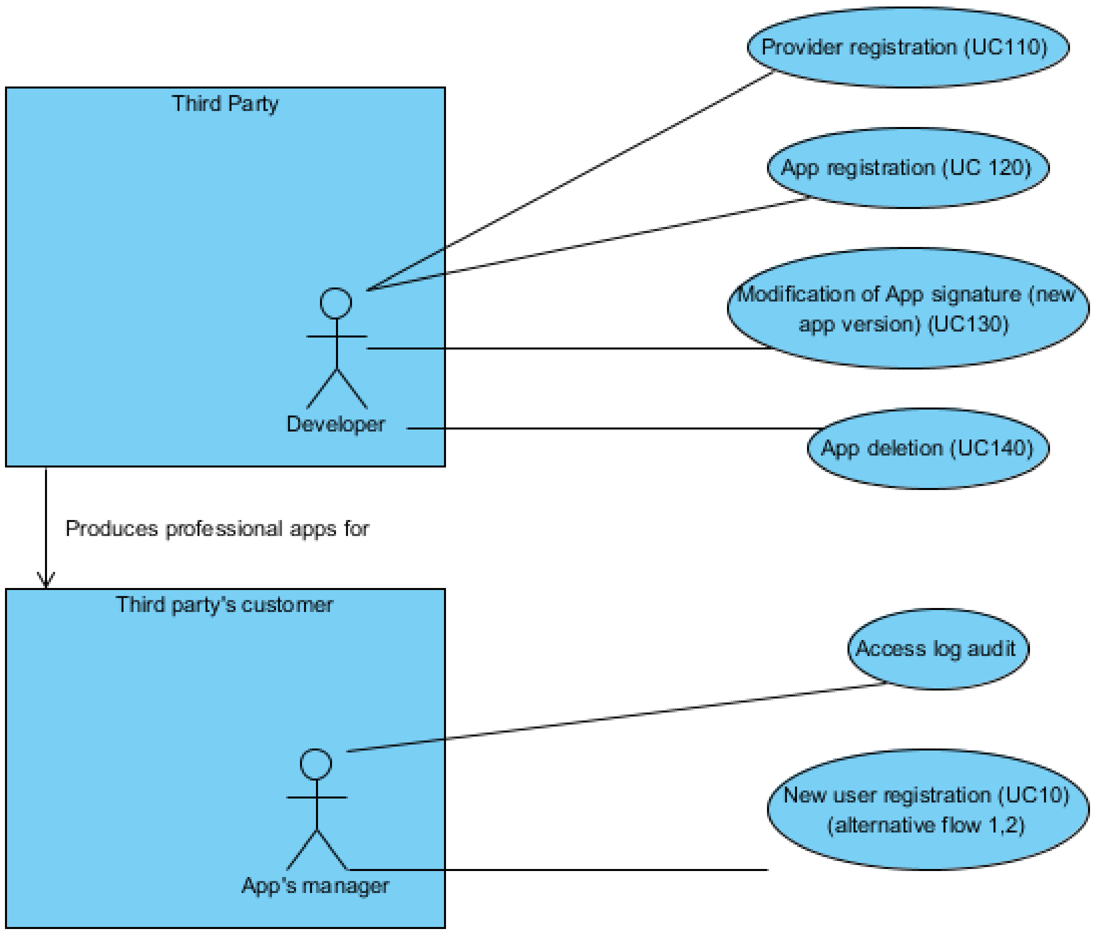

Third Party
Developer
Provider registration (UC110)
App registration (UC 120)
Modification of App signature (new app version) (UC130)
App deletion (UC140)
Produces professional apps for
Third party's customer
App's manager
Access log audit
New user registration (UC10) (alternative flow 1,2)

Figure 4 - App administrator use cases overview diagram

Use case unique ID	**UC 110 – Provider registration**
Description	A third party subject wants to register into the system as provider to deploy her application
Actors	Third party developer
PreConditions	The third party is not yet registered into the system
PostConditions	The third party is registered into the system
Normal Flow	1) The third party developer access to the platform's developer sign-on area 2) She inserts all the useful information about the third party (such as email and company name) and confirms them 3) The Platform sends an email directly to the developer, with a password generated by the system, and with a link to confirm the subscription 4) The developer can have a look at the term and conditions and must accept them to be subscribed. 5) The new registration is then confirmed.

Use case unique ID	**UC 120 – App registration**
Description	A third party's developer wants to add a new application's minimum policy requirements to allow the app, once deployed, to be recognized by the platform
Actors	Developer
PreConditions	The third party's developer is registered and logged into the system while the app is not yet registered
PostConditions	The application is registered into the system and grants the rights to dialog with the platform.
Normal Flow	1) The third party's developer goes to a specific page within her personal area in which she can add a new "app signature" 2) She inserts all the useful information about the app (such as name, version, minimum policy grants needed, …) 3) The system registers it in the index of the official apps of the system.

Use case unique ID	**UC 130 – Modification of app signature (new app version)**
Description	A third party's developer has added new functionalities to the app and needs to extend/modify the minimum policy requirements of the app
Actors	Developer, users of the app
PreConditions	The third party's developer is registered and logged into the system and the app is already registered
PostConditions	The new app profile is registered into the system with the new related privacy policies
Normal Flow	1) The third party's developer goes to a specific page within her personal area in which she can see all her app registered in the system and she selects the app of which she wants to modify the privacy policy. 2) The system asks the new App signature info (version, new privacy policies, …) 3) The responsible confirms the changes. 4) The system will send a notification to the app's users (those users that already have a relation with the app) and asks them to accept the new privacy policies User can either o accept the new policies o deny the new policy and maintain the old privileges (this might lead to app malfunctioning, but it will be responsibility of the app providers to solve the issue)

Use case unique ID	UC 140 – App deletion
Description	A third party's developer wants to delete permanently her app to the platform
Actors	Third party developer, users of the app
PreConditions	The third party's developer is registered and logged into the system and the app is already registered
PostConditions	The app is not registered anymore
Normal Flow	1) The developer goes on a specific page in the restricted area on the platform website to remove apps 2) She selects the app she wants to remove and confirms it 3) The system will send a notification message to all the app's user with the reminder that within a certain period of time (e.g., 30 days) the application will be permanently deleted from the platform 4) After a given time period (e.g., 30 days), the system: a. Will remove automatically the app signature and the app will no longer be able to use the T-PaaS API. b. Will remove the relation between the users and the app

Use case unique ID	UC 160 – App Manager requests the access log
Description	The Manager of the application asks an audit to the platform at API level for the specific application she administer
Actors	The app's Manager
PreConditions	The app's Manager is logged in the platform website
PostConditions	The app's Manager can see the access log of the app
Normal Flow	1) The Manager goes to a specific page within her private area and performs an access audit, given a specific time window, of a specific application she administers. 2) The platform website provides a log (which integrity is guaranteed) with the information about which professional user of the app has accesses which data.

2.2.4.3 Platform Administrators' activities

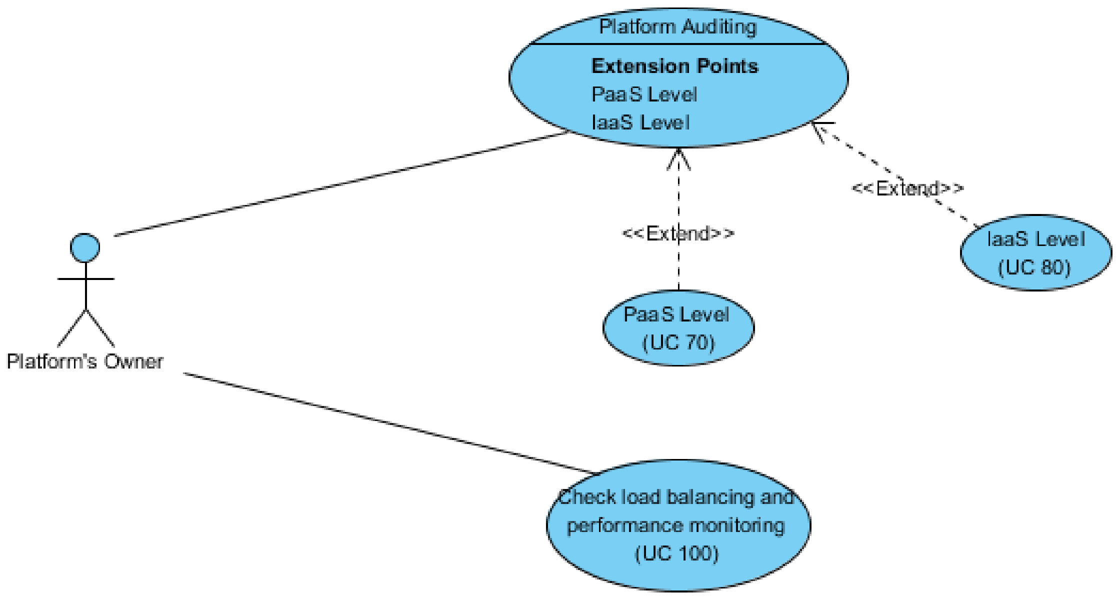


Figure 5 - Platform administrator use cases overview diagram

Use case unique ID	**UC 100 – Check load balancing and performance monitoring**
Description	The platform administrator wants to perform a performance monitoring of the system
Actors	The platform's administrator
PreConditions	The platform's administrator is logged into the system
PostConditions	The administrator is able to see the monitoring she needs to do
Normal Flow	1) The administrator goes to a specific page within her private area in the app website to see the dashboard of all the platform performances 2) She can decide whether to instantiate new VM or manage single node of the platform in order to increase the performances

Use case unique ID	**UC 80 – Platform Auditing (IaaS Level)**
Description	The Platform's owner wants to perform an audit at Infrastructure level to retrieve the log about the images instantiated / running app servers…
Actors	Platform owner
PreConditions	The owner is logged into the running system
PostConditions	The owner is able to see the log of the system image(s) instantiated on the infrastructure

Use case unique ID	UC 80 – Platform Auditing (IaaS Level)
Normal Flow	1) The platform's owner access to a specific page within her personal area in the platform's website. 2) She selects the data range and starts the audit session 3) The system shows a secure log that certify its integrity and shows the info at Infrastructure level

Use case unique ID	UC 70 – Platform Auditing (PaaS Level)
Description	The Platform's owner needs to perform an audit on users' data access (e.g., asked by a tribunal)
Actors	Platform owner
PreConditions	The owner is logged into the system
PostConditions	The owner is able to see the log of data access of a selected user
Normal Flow	1) The platform's owner access to a specific page within her personal area in the platform's website. 2) She selects the data range and the username to make the audit to. 3) The system shows the log entries and highlights those entries that are exploited

2.3 Legal analysis

By analyzing the use cases, several questions arise regarding legal implications, specially related with cross border data transfer (transferring of data within EU country and outside), deletion/update of data and data ownership. An eye has to be thrown also on the fact that the platform, the applications and the databases can be located in three different sites around the globe thus has to be analyzed the scenario in which storage (DB level), aggregations (platform level) and elaboration (app level) of data might arise three different legal issues.

A deep legal analysis is demanded in D3.1.3 and the implementation of T-PaaS will take into account all the requirements that the Legal Experts will find.

Chapter 3

Existing Health PaaS offerings

Chapter Authors:

Ya Liu (PHI), Mina Deng (PHI)

An ever-growing list of personal health records service providers have adopted Cloud computing for service improvement and security enhancement. According to OpenCrowd, the top healthcare and wellness service providers in cloud computing market are HealthVault, TPSC (the patient safety company) Cloud, WebMD, HeritageDigital and Google Health, among which HealthVault, TPSC Cloud and Google Health provide platform-based healthcare service.

Platform-based healthcare service or Health PaaS is a technology for managing and sharing personal health records with hospitals, healthcare professionals and other 3rd parties. Different from traditional web-based personal healthcare service, health PaaS applies secure and resilient cloud infrastructure for computing capability and storage, on top of which it generally provides identity management, access control for user records and an open platform for 3rd party integration to enrich its healthcare service. Since Google Health discontinued its service in 2011, this chapter will provide an overview of HealthVault and TPSC Cloud in utility, security, privacy and development aspects.

3.1 Introduction of HealthVault

Launched on October 4, 2007, HealthVault is a platform-based free personal health record (PHR) service from Microsoft providing storage for user's health data, basic health information management and an open platform for 3rd party integration.

3.1.1 Utility

3.1.1.1 User accounts

Currently HealthVault accepts Windows Live ID, Facebook account and OpenID from certain identity providers. However, the registration of HealthVault is currently available only in the following regions/countries: US, American Samoa, Guam, Northern Mariana Islands, Puerto Rico, Virgin Islands and UK. With one account, user can manage the health of a child, parent, or other family members, by adding records for them.

3.1.1.2 Data collection

HealthVault provides a few ways to upload health information. User can type it in manually and upload documents including pictures and patient records. HealthVault also integrates applications that can be used to aggregate and transform user record into digital data. Various authenticated devices, like pedometers and blood pressure monitors, can be applied to add data directly. Meanwhile, user can authorize pharmacies, labs, hospitals, and clinics online to send existing information to HealthVault.

3.1.1.3 Information usage

The services users can benefit from HealthVault highly depend on 3rd party applications. A variety of health tools authenticated by HealthVault can help user to generate reports, offer health-related education and professional recommendations. User can track records and analyze trends and even receive prediction and suggestion from 3rd party applications. Currently, there are 107 3rd party applications and 21 compatible devices authorized by HealthVault.

3.1.2 Security and Privacy

3.1.2.1 Record sharing and access control

Once user creates an account, he/she will be the owner of this account with access to all the records including read, write and delete. In addition, the owner can grant to other users different level of access to the record and revoke the access of anyone to a certain record.

User can also share personal and health information with 3rd party services by authorizing their access. The access request will be shown to the user when he/she applies a new service. The request includes what and how the application will access the user's information. A complete log of how the record is accessed can be monitored by users. (Solutions, 2011)

3.1.2.2 Record deletion

Data deletion in HealthVault can be either permanent or temperate. Once the record is deleted by the owner, the record will be removed from all sharing points temperately. HealthVault will keep the deleted data for 90 days before permanently revoke in order to avoid accidental or malicious deletion of any critical health information. (Solutions, 2011)

3.1.2.3 Authentication

For 3rd party applications, both web applications and desktop applications need to be associated with security certificates that need to be issued by the HealthVault Application Manager. Therefore, the integration is within a trusted domain where user credentials for HealthVault will never be leaked to untrustworthy3rd parties.

3.1.2.4 Others

All communications sent by HealthVault, except e-mail, is through encrypted channel.

3.1.3 For developers

3.1.3.1 3rd party integration

HealthVault platform offers different level integration for developers. Application can just use the storage provided by HealthVault (similar to IaaS) as integration, or establish a full links between its own PHR and HealthVault record. The list below illustrates the difference of each integration level: (Sean, 2010)

- Native HealthVault: the application uses HealthVault user authentication, data types and storage.
- Linking: The user authorizes an application to link to their HealthVault record so that the application can read from or write data to HealthVault with or without user interventions.
- Patient connect: The user authorizes an application to read or write data to their HealthVault record via an experience on HealthVault.com

- Drop-off/Pick-up: The application drops off data to HealthVault and user picks it up and adds it to their record.
- Software on Device Authentication: Software on Device Authentication (SODA) applications are client applications whereby each install of application has a unique identity in HealthVault.

3.1.3.2 Libraries

Microsoft HealthVault only provides a .NET Developer SDK for development. Even though there are some independent groups building Java and Ruby library for Microsoft HealthVault integration, the functionality is limited for other programming language. Further, Microsoft's HealthVault Application Manager, which need to be uploaded to the platform and used with the library, is only .NET compatible.

3.1.4 Comments

Highlights:

- Outstanding market awareness for its IT services
- Easy sign-up for users (in US and UK)
- High flexibility of data collection and sharing
- Secure and various 3rd party integration
- Extensive APIs and documents for a better developer experience

Cautions:

- Limited service area
- Only .NET SDK is provided

3.2 Introduction of TPSC Cloud

The TPSC Cloud[1] is a PaaS provider for healthcare organization management and application development. Different from HealthVault which is a user oriented service, TPSC Cloud is a developer oriented service for healthcare system management and application development by offering its cloud infrastructure.

3.2.1 Utility

3.2.1.1 Cloud infrastructure

Different from HealthVault, TPSC Cloud provides cloud infrastructure for customers, which includes DB storage, Objects storage, network traffic and etc. In addition, data backup and restore are also offered as service as well as technical support.

3.2.1.2 Platform management

Various standard modules (TPSC Document, TPSC Task, TPSC Investigate, TPSC Anticipate and TPSC Improve) above the infrastructure give customers a wide range of business and healthcare functionality. For instance, user can track the tasks by creating graphical overview and setting E-mail notifications. TPSC Document provides various ways

[1] TPSC Cloud - The Patient Safety Company: http://www.patientsafety.com/nl/tpsc-cloud

to manage the file system of the platform, including operations like create, distribute, modify and search.

3.2.1.3 All about Apps

Standard integrated applications are offered by default, such as AHRQ(Adversary event) for healthcare event reporting and Complaints for collecting managing and monitoring patient complaints. More applications can be purchased from App Center. Meanwhile, developers can also design and create applications using tools from TPSC platform. By easy "drag & drop", developers can design workflow and create online forms.

3.2.2 Security and Privacy

3.2.2.1 Infrastructure security

The security and resilience of physical environment of its data centers is enhanced by UPS (Uninterruptible Power Supply) power with HVAC (Heating Ventilation Air Conditioning) system. Data backup and restore is provided as default service for all users with different backup zones and 14 recovery points. All the communication within the Cloud is secured by SSL. In addition, TPSC Cloud has obtains industry compliance based certifications includes SAS 70 Type II and HIPAA. However, the data isolation and network security management are not specified from web resource.

3.2.2.2 Security and Privacy as Service

The development platform of TPSC Cloud offers managed or standard tools for developers to enhance security and privacy of their applications. It includes user authentications, user location management, application-based firewall configuration and role-based access control.

3.2.3 For developers

Since TPSC Cloud is developer-oriented, it offers a full range of support from application development to task and file management. Especially, a considerable amount of work will be reduced for the developers by its "drag & drop" feature. However, this advantage limits the flexibility of customization, which means developers can only create their own applications over this platform but nothing else.

3.2.4 Comments

Highlights:

- Ideal for private and customised healthcare management.
- No programming task for app development
- Full range of control for the platform administrator

Cautions:

- Unclear cloud infrastructure, especially for security related features
- Limited 3rd party application to choose from (only from App Center)
- Poor documents and technical information on web

Chapter 4

Proposed concept of health T-PaaS

Chapter Authors:

Marco Nalin (HSR), Marco Abitabile (HSR)

4.1 T-PaaS

4.1.1 Description of the Health T-PaaS

This section describes the details of the functionalities and services offered at the Platform as a Service (PaaS) layer, trying to outline what can be called a Health Trusted PaaS, or Health T-PaaS. Indeed, some of the services actually have an interface also toward end users, and we can consider them Software as a Service (SaaS). The Health T-PaaS wants to be a multilevel platform whose aim is to provide novel services as:

- Stores trustworthily health-related data (relying on a trusted IaaS);
- Provides API for 3rd party apps to access to users' health data, in a privacy-preserving manner (PaaS);
- Provides API for 3rd party apps to use identity/role management services (PaaS);
- Provides an interface to allow 3rd party apps' developers to register their application(s) in order to access the users' data (SaaS);
- Allows 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 has been produced in the TClouds project, and it is shown in figure 1. Even if the picture is very high level, it shows the boundaries of the Health T-PaaS, 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, described in section.

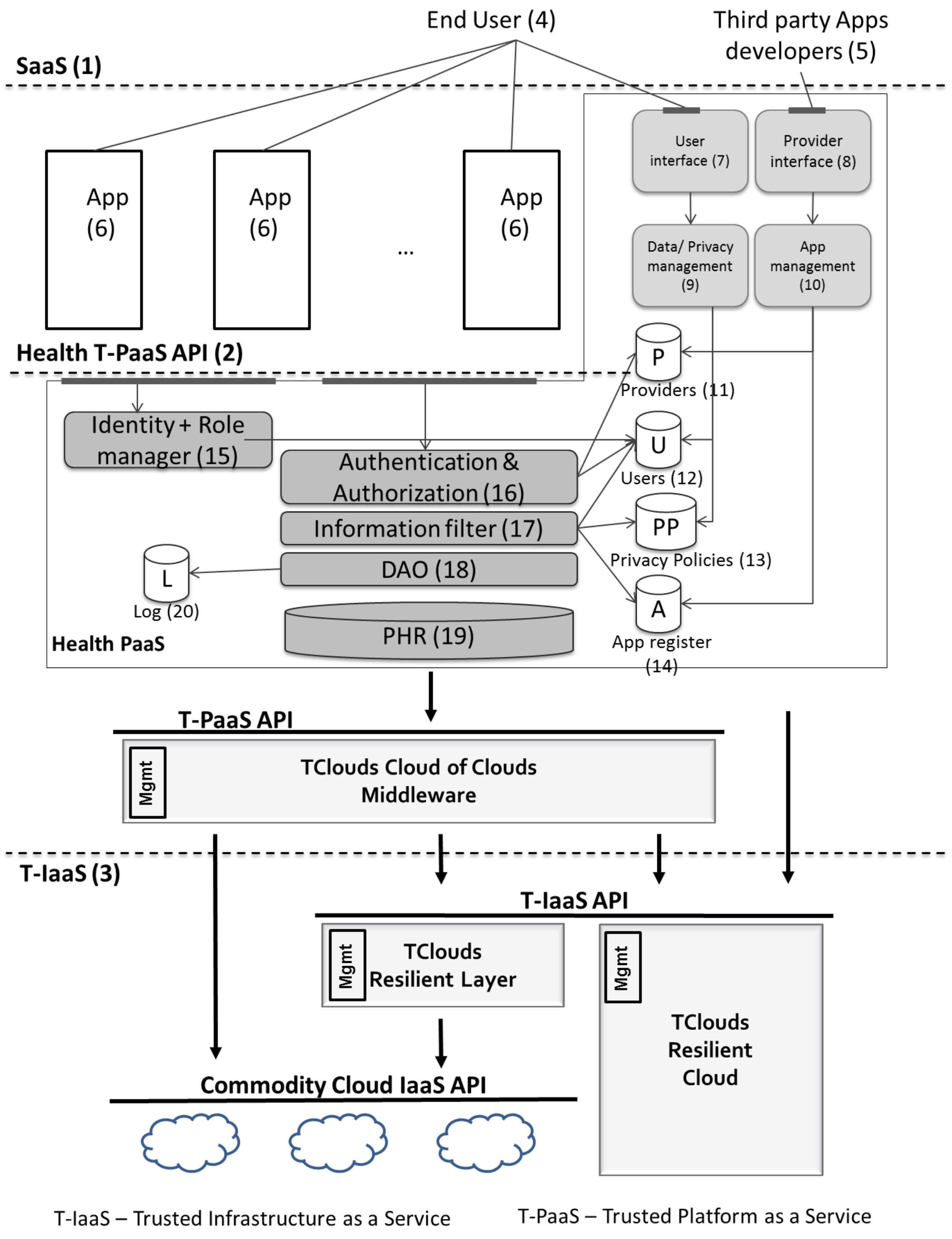


Figure 6 - Prototype architecture for Health T-PaaS

4.1.2 Application interfaces to the platform (PaaS)

It is expected that many different external applications (#6 in figure 1) will be deployed by different Service Providers, using their own infrastructure (or the Trustworthy IaaS cloud, but this isn't mandatory), and they will be able to access through ad hoc web services to the functionalities offered by the platform. The middleware part of the platform (#11 to #20) is the core of the Health platform. Basically it provides:

- An interface for applications to retrieve roles and identities of the platform users (OpenID-like APIs, with improved privacy protection, as once the user will be logged, just basic information will be provided and not the full access to patients' data);
- Interface for applications to CRUD (Create, Read, Update, and Delete) operations over the users' personal data.
    - Applications must be authenticated (with OAuth like features (see Chapter 6 for more details about OAuth))
    - Applications can access only to a subset of patient's record, where information are filtered based on the privacy policy specified by the end users.

It is important to underline the difference between the two functionalities offered. Most of the personal information will not available to all the applications once the user logs in through her OpenID-like account. Just a small subset of the data is made available (username, name and surname, role in platform, etc.) while the rest must be retrieved by the applications through dedicated web services. These web services will require an OAuth like authentication protocol (most likely an extension of this protocol), and the access to specific CRUD operations will be granted only depending on users' privacy settings.
The databases represented (#11 to #14, and #19) are just an indication of the models that the platform will be able to handle. There is the concept of Service Provider (#11), meaning an external provider who wants to leverage on the features offered by the Health T-PaaS. Every Service Provider can deploy one or more Applications (#14), actual software developed and offered directly to their customers.
Of course also the Users (#12) are adequately represented with all the needed information to access the portal (e.g., username, passwords, profile image, etc.). All the personal information that the platform can hold are saved in the Personal Health Record database (PHR, #19). Another important database is the access log (#20), which, among the other things, will save all the accesses to personal data, and will be the base for the auditing tool. An interesting functionality of the Health T-PaaS is indeed the possibility to provide logs both to the applications and to the end users. In case of controversies related to privacy and personal data protection, the end user will have full transparency on who accessed their data, as well as the Service Providers, if sued by end users of infringing privacy laws, can require a log to the Health T-PaaS and demonstrate that its access to the data was always lawful and approved by the user.

4.1.3 User interfaces to the platform (SaaS)

As mentioned before, the end users need to have also a direct access to the platform, and for this reason we can define a SaaS layer to manage and control the platform. There are in particular two types of users:

1) Application developers: These are the actual developers working for the Service Provider (or for the technology provider of a Service Provider, like for example the Electronic Health Record provider for a hospital). Developers will have means to access the platform, register themselves with all the legal information of their company, and register new applications. When registering an application, the developer must specify which type of data this application will access, and this step (which will be enforced by the authentication layer, #16) will create a sort of signature of the application. When an end user will authorize this application to operate with her personal data, she will be able to define exactly to which information the application will have access to. This is particularly useful to avoid for example a physical activity application to access to protected medical data, without the user's explicit consent, or forcing her to accept this compromise (i.e., everything or nothing) in order to use the application. With the registration step, the App is inserted into an application list, which can work also as a Yellow Pages index when users are looking for specific applications to manage specific activities related to lifestyle (e.g., diet, sleep, etc.).

2) End users: The actual persons who will insert their personal data into the portal, as well as healthcare and wellbeing professionals (from doctors, to nutritionists, to personal trainers, etc.) who will monitor patients/customers' data. The users will have full control over their personal data, which means that they will be empowered to enable/inhibit other users, applications or service providers to access their data, as well as to decide the access level (read, write, delete, etc.) for every actor and, most important, to monitor and audit at any time who, when and how accessed their data. Should a user lose confidence in a given provider, user or application, she will be able to immediately inhibit it from accessing her data. Furthermore, if a given application changes its signature (for example asking to access to more data than originally planned and authorized), the user will be immediately notified and the application won't have access to the data until the user explicitly authorizes it.

4.2 Prototype and benchmark applications

The first prototype of the Health T-PaaS is under development in the TClouds project, it is leveraging on the TClouds IaaS, and it will be tested by some benchmark applications. The purpose is to demonstrate how it's possible to take advantage from the platform, through the usage of the APIs, with clear examples of applications that work on it and clarifying how the sharing of data could takes place.

4.2.1 Benchmark applications

The two health benchmark applications developed in the TClouds project are created to illustrate a process that includes the collection of data from a personal device, storing them in the platform, and sharing them with proper privacy management. In particular, a first application is centered on a device that collects sleep and light data, which is very significant for depressed patients. Sometimes these information need to be shared with other applications that provide actual services, using medical data collected, as, for example, the second health benchmark application, which provides a Well-Being Portal for the depressed patients.

1) The personal device application: The first application is a basic example of software created to upload data collected by an external device. In general, given the device selected (Philips-Respironics Actiwatch) the upload could be done only by a doctor, but this application allows the patient to manage her data and to ensure its trusted redistribution to others Services Providers, if needed.
2) The Wellbeing Portal application: The second prototype application is a portal which allows the depressed patients to self-manage their disease. It contains a lot of services to collect and analyze data, including the possibility to show the trends of sleep activity, light exposure and mood variations. The patient can authorize her doctor (e.g., her psychiatrist) to watch her data, and the application allows to the patient to specify privacy policies to limit the information that should be displayed to the doctor. To add data in her personal records, the patient has two possible ways: the system can reuse data uploaded from other Service Providers properly registered, authenticated and authorized (e.g., the personal device application described above), or the patient can insert manually data in the system, through the Wellbeing Portal application.

4.2.2 Usage example

To clarify how the actual platform could be used, we describe a brief use case, involving the two prototype applications described above. The use case includes the following steps:

- Philips wants to upload data collected with the devices in its portfolio (e.g., the ActiWatch) to the Health T-PaaS. A developer from Philips registers himself to the

Health T-PaaS and insert all the information about his company. After this step Philips is officially registered as a Service Provider in the Health T-PaaS.

- The developer can now start to create the ActiWatch application, and he registers it in the Health T-PaaS, by declaring to which data and with which methods (create, read, update, delete) the application will access to.
- A new patient, Alice, is subscribed to the Health T-PaaS, and she can manage her applications, contacts (e.g., friends, doctors, etc.), and privacy settings.
- The patient notices the ActiWatch application (i.e., the personal device application described above) and decides to authorize it to access her data.
- The patient now accesses the Actiwatch application provided by Philips, the data is periodically downloaded from the device and transferred to the Health T-PaaS.
- The patient adds also the Wellbeing Portal to her group of applications and authorizes it to use her data (specifying which data are allowed through the Health T-PaaS privacy settings interface). Now she (and the other authorized users, like her doctor) is able to monitor the data using the Wellbeing Portal as interface, with all the services provided on top of the patient's information.
- Philips decides to add also physical activity monitoring to the ActiWatch application. The Philips developer can implement the new features, but he must specify the new data access policy for the same application.
- Alice is immediately notified of a change in the application requirements, and the ActiWatch application is blocked from accessing Alice's data until she provides an explicit consent.

4.2.3 Benefits for the application and for the user

The Health T-PaaS offers many services which provide benefits both to the end-user side and to the application Service Providers.
The major interest for the end-user is to ensure the security for her own data (in particular, medical data are very sensitive and need to be stored and managed with more accuracy), and to check that her privacy rights are guaranteed. The Health T-PaaS offers three principal benefits:

1) Trusted platform: the whole structure of the platform is studied to ensure a trusted environment, compared with the general services offered by the others Cloud Providers. The trusted platform is necessary to preserve data from malicious users and to grant that the system will distribute the data accordingly to the patient's wishes.
2) Full control over personal data: When the end-user is a patient, she can specify which applications and which persons can see her data, and she can change these settings at any time. This ensures the patient to own the full control over her personal data. Moreover, every time an application, or a different user (e.g., the psychiatrist) wants to access a new kind of data, an alert is sent to the patient, and she can decide to issue or not the permission.
3) Data access control: The third benefit for the end users is the monitoring of the user's data accesses. A patient can check if the privacy settings are observed by looking at the list of the accesses to her personal data, and she can be fully aware about which applications/ persons actually observed the data and how frequently (e.g., useful to check possible misuses).

There are several benefits also for the applications Service Providers to use this platform instead of developing their own repository to store PHR data. These benefits are:

1) Increase potential number of users: As several Service Providers can use the platform, a newcomer can benefit from a quite large pool of already available users as potential customers for its application. Furthermore the platform offers an application indexing feature, which will provide a distribution channel for new applications.

2) Single Sign On: The login is granted by an OpenID-like mechanism, with some improvements to ensure more privacy. The applications don’t need to create an identification mechanism for their users, they take advantage form the structure given by the platform.
3) Unique data structure: If two applications want to communicate among them, it’s not required to agree on a common data exchange format, as they can communicate in a secure way through the platform. The applications that want send or receive data have to use the APIs available from the platform, where the data format is well established and univocal for all the applications subscribed.
4) Dynamic service composition: The platform is studied to support different Service Providers that can produce or make use of patients’ personal data, allowing a more dynamic service composition and marketing strategy. This makes it easier for a provider to change the commercial partner if needed, as this will not impact the software development.

4.3 Business Model

In the image below is depicted the relation between final users, third party developers, applications and the platform.

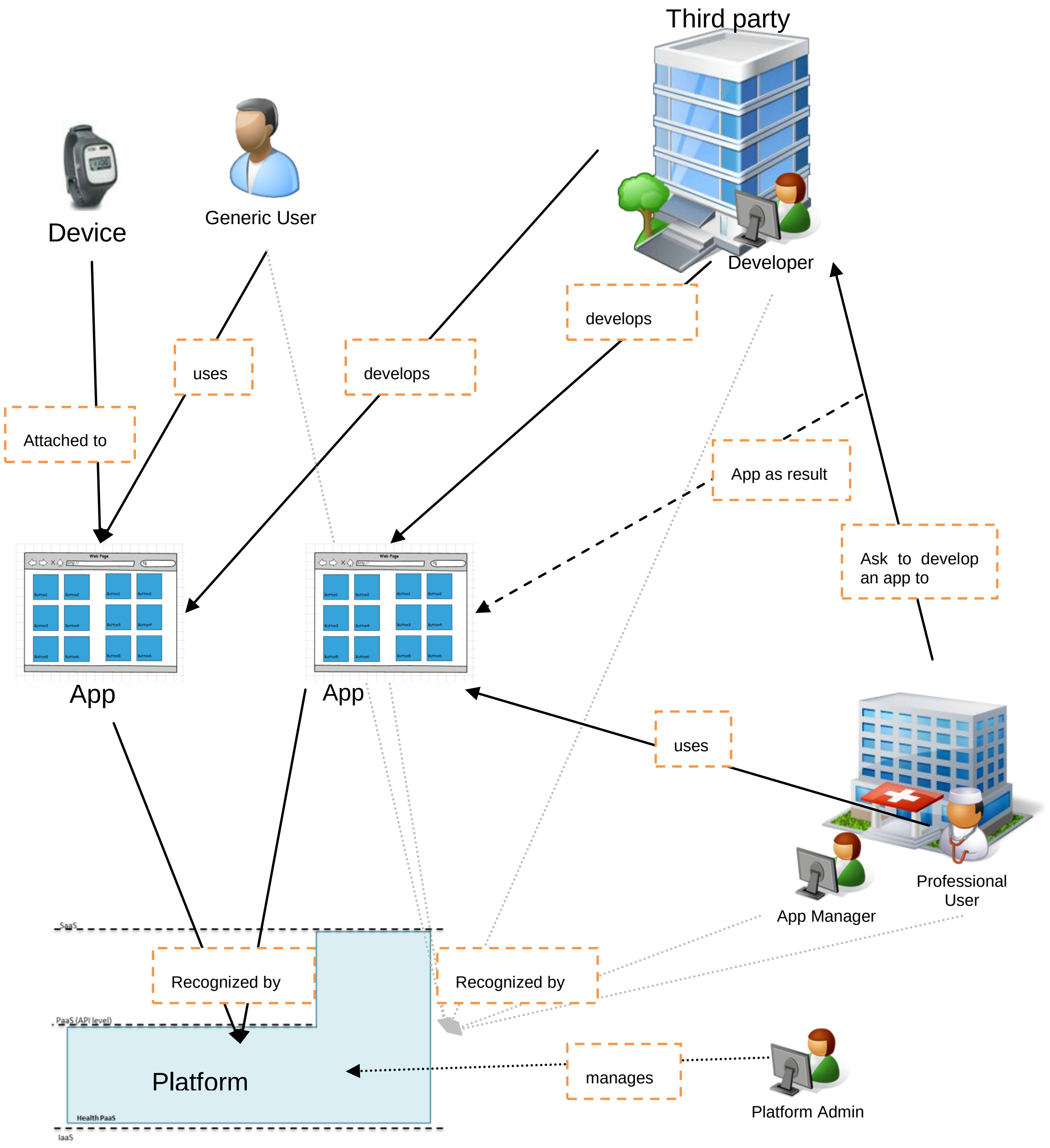


Figure 7 - Overview of T-PaaS and relations with users

This will allow the platform to have a complete control of the data and the accesses to it.

Chapter 5

Health T-PaaS API definition and application side trust protocols

Chapter Authors:

5.1 *Ya Liu, (PHI), Sebastian Banescu (PHI), Marco Nalin (HSR), Marco Abitabile (HSR)*OAuth APIs

In order to establish a relationship between user and application, the user is able to authorize certain applications to access his/her data through OAuth[2]. Figure 2 presents the authorization flow of OAuth. It involves three entities: the user, the third party application and the Trusted PaaS. The API for the requests in each step is presented in the following subsections.

5.1.1 Step 1: Obtain request token

After the application has been registered on the platform it receives a *Customer Key* and a shared secret is established between the two entities. In order to initiate the user authorization step, a request token first needs to be obtained by the application from the platform.

Request:

URL	oauth/get_request_token
Method	GET/POST

Parameters	**Description**
oauth_consumer_key	(required) Consumer Key. It is provided by T-PaaS during app registration.
oauth_nonce	(required) Nonce. A random string.
oauth_signature_method	(required) The signature method should be “HMAC-SHA1” here.
oauth_signature	(required) Signature which is signed by the consumer secret that was issued to the application.
oauth_timestamp	(required) Timestamp. Current timestamp of the request
oauth_callback	(required) Call back URL. The value specified here will be used as the URL that users are redirected to after they authorize access to their private data
oauth_scope	(optional) list of URLs identifying the services offered by T-PaaS that need to be accessed
oauth_version	(required) The oauth_version is 1.0

[2] http://oauth.net/

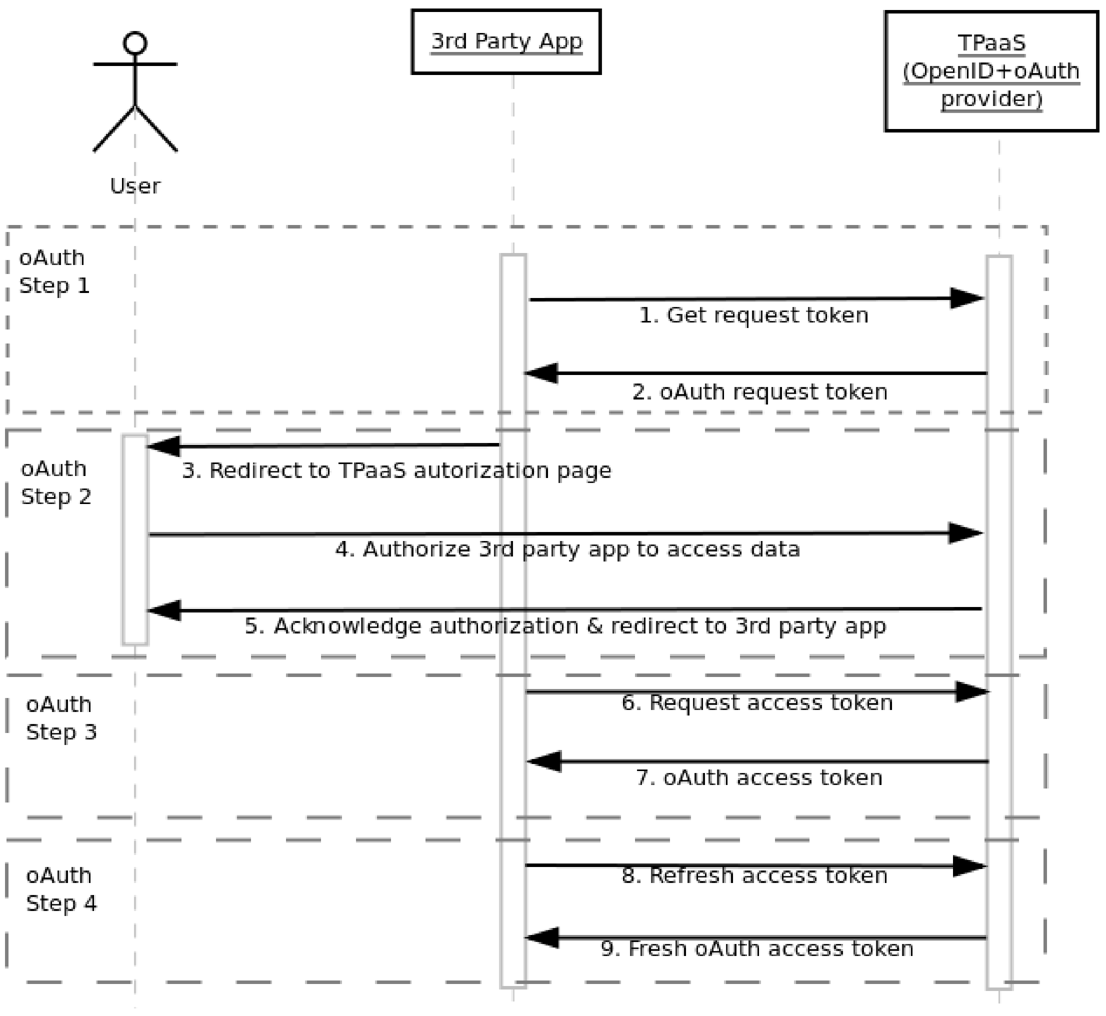

3rd Party App
TPaaS (OpenID+oAuth provider)
User
oAuth Step 1
1. Get request token
2. oAuth request token
oAuth Step 2
3. Redirect to TPaaS autorization page
4. Authorize 3rd party app to access data
5. Acknowledge authorization & redirect to 3rd party app
oAuth Step 3
6. Request access token
7. oAuth access token
oAuth Step 4
8. Refresh access token
9. Fresh oAuth access token

Figure 8 - OAuth Authorization Flow

Response:

Parameters	**Description**
oauth_token_secret	The secret associated with the request token.
oauth_expires_in	The lifetime of the request token in seconds.
oauth_request_auth_url	The URL to the T-PaaS authorization page.
oauth_token	The Request Token that T-PaaS returns
oauth_callback_confirmed	It is always “true”

5.1.2 Step 2: Direct user to authorization page

After obtaining the request token the application can redirect the user to the T-PaaS authorization page. Once user authorizes the third party application to access his/her data, T-PaaS will respond to the application with oauth_token (Request Token) and oauth_verfier which is a code tied to Request Token.

Request:

URL	oauth/authorize
Method	GET

Parameters	**Description**
oauth_token	(required) Request Token obtained from Step 1.
oauth_nonce	(required) Nonce. A random string.

5.1.3 Step 3: Obtain Access Token

In order to obtain access to the actual user data, the application needs to exchange the request token for an access token from the platform. The access token tells the platform that the user has allowed the application to access his/her data.

Request:

URL	oauth/get_request_token
Method	GET/POST

Parameters	**Description**
oauth_consumer_key	(required) Consumer Key. It is provided by T-PaaS during app registration.
oauth_nonce	(required) Nonce. A random string.
oauth_signature_method	(required) The signature method should be “HMAC-SHA1” here.
oauth_signature	(required) Signature which is signed by the consumer secret that was issued to the application.
oauth_timestamp	(required) Timestamp. Current timestamp of the request
oauth_verifier	(required) Verification code. It is obtained from Step 2. For OpenID+OAuth, verifier is not needed.
oauth_token	(required) Request Token
oauth_version	(required) The oauth_version is 1.0

Response:

Parameters	**Description**
oauth_token	The Access Token that T-PaaS returns
oauth_token_secret	The secret associated with the Access Token.
oauth_expires_in	The lifetime of the request token in seconds.

5.1.4 Step 4: Refresh access token.

An access token can be used for a limited amount of time (e.g. 1 hour). In order to obtain a new access token, the old token has to be sent as a parameter when calling the oauth/get_request_token method.

Request:

URL	oauth/get_request_token
Method	GET/POST

Parameters	**Description**
oauth_consumer_key	(required) Consumer Key. It is provided by T-PaaS in registration.
oauth_nonce	(required) Nonce. A random string.
oauth_signature_method	(required) The signature method should be "HMAC-SHA1" here.
oauth_signature	(required) Signature which is signed by the consumer secret that was issued to the application.
oauth_timestamp	(required) Timestamp. Current timestamp of the request
oauth_token	(required) Expired Access Token
oauth_version	(required) The oauth_version is 1.0

Response:

Parameters	**Description**
oauth_token	The Access Token that T-PaaS returns
oauth_token_secret	The secret associated with the Access Token.
oauth_expires_in	The lifetime of the request token in seconds.

5.1.5 OAuth Error Code

Error code	**Description**
400 Error	Bad Request. It might be due to absence of parameters, rejected OAuth version, invalid signature and expired token.
401 Error	Unauthorized. This can be caused by unverified OAuth signature and wrong OAuth tokens.
403 Error	Forbidden. This error happens when the application sends a request without user permission.
500 Error	Internal Error. This is generated within the healthcare T-PaaS.
501 Error	Unsupported Method. The application send a request with wrong method.

5.2 OpenID+OAuth API

The hybrid protocol, which is the composition of OpenID and OAuth, can enable user to approve login and service access at the same time.

Request:

URL	openid/oauth
Method	POST/GET

Parameters	**Description**
openid.mode	(required) Interaction mode. Valid values are “checkid_setup” (interaction allowed) and “checked_immediate” (No interaction allowed).
openid.ns	(required) Protocol version. This parameter should always be http://specs.openid.net/auth/2.0 for OpenID 2.0 requests.
openid.return_to	(required) Return URL. After signing in, the user is taken to this URL.
openid.assoc_handle	(optional) Association handle. Developer can set this association if the app and T-PaaS have established an association
openid.claimed_id	(optional) Claimed identifier. This should always be http://specs.openid.net/auth/2.0/identifier_select
openid.realm	(required) Authenticate realm. URL pattern of the domain that a user should trust
OAuth extension: If OAuth is required for service access	
openid.ns.oauth	(required) Indication of OAuth token. OAuth-specific parameter should always be: http://specs.openid.net/extensions/oauth/1.0
openid.oauth.consumer	(required) Consumer key. OAuth-specific parameter is the OAuth Consumer Key provided by T-PaaS in registration.
openid.oauth.scope	(optional) list of URLs identifying the services offered by T-PaaS that need to be accessed

Response:

Parameters	**Description**
openid.ns.oauth	The value is always: http://specs.openid.net/extensions/oauth/1.0
openid.oauth.request_token	Request Token. It is obtained from T-PaaS to start OAuth

After Request Token is obtained, it can be exchanged to an Access Token, which can be continued with Step 3. Figure 2 describes the basic flow of the hybrid protocol.

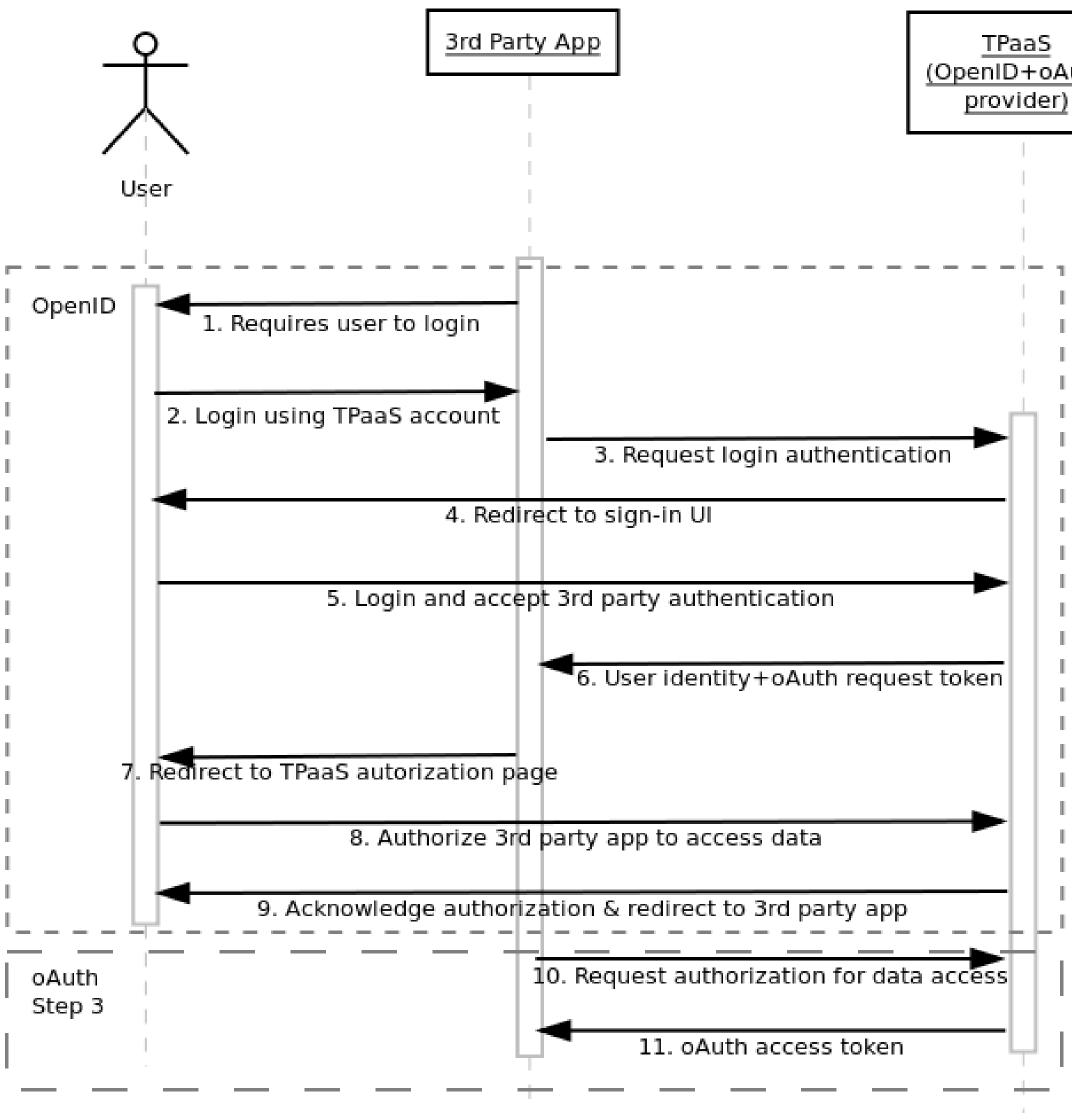

3rd Party App
TPaaS (OpenID+oAuth provider)
User
OpenID
1. Requires user to login
2. Login using TPaaS account
3. Request login authentication
4. Redirect to sign-in UI
5. Login and accept 3rd party authentication
6. User identity+oAuth request token
7. Redirect to TPaaS autorization page
8. Authorize 3rd party app to access data
9. Acknowledge authorization & redirect to 3rd party app
oAuth Step 3
10. Request authorization for data access
11. oAuth access token

Figure 9 - Flow of OpenID+OAuth Hybrid Protocol

5.3 CRUD operation via HTTPRest

5.3.1 *Resource overview and behavior*

The following table shows all the API resources available from the platform to access users' data. Developers can use a list of RESTful web services to perform basic CRUD (Create Read Update Delete) operations over the data. They all work only with a valid OAuth tokens/secrets, thus the entire following API will not be accessible with the wrong credentials.

Resource/HTTP method	POST (create)	GET (read)	PUT (update)	DELETE (delete)
/users/{access_token}/{data_type} Rationale: it has to be considered as a resource from which you can make operations on a set of data of type data_type.	**Create new data of type data_type**	**Show list of data of type data_type**	**If exists update data.** **If not, error**	**Delete data**

5.3.2 Use of OAuth with T-PaaS identity principles

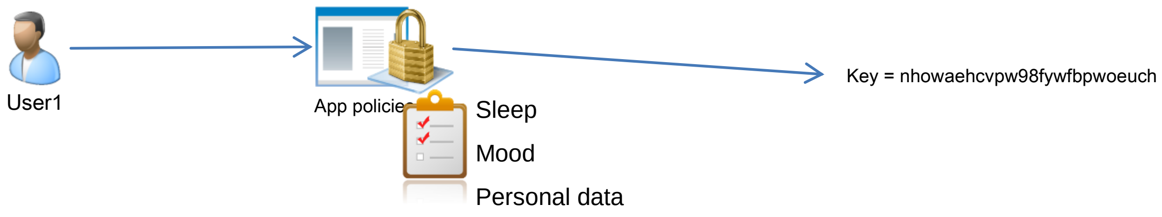
In order to understand properly how every single API works, it is important to underline some principles that have been adopted in building the T-PaaS platform.

More specifically, OAuth is considered as a tool to allow application to see data on behalf to a specific user (that has previously set her privacy policy details). For better understanding, the following examples show the mechanism that an app needs to adopt in order to access to a certain user's data.

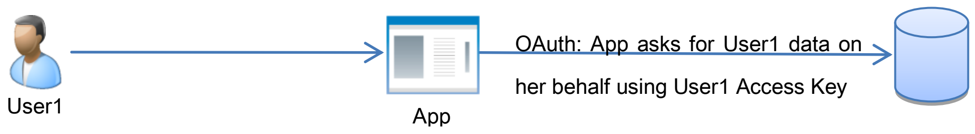
5.3.2.1 The application grants the rights

In order to let an application access to a certain user's data, these are the needed steps:

- The app developer registers its app and the platform provides the consumer key and secrets
- The user selects the App to access her data and accepts the minimum policy requirements in order the app to work, the system will produce the request token that the app has to use in combination with the consumer's keys/secrets
- The system grants to the App an access token that can be used as a key to access patient's data



5.3.2.2 Users access to her data via an application

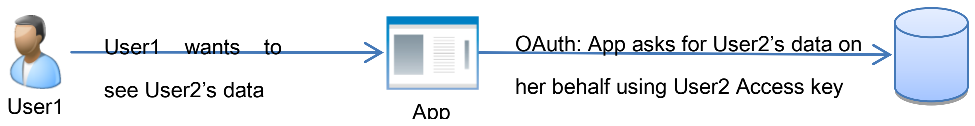

User1
App
OAuth: App asks for User1 data on her behalf using User1 Access Key

This is the normal flow, in which the application asks for User1's data and User1 is the actual user using the app.

This flows involes that:

- User1 has expressed her privacy policy about how the App has to access her data
- As soon as User1 asks to the App to see her data, the application uses the access token of User1 to connect to the platform.

5.3.2.3 User1 access to User2's data via an application


User1
User1 wants to see User2's data
App
OAuth: App asks for User2's data on her behalf using User2 Access key

This flow involves that:

- User1 has expressed her privacy policy about how the App has to access her data

- User2 has expressed her privacy policy about how the App has to access her data
- It exists the relation: User2→User1 that allows User1 to access User2's data (according to certain policies)
- As soon as User1 asks to the App to see User2's data, the application uses the access token of User2 to connect to the platform.

5.3.3 APIs

The methods to retrieve the data will have the following structure:

/users/{access_token}/{data_type}

It has to be considered as a resource from which you can make operations on a set of data of type data_type.

5.3.3.1 GET

Show list of data of type data_type. Search filter data are passed through the URI as well to allow the addressability of each single data.

URL:	users/{access_token}/{data_type}{?asker_access_token, start_datetime, end_datetime, serial_num, id_data}
Method:	GET
Auth:	OAuth credentials (e.g. consumer_key, access_token, nonce, …)

Parameters:	`asker_access_token`: Username of who is asking the request (same as {access_token} if it's a user watching her data)
	`start_datetime`
	`end_datetime`
	`serial_num`
	`id_data`
	the parameters has to respect the following RegEx:
	`(asker_username, ((start_datetime, end_datetime, serial_num?)|id_data+))`

Returns:	The output is an XML, which structure is described with the following DTD:
	If there are no errors it returns the data contained between start_datetime and end_datetime (with a specific serial_num) or the data correspondent to the specific id_data
	`<?xml version="1.0"?>`
	`<!DOCTYPE T-PaaS_response [`
	    `<!ELEMENT T-PaaS_response (data*|error+)>`
	    `<!ELEMENT error (id_error, error_description)>`
	    `<!ELEMENT data (id_data, data_type, unit_of_measure, serial_num, value, start_datetime, end_datetime, note)>`

	`<!ELEMENT id_data (#PCDATA)>`
	`<!ELEMENT data_type (#PCDATA)>`
	`<!ELEMENT unit_of_measure (#PCDATA)>`
	`<!ELEMENT serial_num (#PCDATA)>`
	`<!ELEMENT value (#PCDATA)>`
	`<!ELEMENT start_datetime (#PCDATA)>`
	`<!ELEMENT end_datetime (#PCDATA)>`
	`<!ELEMENT note (#PCDATA)>`
	`<!ELEMENT id_error (#PCDATA)>`
	`<!ELEMENT error_description (#PCDATA)>`
	`]>`

HTTP errors:	- 404 - not found: username not found - 404 - not found: data_type not found - OAuth errors
Other return errors:	- Invalid asker_username – `ERR_INVALID_ASKER_USERNAME` - Invalid start_datetime – `ERR_INVALID_START_DATETIME` - Missing start_datetime – `ERR_MISSING_START_DATETIME` - Invalid end_datetime – `ERR_INVALID_END_DATETIME` - Missing end_datetime – `ERR_MISSING_END_DATETIME` - App not allowed to retrieve the data due to policy restrictions – `ERR_APP_NOT_ALLOWED` - {username} not allowed to retrieve the data due to policy restrictions – `ERR_USER_NOT_ALLOWED` - {username} has no relation with the user that is asking the request – `ERR_USER_NOT_RELATED` - App has no relation with the user that is asking the request – `ERR_APP_NOT_RELATED`

5.3.3.2 POST

Create new data of type data_type

URL:	users/{username}/{data_type}
Method:	POST
Auth:	OAuth credentials (e.g. consumer_key, access_token, nonce, …)

Parameters (body):	Request DTD:
	`<?xml version="1.0"?>`
	`<!DOCTYPE T-PaaS_request [`

```
	<!ELEMENT T-PaaS_request (asker_access_token, data+)>
	<!ELEMENT asker_access_token (#PCDATA)>
	<!ELEMENT data (unit_of_measure, serial_num?, value,
	start_datetime, end_datetime, note?)>
	<!ELEMENT unit_of_measure (#PCDATA)>
	<!ELEMENT serial_num (#PCDATA)>
	<!ELEMENT value (#PCDATA)>
	<!ELEMENT start_datetime (#PCDATA)>
	<!ELEMENT end_datetime (#PCDATA)>
	<!ELEMENT note (#PCDATA)>
]>
```

Returns:

This service returns an XML, which is described with the following DTD:

```
<?xml version="1.0"?>
<!DOCTYPE T-PaaS_response [
	<!ELEMENT T-PaaS_response (data+)>
	<!ELEMENT data (id_data|error+)>
	<!ELEMENT error (id_error, error_description)>
	<!ELEMENT id_error (#PCDATA)>
	<!ELEMENT error_description (#PCDATA)>
]>
```

HTTP errors:

- 404 - not found: username not found
- 404 - not found: data_type not found
- OAuth errors

Other return errors:

- Invalid asker_username – `ERR_INVALID_ASKER_USERNAME`
- Invalid unit of measure (for the given data_type) – `ERR_INVALID_UNIT_OF_MEASURE`
- Invalid value (for the given data type/ unit of measure) – `ERR_INVALID_VALUE`
- Invalid start_datetime – `ERR_INVALID_START_DATETIME`
- Invalid end_datetime – `ERR_INVALID_END_DATETIME`
- Missing start_datetime – `ERR_MISSING_START_DATETIME`
- Missing end_datetime – `ERR_MISSING_END_DATETIME`

- App not allowed to retrieve the data due to policy restrictions – `ERR_APP_NOT_ALLOWED`
- {username} not allowed to retrieve the data due to policy restrictions – `ERR_USER_NOT_ALLOWED`
- {username} has no relation with the user that is asking the request – `ERR_USER_NOT_RELATED`
- App has no relation with the user that is asking the request – `ERR_APP_NOT_RELATED`

5.3.3.3 PUT

If exists update data. If not, error

URL:	users/{username}/{data_type}
Method:	PUT
Auth:	OAuth credentials (e.g. consumer_key, access_token, nonce, …)

Parameters (body):	`<?xml version="1.0"?>`
	`<!DOCTYPE T-PaaS_request [`
	`<!ELEMENT T-PaaS_request (asker_access_token, data+)>`
	`<!ELEMENT data (id_data, unit_of_measure?, serial_num?, value?, start_datetime?, end_datetime?, note?)>`
	`<!ELEMENT asker_access_token (#PCDATA)>`
	`<!ELEMENT id_data (#PCDATA)>`
	`<!ELEMENT unit_of_measure (#PCDATA)>`
	`<!ELEMENT serial_num (#PCDATA)>`
	`<!ELEMENT value (#PCDATA)>`
	`<!ELEMENT start_datetime (#PCDATA)>`
	`<!ELEMENT end_datetime (#PCDATA)>`
	`<!ELEMENT note (#PCDATA)>`
	`]>`

Returns:	The output is an XML, which structure is described with the following DTD:
	`<?xml version="1.0"?>`
	`<!DOCTYPE T-PaaS_response [`
	`<!ELEMENT T-PaaS_response (data+)>`
	`<!ELEMENT data (id_data|error+)>`
	`<!ELEMENT error (id_error, error_description)>`
	`<!ELEMENT id_error (#PCDATA)>`
	`<!ELEMENT error_description (#PCDATA)>`
	`]>`

HTTP errors:	- 404 - not found: username not found - 404 - not found: data_type not found - 404 - not found: id_data not found - OAuth errors -
Other return errors:	- Invalid asker_username – `ERR_INVALID_ASKER_USERNAME` - Invalid unit of measure (for the given data_type) – `ERR_INVALID_UNIT_OF_MEASURE`

	- Invalid value (for the given data type/ unit of measure) – `ERR_INVALID_VALUE` - Invalid start_datetime – `ERR_INVALID_START_DATETIME` - Invalid end_datetime – `ERR_INVALID_END_DATETIME` - Invalid serial_num – `ERR_INVALID_SERIAL_NUM` - App not allowed to update the data due to policy restrictions – `ERR_APP_NOT_ALLOWED` - {username} not allowed to update the data due to policy restrictions – `ERR_USER_NOT_ALLOWED` - {username} has no relation with the user that is asking the request – `ERR_USER_NOT_RELATED` - App has no relation with the user that is asking the request – `ERR_APP_NOT_RELATED`

5.3.3.4 DELETE

Delete data

URL:	users/{username}/{data_type}.format{?asker_access_token, id_data, start_datetime, end_datetime, serial_num}
Method:	DELETE
Auth:	OAuth credentials (e.g. consumer_key, access_token, nonce, …)

Parameters:	`asker_access_token`: Username of who is asking the request (same as {access_token} if it's a user watching her data)
	`start_datetime`
	`end_datetime`
	`serial_num`
	`id_data`
	the parameters has to respect the following RegEx:
	`(asker_username, ((start_datetime, end_datetime, serial_num?)|id_data+))`

Returns:	The output is an XML, which structure is described with the following DTD:
	`<?xml version="1.0"?>`
	`<!DOCTYPE T-PaaS_response [`
	`    <!ELEMENT T-PaaS_response (error*)>`
	`    <!ELEMENT error (id_error, error_description)>`
	`    <!ELEMENT id_error (#PCDATA)>`
	`    <!ELEMENT error_description (#PCDATA)>`
	`]>`

HTTP errors:	- 404 - not found: username not found - 404 - not found: data_type not found - 404 - not found: id_data not found - OAuth errors -
Other return errors:	- Invalid asker_username – `ERR_INVALID_ASKER_USERNAME` - App not allowed to delete the data due to policy restrictions – `ERR_APP_NOT_ALLOWED` - {username} not allowed to delete the data due to policy restrictions – `ERR_USER_NOT_ALLOWED` - {username} has no relation with the user that is asking the request – `ERR_USER_NOT_RELATED` - App has no relation with the user that is asking the request – `ERR_APP_NOT_RELATED`

Chapter 6

Overview of relevant TClouds security components

Chapter Authors:

Mina Deng (PHI), Paolo Smiraglia, Davide Vernizzi (POL), Klaus Stengel (FAU), Johannes Behl (FAU), Stefan Nürnberger (TUDA), Norbert Schirmer (SRX), Imad M. Abbadi (OXFD), Sören Bleikertz (IBM)

6.1 Log service

6.1.1 Main functionalities

The main focus of the Log Service[3] is to log and track events originated at the Cloud infrastructure layer, such as creation, destruction or migration of a Virtual Machine (VM), or allocation and deletion of a bucket of storage. However, this component could also be used by applications deployed on TClouds to log and audit events happen in the application layer. For instance, Log Service can be used as basis for auditing or reporting the Service Level Agreement (SLA) compliance to the User. This is, the User paid for some exclusive access to the physical nodes and verifies that such a SLA policy is in compliance by accessing the log that lists the VMs running on these nodes.

6.1.2 Security features useful for WP3.1

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. This Log Service is mainly based on secure logging schemes, such as the ones proposed by Schneier and Kelsey (Bruce Schneier, 1999), and Ma and Tsudik (Di Ma, 2009). It provides most of the security features required by a logging system, and namely integrity, privacy and access control of log entries. Moreover, Log Service relies on a resilient storage4 to guarantee availability of logs, also for long periods of time.

6.1.3 Usage of the component in WP3.1

Primarily, the Log Service from A2 provides the logging for events from the IaaS layer. A3 would extend this logging feature into the platform and application layer, in order to provide accountability and auditability for the healthcare applications deployed on TClouds. Actors from A3 act as consumers of the cloud and therefore can access the Log Service. For instance, an A3 healthcare administrator may want to check if his application deployment complies with the SLA and the patient's consent, using the Management Console provided

[3] Chapter 6 Log Service, TClouds deliverable D2.1.1 Technical Requirements and Architecture for Privacy enhanced and Resilient Trusted Clouds

[4] Chapter 6 Object Storage, TClouds deliverable D2.2.1 Preliminary Architecture for Middleware for Adaptive Resilience

by TClouds. Moreover, developers of A3 applications could take advantages from Log Service to create secure log entries without having to setup own log system.

6.1.4 Current status

The current API is only intended to be used by IaaS and PaaS Cloud Components (i.e., by A2 components), but this can be extended to be used by applications as well. At the moment, the Log Service only implements Symmetric key cryptographic scheme (Bruce Schneier, 1999) which provides most of the security features required by a logging system, but it lacks the protection against particular attacks, namely truncation attack and delayed deletion attack (Di Ma, 2009).

6.2 Tailored Cloud Services (Tailored memcached)

6.2.1 Main functionality

The memcached[5] 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.

The cached data is mainly organized using a flat namespace, where each entity has a unique name (called key) assigned by the user of the service. While keys are restricted to 250 characters and must be in plain-text format, there are no such limitations imposed on the actual data (called values) stored in the cache. Additionally, memcached offers a "flags" field, which allows the user to store 16 bits (32 bits in newer versions) of auxiliary data together with the actual cached data value.

When storing or updating an entry, it is also possible to specify an absolute or relative expiration time, after which it will be removed automatically. In order to track changes and allow automatic updates, the service maintains a version counter for each entry that can be queried but not manipulated.

The memcached offers usual functions for storing, retrieving and updating entries given their secret key. Three different mechanisms are responsible for removing data from the cache: Entries can be deleted on request or automatically. Such automatic deletions happen either by the expiration mechanism mentioned in the previous paragraph or if the cache runs out of memory. To determine which entries to remove for the latter case, the service maintains internal usage statistics and selects the least frequently accessed items.

6.2.2 Security features useful for WP3.1

The service is designed to improve availability and performance by utilizing type-safety in the programming language and a custom tailoring process to remove unnecessary features based on the deployment scenario. In combination with a minimal operating system layer,

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

the trusted computing base (TCB) of the final component is considerably smaller than in a traditional deployment. Encryption and authentication functions could be added in order to provide additional layers of security compared to the original memcached implementation. With the original memcached, anyone who can reach the service on the network may read/alter cached data, which means there is currently no support for authentication or encryption. For this reason, these security functions might be some useful configurable extensions for the service, such as by using passwords to protect the cached data.

Regarding the differences between the proposed components comparing with the one from the original memcached[6], the external protocol will be basically. However, the difference is that there is no need to use a complete operating system to run it and a safer programming language to implement it. Therefore, it is much more unlikely that the proposed implementation will have security critical issues like buffer overflows, which would allow information to leak. The original implementation had such a bug at least once.

6.2.3 Usage of the component in WP3.1

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. An example is used to present a typical usage pattern for the memcached. Assume a function to create a dynamic section of a website with average statistics over the last month. The generation requires some expensive database queries. To improve performance when this page is accessed frequently (e.g. a patient's personalized start page), it is possible to feed the results into a memcached instance. Dedicated Java code might look like this:

```
String getUserStatistics(String user_id) {
      String result;
      Float avg_doses_taken, avg_sleep;
      avg_sleep = querySqlDatabaseSleep(user_id, LAST_MONTH);
      avg_doses_take = querySqlDataBaseDoses(user_id, LAST_MONTH);
      result = "<div id=\"sleep\">Average Hours of sleep: "
            + avg_sleep + "</div>"
            + "<div id=\"doses\">Average number of doses"
            "taken:" + avg_doses_taken + "</div>";
      return result;
}
```

Next one can add a simple query to the cache service and check if the data was already calculated recently and use that information instead:

```
String getUserStatistics(String user_id) {
      String result = queryMemcache("statistics-" + user_id);
      if (result != NULL) return result;
      /* else not cached yet... */

      Float avg_doses_taken, avg_sleep;
      avg_sleep = querySqlDatabaseSleep(user_id, LAST_MONTH);
      avg_doses_take = querySqlDataBaseDoses(user_id, LAST_MONTH);
```

[6] Memcached: http://memcached.org/

```
      result = "<div id=\"sleep\">Average Hours of sleep: "
            + avg_sleep + "</div>"
            + "<div id=\"doses\">Average number of doses"
            "taken:" + avg_doses_taken + "</div>";

      saveMemcached("statistics-" + user_id, result, 12*HOURS);
      return result;
}
```

6.3 Resilient BPEL

6.3.1 Main Functionalities

RBPEL[7] is a platform for the fault-tolerant execution of Web-service-based workflows particularly within clouds. These workflows have to be described in a special language, namely the Web Services Business Process Execution Language (WS-BPEL or short BPEL). BPEL is XML-based and allows specifying Web services that are composed of other Web services. These special Web services are also called composite Web services and their corresponding BPEL descriptions are called process definitions. Within RBPEL, engines responsible for the execution of such composite Web services are replicated actively. The same holds for the Web services the composite Web services are based on. 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. RBPEL is only about tolerating crash failures but not tolerating Byzantine that is arbitrary failures. The current RBPEL infrastructure is able to tolerate crashes of a certain number of its components, but it isn't able to detect or mask, for instance, intrusions or bit flips.

6.3.2 Dependability and Security Features

In its current form, RBPEL uses active replication in order to tolerate web service crashes and for providing a highly available Web services from the TClouds healthcare platform.

6.3.3 Usage of the component in WP3.1

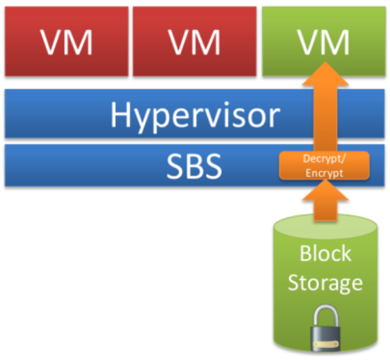
RPBEL can be used whenever highly available composite Web services are needed. If there are multiple Web services that need to be coordinated, orchestrated or combined in whatever

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

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. If traditional Web services were used for the communication between the different components (Management Application, Patient Application, Paten PC etc.) instead of RESTful ones, there could be the chance to identify workflows that can be modelled as composite Web services.

6.4 Secure Block Storage (SBS)

6.4.1 Main Functionalities



Secure Block Storage (SBS)[8] will provide a transparent layer that provides security properties, such as confidentiality, integrity, and authenticity for block devices. Block storage is non-linear raw memory attached to VM instances as block device (virtual hard disk, e.g. iSCSI). The SBS is also responsible for user-centric key management.

The focus is on an open source cloud, because SBS cannot influence either the storage backend or VM images deployed to a commodity public cloud. The latter solution furthermore has the advantage, that legacy VMs (i.e. VMs not aware of security objectives) can be used, as the modified hypervisor then functions as a translation layer between ciphertext and plaintext.

6.4.2 Security features useful for WP3.1

The SBS is able to provide security properties including confidentiality, integrity and authenticity, and version control and replay attacks 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 rest 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.

[8] Chapter 7.3 Secure Block Storage (SBS), TClouds deliverable D2.4.1 TClouds Prototype Architecture, Quality Assurance Guidelines, Test Methodology and Draft API

6.4.3 Usage of the component in WP3.1

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.

6.5 Trusted virtual domain (TVD)

6.5.1 Main Functionalities

A TVD[9] is a set of compartments that trust each other, share a common security policy and enforce it independently of the particular physical hardware platform they are running on. Moreover the TVD infrastructure contains a hypervisor, also called VMM, and physical components, such as CPU, memory, and hardware security modules, on which the compartments rely to enforce the policy.

Major strength of TVDs is the transparent data protection and enforcement of access control policies. The TVD ensure that platforms and users logically assigned to the same TVD can access distributed data storage, network services, and remote servers without executing any additional security protocols, while the resources belonging to different TVDs are strictly separated. Those resources remain inaccessible for unauthorized participants. Moreover, data that is stored on mobile storage devices is automatically protected by encryption and can only be decrypted within the same TVD the device has been assigned to. Hence, users cannot forget to employ encryption, and data on mobile storage devices such as memory sticks cannot be used outside the TVD.

6.5.2 Security features useful for WP3.1

The relevance to the WP3.1 medical use case is straightforward. The main features of the TVD infrastructure are:

- Isolation of tenants: Isolation of stakeholders within the TClouds healthcare applications (such as HSR and PHI) and enforcement of information flow rules. Containment boundaries to compartments from different TVDs are provided by the underlying secure hypervisor (Security Kernel and VMM), allowing an isolated execution of several different TVDs on the same physical hardware platform.
- Secure communication channels: Compartments belonging to the same TVD are connected through a virtual network that can span over different platforms and that is strictly isolated from the virtual networks of other TVDs.
- Information flow control: information can only flow between compartments belonging to the same TVD. This is defined by the information flow policies.
- Central management: as the main feature of the Trusted Infrastructure, different TVDs, their infrastructure, trust relationships, and security policies are centrally manageable.

[9] Chapter 12.5 Trusted Virtual Domains, TClouds deliverable D2.1.1 – Technical Requirements and Architecture for Privacy enhanced and Resilient Trusted Clouds

Centralized infrastructure management allows registration and authentication of all hardware platforms, security services and VMs.

6.5.3 Usage of the component in WP3.1

For year 2 there are no requirements to WP3.1 as we will deploy the complete application (possibly 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.

6.6 Resilient Object Storage

6.6.1 Main Functionalities

Resilient Object Storage[10] 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 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.

6.6.2 Security features useful for WP3.1

The storage system provides confidentiality through encryption, integrity through cryptographic data authentication, and reliability through data replication and erasure coding. Key management for encryption and authentication keys is integrated.

- Availability is provided via exploiting replication and diversity to store the data on several clouds, it allows access to the data as long as a subset (generally, a large enough majority) of them is reachable.
- Integrity is ensured that data can be retrieved correctly even if some of the clouds corrupt data, lose it, or adversarial manipulate it. The system builds on so-called Byzantine fault-tolerant replication that stores data on several providers.

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

- Confidentiality of the stored data against disclosure to one or more cloud providers is ensured through encryption. The system may use a novel secret sharing scheme, whereby encryption keys are maintained collaboratively by a (sufficiently large) majority of the cloud providers. No (or sufficiently small) faulty minority can learn anything about the stored data, not even by colluding.

6.6.3 Usage of the component in WP3.1

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 HSR) using the untrustworthy clouds as coordination media.

6.7 Access Control as a Service (ACaaS)

6.7.1 Main Functionalities

ACaaS[11] 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).

6.7.2 Security features useful for WP3.1

Security feature that ACaaS can provide is to consider 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.

A number of assumptions hold for this component. Although, we do not assume that all cloud employees are trusted, we assume that cloud providers trust a set of employees who interact and manage the ACaaS. Moving physical Computing Node across physical locations is controlled by the set of trusted cloud Admin who are trusted to reflect such movement in the ACaaS. The hardware of Computing Node is secure and trusted, incorporating a Trusted Platform Module chip (TPM).

[11] 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

6.7.3 Usage of the component in WP3.1

Managing clouds' hosting environment 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 will take 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.

6.8 Security Assurance for Virtual Environment (SAVE)

6.8.1 Main Functionalities

SAVE (Security Assurance for Virtual Environment)[12] 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 an isolation breach diagnosis.

6.8.2 Security features useful for WP3.1

This automated audit mechanism is able to validate that isolation of different tenants in the cloud infrastructure is given by analyzing the current configuration. The assumption is that the automated validation requires access to the configuration information on the physical nodes or through a central management interface. The OpenStack infrastructure could be extended and used to extract the information.

6.8.3 Usage of the component in WP3.1

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

[12] 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

Chapter 7

Conclusions

Chapter Authors:

Mina Deng (PHI), Ya Liu (PHI)

In this deliverable we introduce Health T-PaaS, a multilevel healthcare platform, with its overview concept, utility, architecture and value added security components as well as its legislation analysis.

Different from traditional web-based and platform-based personal healthcare services in the market, Health T-PaaS applies secure and resilient cloud infrastructure for computing capability, secure storage, and security/privacy management, on top of which an open platform generally provides identity management, access control as a service, log as a service for users and developers.

The use cases of the health trustworthy PaaS described. The document highlights the actors involved in the system and their relationships in various scenarios, which include both SaaS and PaaS use cases. In terms of SaaS oriented scenarios, it offers user registration, data operation by user, user relationship definition and account deletion. In order to provide PaaS functionality, PaaS-based scenarios are described, such as application's privacy policy profile specification, data access auditing, application registration and PaaS/IaaS level monitoring.

The high level RESTful APIs provided, intended to describe the possibility and flexibility of 3rd party integration with Health T-PaaS. By applying OAuth and OpenID application side trust protocols, user authentication and application authentication and authorization can be achieved in a secure and trust way with high compatibility.

The benefits of adopting Health T-PaaS are largely based on applying the trustworthy cloud environment, the feature of which can be easily inherited. For instance, log service is to log and audit events happen in the both infrastructure and application layers, RBPEL can provide a fault-tolerant execution of Web-service-based workflows particularly within clouds and TVD ensures that platforms and users logically assigned to the same domain can access distributed data storage, network services, and remote servers.

we discuss the integration between A2 and A3 by proposing a list of A2 security components to be embedded within the Health T-PaaS. These security components include Log Service, Tailored Cloud Services (Tailored memcached), Resilient BPEL (RPEL), Secure Block Storage (SBS), Trusted Virtual Domain (TVD), Resilient Object Storage, Access Control as a Service (ACaaS), and Security Assurance tool for Virtual Environment (SAVE).

The outcome of this work is the fundamental of our future plan in establishing and implementing a healthcare trustworthy platform in TClouds. Our next objective is to implement its functionalities above the TClouds proposed infrastructure, starting from the APIs described in Chapter 5.

Chapter 8

List of Abbreviations

ACaaS	**Access Control as a Service**
API	Application programming interface
App	Applications
CRUD	create, read, update, delete operations
IaaS	Infrastructure as a Service
OAuth	open standard for authorization
PHR	Personal Health Record
RBEPL	Resilient BPEL
SaaS	Software as a Service
SAVE	Security Assurance for Virtual Environment
SBS	Secure Block Storage
T-PaaS	Trusted Platform as a Service
TPSC	The Patient Safety Company
TVD	Trusted Virtual Domain

Chapter 9 Bibliography

Bruce Schneier, J. K. (1999). Secure Audit Logs to Support Computer Forensics. *ACM Trans. Inf. Syst. Secur. 2(2)* , 159-176.

Di Ma, G. T. (2009). A new approach to secure logging. *ACM Transactions on Storage, Volume 5 (1)* .

Sean, N. (2010, February 19). *HealthVault Application Integration Recommendations.* Retrieved from Microsoft MSDN: http://msdn.microsoft.com/en-us/library/ff803594.aspx

Solutions, M. H. (2011, October). *Microsoft HealthVault Account Privacy Statement*. Retrieved from Microsoft HealthVault: https://account.healthvault.com/help.aspx?topicid=PrivacyPolicy