MULTICLOUD APPLICATIONS TOWARDS THE DIGITAL SINGLE MARKET

Deliverable D5.2

Initial Advanced Cloud Service meta-Intermediator

Editor(s):	Marisa Escalante
Responsible Partner:	TECNALIA
Status-Version:	Final - v1.0
Date:	30/11/2017
Distribution level (CO, PU):	CO

Project Number:	GA 726755
Project Title:	DECIDE

Title of Deliverable:	D5.2 – Initial Advanced Cloud Service meta-Intermediator
Due Date of Delivery to the EC:	30/11/2017

Workpackage responsible for the Deliverable:	WP5 – Continuous cloud services mediation
Editor(s):	TECNALIA
Contributor(s):	Anna Shevchenko, Andrey Sereda (CB) Maria Jose Lopez, Marisa Escalante Gorka Benguria Leire Orue-Echevarria, Juncal Alonso, Iñaki Etxaniz (TECNALIA)
Reviewer(s):	Miguel Ángel Pérez; ARSYS
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP3, WP4, WP6

Abstract:	This deliverable contains the initial version of the implementation of the Advanced Cloud Service meta-Intermediator (ACSmI)). This deliverable is the result of T5.1 – T5.5. The software will be accompanied by a Technical Specification Report
Keyword List:	Broker; Services Discovery; Services Contracting,CSP.
Licensing information:	It is released under Apache 2 Licence. The document itself is delivered as a description for the European Commission about the released software, so it is not public.
Disclaimer	This deliverable reflects only the author's views and the Commission is not responsible for any use that may be made of the information contained therein.

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.0	06/09/2017	ToC	TECNALIA
v0.1	18/10/2017	General sections completed	TECNALIA
v0.2	02/11/2017	Changes in the structure and section 1, 2 and 6 completed	TECNALIA
v0.3	07/11/2017	Section 5 completed	TECNALIA
v0.4	08/11/2017	Section 4 updated	CB
v0.5	16/11/2017	Section 3 updated and general changes	TECNALIA
v0.6	17/11/2017	Changes in the description of the packages in section 3	TECNALIA
v0.7	21/11/2017	Internal review of the document	ARSYS
v0.8	22/11/2017	Changes implementing the suggestions of the internal review	TECNALIA
V0.9	23/11/2017	Section 4 and adjustments in the other section	TECNALIA, CB
V1.0	26/11/2017	Ready for submission	TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

ACSmI	Advanced Cloud Service meta Intermediator
API	Application Programming Interface
CRUD	Create, Read, Update and Delete
CSP	Cloud Service Provider
DB	Database
EC	European Commission
NFR	Non-Functional Requirement
SLA	Service Level Agreement
UI	User Interface

Executive Summary

This document describes the M12 prototype of ACSmI. The general objective of this M12 prototype is to discover services based on a set of requirements, to select the metrics to be measured based on the NFR and SLAs and to contract the services selected. This is realized by two main components: ACSmI discovery and ACSmI contracting and complemented with the first design of ACSmI monitoring.

The objective of ACSmI discovery is to discover services and also to allow the endorsement of cloud services to the service registry by CSPs. The objective of ACSmI contracting is the execution and management of the core functions with respect to the service contracts. In addition to these two main components, the first design (Functional and Technical) of the ACSmI monitoring component is detailed.

The document presents the mission, scope, functional description, technical approach, download and installation instructions as well as user manuals of the components comprising the prototype.

This document will be updated in each release of ACSmI, which are due in M24 and M30, including the new functionalities implemented in each case.

1 Introduction

1.1 About this deliverable

This document is the complement to the delivered software as prototype in the specified date and deliverable name at the head of the document.

1.2 Document structure

This document describes the prototypes D5.2 addressing the implementation and usage details of the ACSmI Discovery, ACSmI contracting and ACSmI monitoring prototypes. It is divided into four main sections describing the general purpose of ACSmI and the three components of the prototype. Architectural and implementation details of the components and how to use and download them are described in detail in the following sections.

The structure of the document is as follows.

Section 2 Implementation

This section provides the general aspects of ACSmI and how this tool fits into the overall DECIDE architecture.

Section 3 ACSmI Discovery

This section introduces the functional (section 3.2) and technical (section 3.3) description of the ACSmI discovery prototype while sub-section 3.4 provides a description of the delivery and usage of the prototype, including installation and downloading instructions.

Section 4 ACSmI Contracting

This section introduces the functional (section 4.2) and technical (section 4.3) description of the ACSmI Contracting prototype while sub-section 4.4 provides a description of the delivery and usage of the prototype, including installation and downloading instructions.

Section 5 ACSmI Monitoring

The sub-section 5.1 details the metrics and parameters associated to each service type to measure the availability. In sub-section 5.2 a vision on how to implement this prototype is defined.

To finalize the deliverable, the conclusions are presented and an appendix is included. This appendix shows a table with the definition of the cloud service attributes considered.

2 Implementation

2.1 Functional description

The Advanced Cloud Service (meta-) intermediator (ACSmI) aims to provide the means for the discovery, contracting, managing and monitoring of different cloud service offerings. ACSmI will provide ways to assess continuously the fulfilment of non-functional properties of cloud service offerings while enforcing the legislation compliance. ACSmI will also provide means to allow the endorsement of service offerings from different CSPs.

These are the main functionalities envisioned:

1. **Endorse a cloud service** into the ACSmI. ACSmI will allow to register services. This can be done either by the CSP itself, by the multi-cloud application operator or by the ACSmI Administrator. The register of each service will cover the different terms defined for the modelling of the CSPs and their service offerings. This will allow the discovery of services from the service registry.
2. **Discover** and benchmark **services**. OPTIMUS will indicate the NFR of the services that shall be delivered to the ACSmI as input. ACSmI will discover, from the services stored in its registry, the most appropriate ones for that set of NFRs. Then, from the set of discovered services, ACSmI will prioritize these services in terms of NFRs fulfilment (including legal aspects) which will be passed onto OPTIMUS as a short list. This short list will include, additionally, the degree of fulfilment of the NFRs requested by the user.
3. **Contract services**. This functionality will allow dealing with all the activities related to the contracts of ACSmI. Depending on the type of services and the CSP, ACSmI will manage the contract in two different ways: 1) ACSmI will facilitate the contracting of services directly to the provider by the user himself and 2) ACSmI will manage the contract itself acting as intermediator with the provider and the user. In this case, ACSmI will have mainly two types of contracts. The first one is the contract with the CSP while the other one is the contract with the user of the services intermediated by the ACSmI.
4. **Manage CSPs**. This functionality will allow the management of the different connectors to facilitate the contracting of the services and to monitor them. This functionality will be the one responsible to inform ADAPT and provide it with the required information for the deployment of the multi-cloud application on the different contracted services.
5. **Monitor NFR CSPs** and manage the violation alerts. This functionality will monitor the SLAs (NFRs) of the services offered by the CSPs to detect any violation of the SLAs. These metrics will be recorded and passed onto ADAPT monitoring during the operation of the services. If a violation is detected, an alert to the CSP will be sent.
6. **Monitor the use and bill the user**. This functionality will allow the calculation of the costs made by the user for the use of ACSmI services, and it will provide him with the corresponding receipt. To be able to generate the billing of the contracted services, the ACSmI shall monitor the use of the different cloud services.

ACSmI will be implemented following an incremental approach adding features in the different releases of the tool (D5.3, D5.4).

This first version of the prototype implements some features and covers some of the requirements identified for ACSmI in deliverable D5.1 “ACSmI requirements and technical design” [1].

Functionality that this prototype offers:

The delivered prototype is the first version of ACSmI contains the following functionality:

- Discovery of services stored in the service registry. The discovery is performed based on the requirements introduced in the UI. In next versions, the discovery will be carried out based on the requirements provided by OPTIMUS.
- Endorsement of services in the service registry. The fields to be completed are adapted depending on the type of service to be endorsed. This will be executed according to the cloud services modelling approach detailed in the deliverable D3.4 [2]
- Allowing the contracting of the selected cloud services. This contracting could be done either through the ACSmI or by the developer directly with the CSP. In any case, the information regarding the service needs to be passed to ACSmI for its monitoring at operation time.
- Definition of the metrics and parameters for each type of services.

Detailed information about which requirements exactly have been implemented for this prototype can be found in subsequent sections of this deliverable (cf.3.1, 4,1 and 5.1)

2.1.1 Fitting into overall DECIDE Architecture

ACSmI is one of the components of the DECIDE architecture. It is present in almost all steps of the operation of a service and it supports mainly two phases (Figure 1):

- Multi-cloud application (pre-)deployment, providing means for discovery and contracting the cloud services.
- Multi-cloud application operation, providing means for monitoring at run-time the cloud services where the multi-cloud application is deployed.

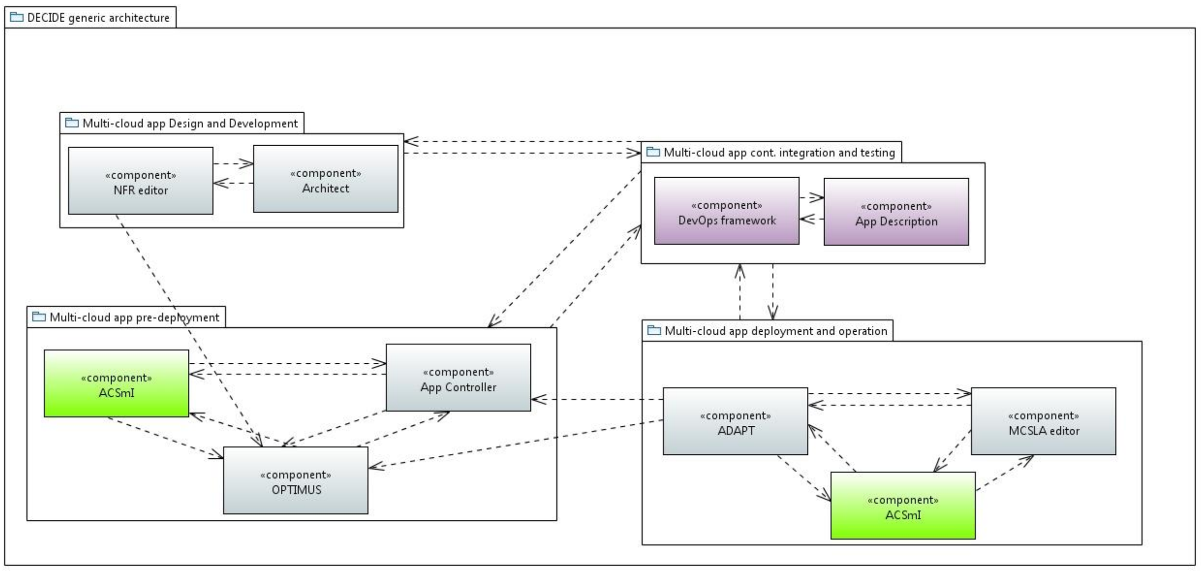

DECIDE generic architecture
Multi-cloud app Design and Development
«component» NFR editor
«component» Architect
Multi-cloud app cont. integration and testing
«component» DevOps framework
«component» App Description
Multi-cloud app pre-deployment
«component» ACSmI
«component» App Controller
«component» OPTIMUS
Multi-cloud app deployment and operation
«component» ADAPT
«component» MCSLA editor
«component» ACSmI

Figure 1. ACSmI in DECIDE architecture

ACSmI interacts with other tools in the DECIDE ecosystem (see Figure 2). To this respect, ACSmI will communicate with OPTIMUS to gather the non-functional requirements that the services that need to be discovered have to fulfil. ACSmI will return OPTIMUS the list of services available in the service registry fulfilling the entered requirements. ACSmI receive information from the Application Controller regarding the services that will have to be contracted and will return the Application Controller the final information about the contracted cloud services. Finally, ACSmI will interact with ADAPT with three main objectives:

1. to provide the information of the cloud services to allow ADAPT deploying the multi-cloud application,

2. to interchange information regarding the violation of the SLAs, and
3. to inform to ADAPT that a service is not available in the service registry due to a withdrawal process.

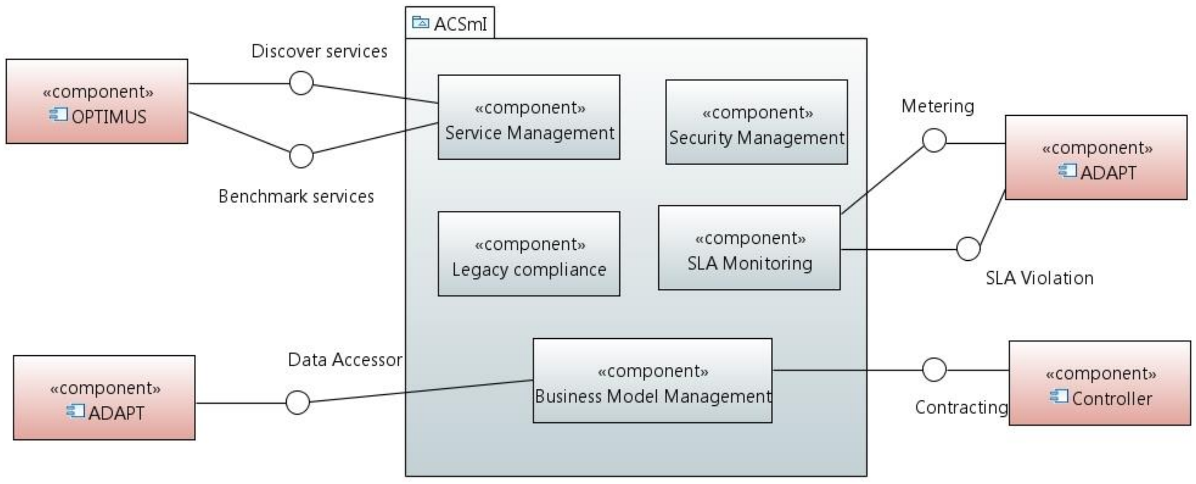


Figure 2. ACSmI Interfaces with other DECIDE tools

3 ACSmI Discovery

3.1 Functional description

ACSmI Discovery component, described in this section, covers the discovery and endorse functionalities, as well as, the modification and deletion of the services endorsed in the service registry.

Requirements covered by the prototype:

Table 1. Requirements covered by the M12 ACSmI Discovery prototype.

Req. ID	Req. Description	Requirement coverage by the prototype
WP5-DIS01	CSPs or the ACSmI administrator (for Large CSPs) register(s) one of its services or large CSP´s services in the service registry. The registry of each service shall cover the different terms defined in the modelling of the CSPs and their services. This will allow the discovery of the services from the registry.	The M12 prototype will allow endorsing services of four different types: • Storage • Database • Virtual Machines • Containers The prototype will request certain information, depending on the service type to the CSP. This information will allow the discovery of these services.
WP5-DIS02	The (non-functional) requirements of the multi-cloud application shall be collected by OPTIMUS and passed to the ACSmI so that services from the service registry fulfilling such requirements can be discovered. The requirements will be specified following the different terms	Requirements are collected from the provided UI instead from OPTIMUS.

Req. ID	Req. Description	Requirement coverage by the prototype
	defined for the modelling of the CSPs and their services. This allows an automatic comparison of the requirements with the services stored in the registry.	
WP5-DIS03	The objective is to provide a list of services from the services registry that fulfil (totally or partially) the requirements specified by the DECIDE operator. These requirements are specified in the DIS02.	The M12 prototype provides a UI that allows to detail the requirements required to the services, and based on these requirements, the list of services available in the service registry is shown.
WP5-DIS07	The objective is to provide means to create, read, update and delete the services registry. This registry shall record not only information provided by the CSPs, but also other information such as which multi-cloud application is using the service, SLAs violations, legal compliance and so on. The service registry management shall be aware of the alerts of potential SLA violations as well as non-legislation compliance (WP5-MON06 & WP5-LEG02) in order to update appropriately the registry.	Together with the requirement WP5-DIS01, this prototype allows changing the values of the attributes and deleting the services. This prototype does not cover the information related to the SLA´s violation and the legal compliance. These points will be covered in the next versions of the prototype.

3.2 Technical description

The discovery of the services will be performed based on common attributes for each service type.

Next figure shows the cloud service model that will allow the ACSmI service discovery.

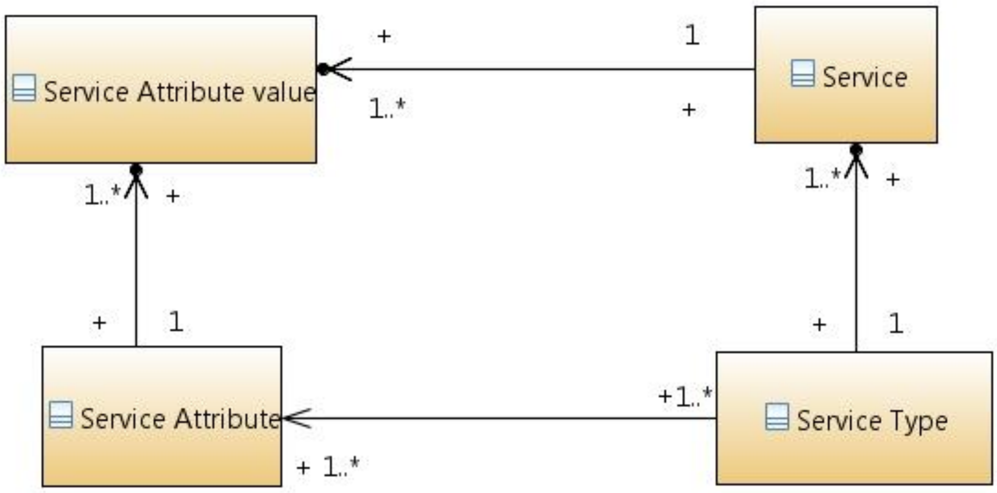


Figure 3. Cloud services model

As shown in the figure above, for this prototype, four service types have been defined with the corresponding service attributes, namely storage, database, virtual machine and container. The table below shows the relationship between the service type and the attributes. Any CSPs that want to endorse their cloud services in the service registry is required to provide this information.

The discovery functionality implemented presents the list of services that are in the registry and that cover all the requirements, which are expressed as attribute – value. The description of each attribute can be found on Appendix 1.

Table 2. Attributes Vs Cloud Service Types

Storage	Database	Virtual Machine	Container
Name	Name	Name	Name
Region	Region	Region	Region
Provider	Provider	Provider	Provider
Type	Type	CPU cores	Storage (GB per month)
Subtype	Subtype	Total Cores	Data transfer out (GB per month)
Capacity	Availability Zone	Hyper-threading	
Access type	Technology	MHz per core	
Data Redundancy	License	Memory	
	Instance Type		
	Size		
	CPU		

3.2.1 Prototype architecture

The current prototype (M12) of the discovery component is structured in four main components, as shown in the next figure.

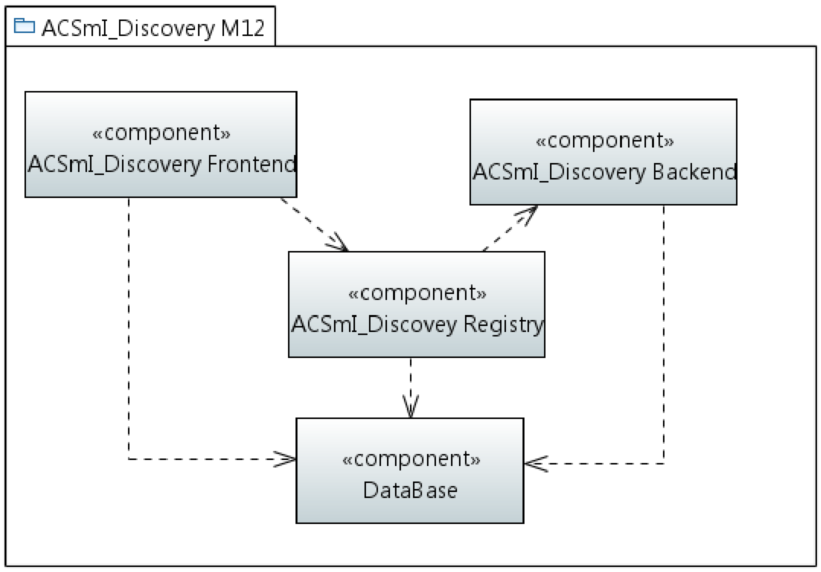

ACSmI_Discovery M12
«component» ACSmI_Discovery Frontend
«component» ACSmI_Discovery Backend
«component» ACSmI_Discovey Registry
«component» DataBase

Figure 4. ACSmI Discovery M12 High-Level Architecture

- *Frontend*. This component is in charge of managing all the aspects related to the ACSmI interface as well as to the user management. Five roles have been defined: Admin, User_Owner, User, CSP_Owner and CSP. Based on the role assigned to the user, different tasks to operate will be shown to him/her.

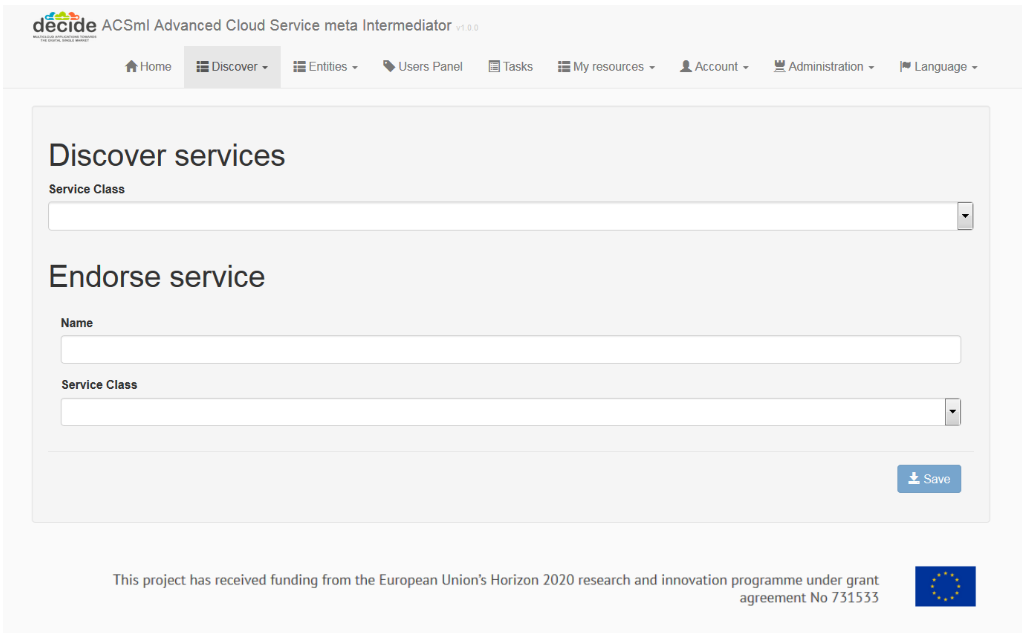
- *Backend*. This component is in charge of managing the aspects related with the discovery and the endorsement of the services. To carry out these activities, this component will manage the database to create, delete, modify the service types, services, CSPs, and so on.
- *Registry*. This component will coordinate the communication between the frontend and the backend.
- *Database*, which stores the services.

3.2.2 Components description

3.2.2.1 Frontend

This component has two main objectives:

1. To implement the graphical interface to allow introducing the requirements for the discovery of the services as well as to allow CSPs to introduce the information regarding their services.



Discover services

Service Class

Endorse service

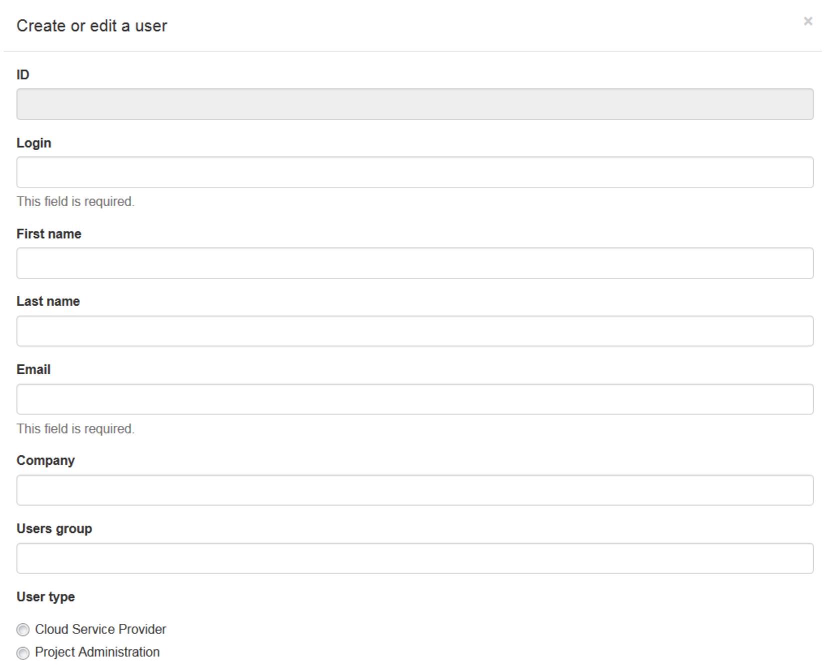
Name

Service Class

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731533

Figure 5. Discover and Endorse services functionalities – Admin view

2. To manage users. This component will enable to create, read, update and delete users in ACSmI discovery. At this moment, five different user roles for the user have been identified:
    - Admin. It is the responsible of the management of the ACSmI.
    - User_Owner. This role is the owner of the account. It is able to create users with the role of users associated to his/her account.
    - User. This role will be able to carry out the discovery of the services.
    - CSP_Owner. This role is the owner of the account. It is able to create users with the role of CSPs associated to his/her account.
    - CSP. This role will be able to carry out the endorsement of the services.



Create or edit a user ×

ID

Login

This field is required.

First name

Last name

Email

This field is required.

Company

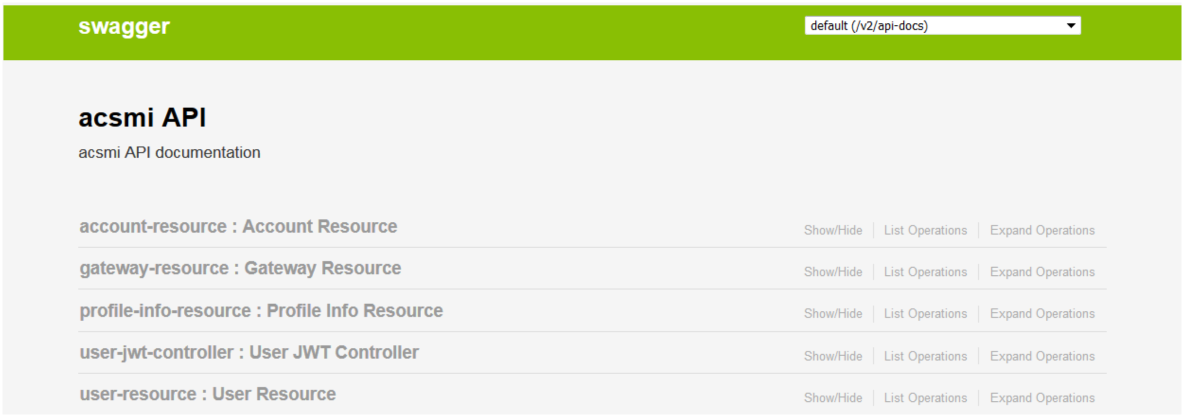
Users group

User type

- ○ Cloud Service Provider
- ○ Project Administration

Figure 6. Create or edit a user functionality

In addition to the graphical interface, this component offers an API with the REST operations shown in Figure 7 that allows automatizing some functionalities.



swagger default (/v2/api-docs)

acsmi API

acsmi API documentation

account-resource : Account Resource	Show/Hide	List Operations	Expand Operations
gateway-resource : Gateway Resource	Show/Hide	List Operations	Expand Operations
profile-info-resource : Profile Info Resource	Show/Hide	List Operations	Expand Operations
user-jwt-controller : User JWT Controller	Show/Hide	List Operations	Expand Operations
user-resource : User Resource	Show/Hide	List Operations	Expand Operations

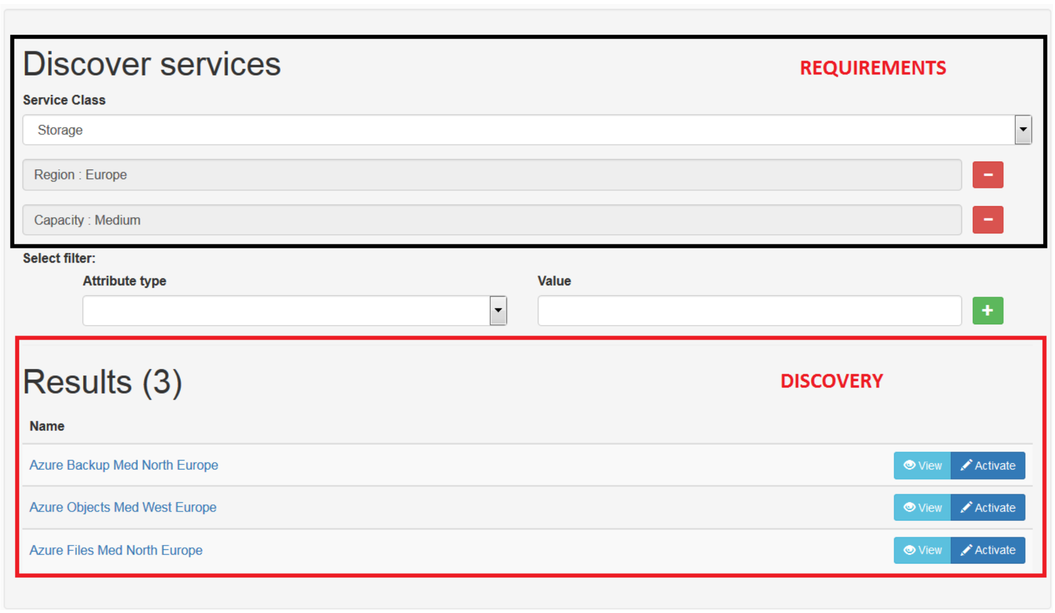
Figure 7. Frontend API Information.

3.2.2.2 Backend

This component is in charge of 1) carrying out the discovery of the services based on the information provided by the user and 2) the endorsement of the services based on the information provided by the CSP through the frontend. This backend is composed of one service, which implements the core functionalities of this prototype.

Discovery services

This functionality allows the discovery of the services taking as input the requirements introduced by the developer. In order to be able to access to this functionality the developer should have assigned a "user" role.


Discover services

REQUIREMENTS

Service Class: Storage

Region : Europe

Capacity : Medium

Select filter:

Attribute type | Value

Results (3)

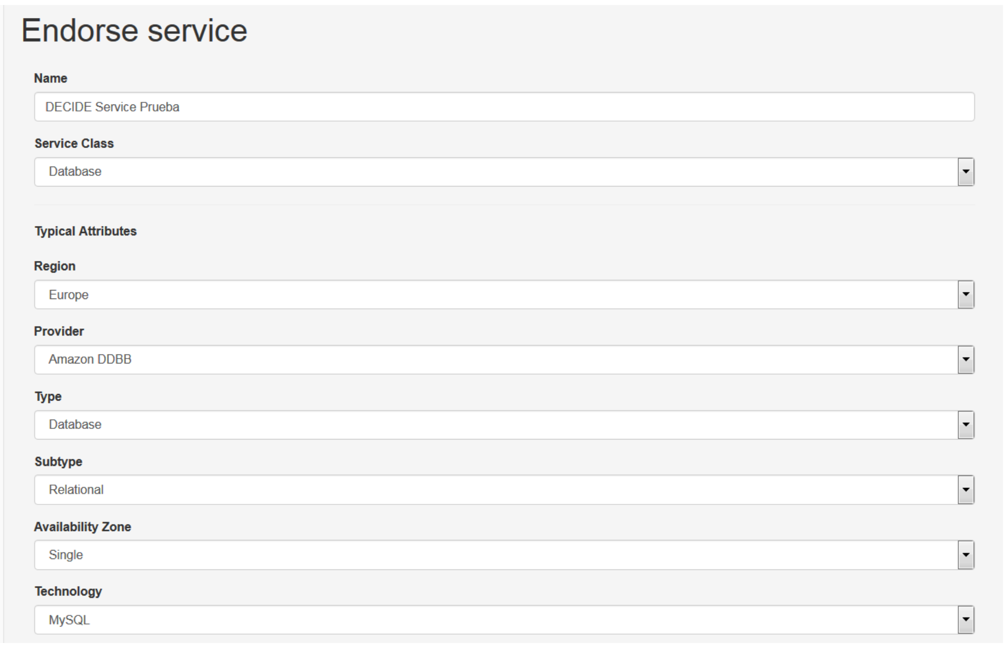
DISCOVERY

Name	
Azure Backup Med North Europe	View Activate
Azure Objects Med West Europe	View Activate
Azure Files Med North Europe	View Activate

Figure 8. Example result of a discovery.

Endorsement of services

This functionality allows to insert all the information required to add a service in the ACSmI service registry. The registry will have some mandatory information complemented with additional information, which could be used in a future release to benchmark the different services. The registry follows the model explained previously in this section. In order to be able to access to this functionality, it is required to have a user with the "CSP" role.


Endorse service

Name: DECIDE Service Prueba

Service Class: Database

Typical Attributes

Region: Europe

Provider: Amazon DDBB

Type: Database

Subtype: Relational

Availability Zone: Single

Technology: MySQL

Figure 9. Example of the endorsement of a service.

3.2.2.3 Registry

This component is totally based on the one provided by JHipster, that it is the baseline technology used to develop this prototype (See details in section 3.2.3).

JHipster Registry [3] is a run-time application, provided by the JHipster team. Like the JHipster generator, it is an Open Source, Apache 2-licensed application. The JHipster Registry has three main purposes:

- It is a Eureka server, which serves as a discovery server for applications. This is how JHipster handles routing, load balancing and scalability for all applications.
- It is a Spring Cloud Config server, which provide runtime configuration to all applications.
- It is an administration server, with dashboards to monitor and manage applications.

All those features are packaged into one convenient application with a modern Angular-based user interface.

3.2.2.4 Database

This component is a MySQL database, following the data model shown in the Figure 3.

3.2.3 Technical specifications

As mentioned previously, this prototype has been developed using JHipster.

JHipster [4] is not a development framework, it is more a frontend and backend generator based on different technologies.

The main objective of JHipster is to generate a modern and complete web application or micro-service architecture, unifying:

- A high-performance and robust Java stack on the server side with Spring Boot [5]
- A sleek, modern, mobile-first frontend with Angular [6] and Bootstrap [7]
- A robust micro service architecture with JHipster Registry [3], Netflix OSS [8], ELK stack [9] and Docker [10]
- A powerful workflow to build your application with Yeoman [11], Webpack [12]/Gulp [13] and Maven [14]/Gradle [15]

In addition, JHipster allows the use of Swagger to define the communication between the frontend and backend. Each functional component communicates with the other through API rest, although these components are in the same service of the backend.

3.3 Delivery and usage

3.3.1 Package information

The delivered package consists of three projects, as shown in Figure 10.

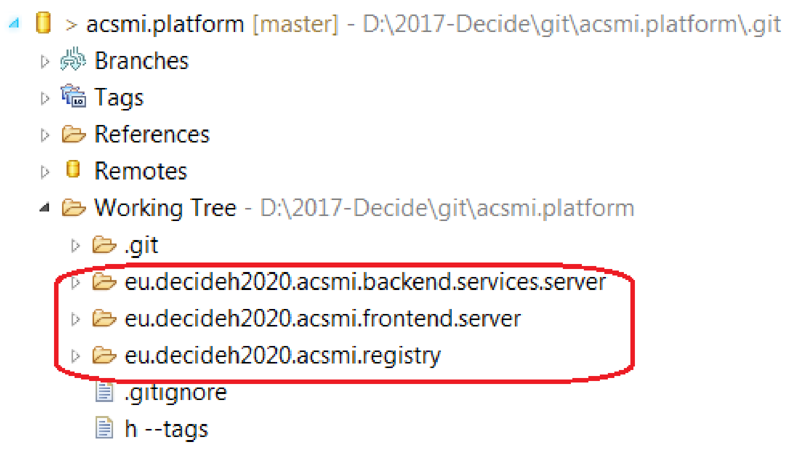
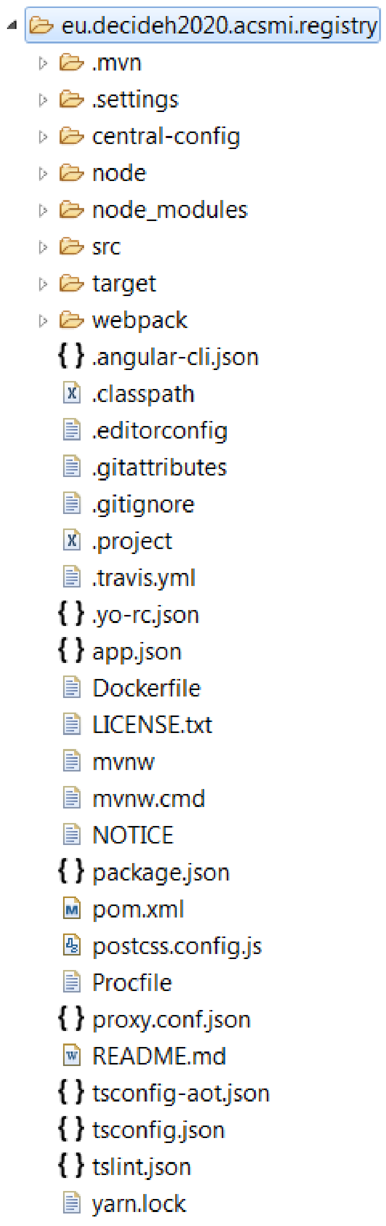

```
> acsmi.platform [master] - D:\2017-Decide\git\acsmi.platform\.git
    Branches
    Tags
    References
    Remotes
    Working Tree - D:\2017-Decide\git\acsmi.platform
        .git
        eu.decideh2020.acsmi.backend.services.server
        eu.decideh2020.acsmi.frontend.server
        eu.decideh2020.acsmi.registry
        .gitignore
        h --tags
```

Figure 10. Package Structure

These projects correspond to the components explained in previous sections. The inside structure of these projects is shown in the following figures.

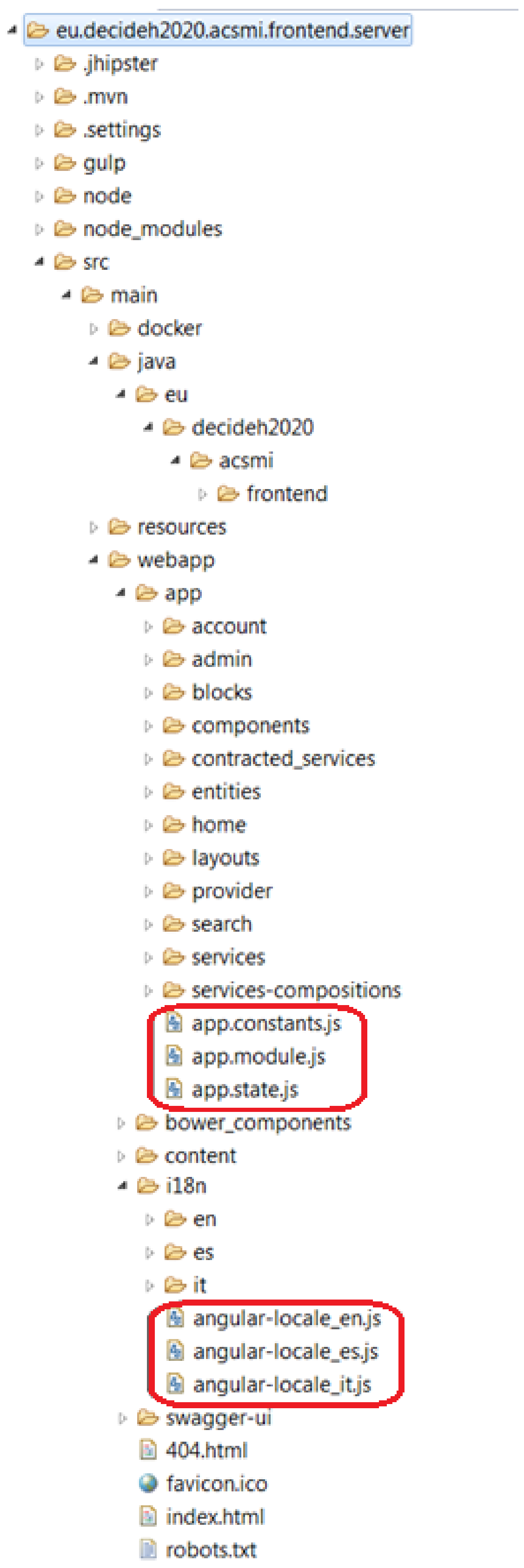
- *eu.decideh2020.acsmi.registry*. This project is a generic one provided by JHipster.



```
eu.decideh2020.acsmi.registry
  .mvn
  .settings
  central-config
  node
  node_modules
  src
  target
  webpack
  .angular-cli.json
  .classpath
  .editorconfig
  .gitattributes
  .gitignore
  .project
  .travis.yml
  .yo-rc.json
  app.json
  Dockerfile
  LICENSE.txt
  mvnw
  mvnw.cmd
  NOTICE
  package.json
  pom.xml
  postcss.config.js
  Procfile
  proxy.conf.json
  README.md
  tsconfig-aot.json
  tsconfig.json
  tslint.json
  yarn.lock
```

Figure 11. Registry component packages structure

- *eu.decideh2020.acsmi.frontend.server.* In this component, as it is the one controlling the graphical interface, it can be seen that in addition to the java source code, the folder Webapp is composed of files with .js extension (Angular).

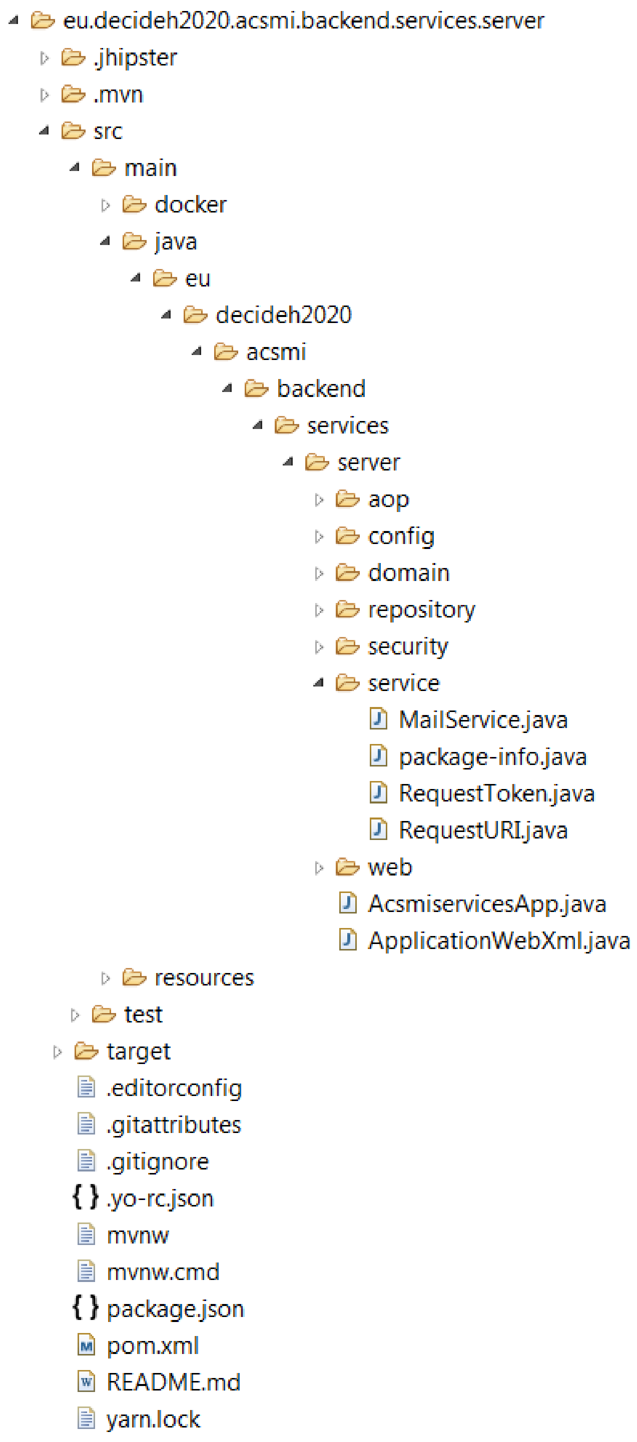


```
eu.decideh2020.acsmi.frontend.server
  .jhipster
  .mvn
  .settings
  gulp
  node
  node_modules
  src
    main
      docker
      java
        eu
          decideh2020
            acsmi
              frontend
      resources
      webapp
        app
          account
          admin
          blocks
          components
          contracted_services
          entities
          home
          layouts
          provider
          search
          services
          services-compositions
          app.constants.js
          app.module.js
          app.state.js
        bower_components
        content
        i18n
          en
          es
          it
          angular-locale_en.js
          angular-locale_es.js
          angular-locale_it.js
        swagger-ui
        404.html
        favicon.ico
        index.html
        robots.txt
```

Figure 12. Frontend component packages structure

The backend component is composed of one project.

- *eu.decideh2020.acsmi.backend.services.server*. This project is in charge of the core functionalities of the prototype. This project contains the source code for the implementation of these functionalities.



```
eu.decideh2020.acsmi.backend.services.server
  .jhipster
  .mvn
  src
    main
      docker
      java
        eu
          decideh2020
            acsmi
              backend
                services
                  server
                    aop
                    config
                    domain
                    repository
                    security
                    service
                      MailService.java
                      package-info.java
                      RequestToken.java
                      RequestURI.java
                    web
                    AcsmiservicesApp.java
                    ApplicationWebXml.java
      resources
    test
  target
  .editorconfig
  .gitattributes
  .gitignore
  .yo-rc.json
  mvnw
  mvnw.cmd
  package.json
  pom.xml
  README.md
  yarn.lock
```

Figure 13. Back-end component packages structure

These packages are accomplished with a database server, detailed in Figure 14.

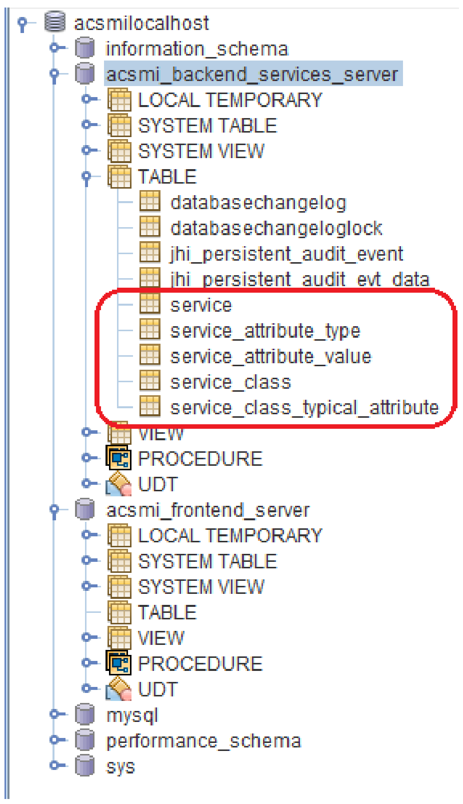


Figure 14. Databases

3.3.2 Installation instructions

The installation process needs to execute four activities:

1. Create the DB.

```
_cmd.bat
  mysqlstop.bat
  rmdir /s mysql-5.7.19-winx64\data
  mysqld --initialize-insecure
  mysqlstart.bat
  mysql -u root -e "CREATE DATABASE acsmi_backend_services_server"
  mysql -u root -e "CREATE DATABASE acsmi_frontend_server"
```

Figure 15. Code to create DB

2. Download the source code for ACSmI Discovery from the DECIDE repository.
3. Download the dependencies.

```
echo Start Frontend
  c:\Windows\System32\timeout.exe /t 5
  cd %CURRENT_DIR%\git\WP5\ACSmI_discovery\eu.decideh2020.acsmi.frontend.server
  start "Frontend" "mvnw" -Pprod

echo Start registry
  c:\Windows\System32\timeout.exe /t 5
  cd %CURRENT_DIR%\git\WP5\ACSmI_discovery\eu.decideh2020.acsmi.registry.server
  start "Registry" "mvnw" -Pprod
```

Figure 16. Code to download dependencies

4. Launch ACSmI.

```
echo Start Database
REM timeout /t 5 does not work due to cygwin
c:\Windows\System32\timeout.exe /t 5
REM pause
call mysqlstart.bat

echo Start Registry
c:\Windows\System32\timeout.exe /t 5
cd %CURRENT_DIR%\git\WP5\ACSmI_discovery\eu.decideh2020.acsmi.registry
start "Registry" "mvnw"

echo Start Contracts Backend
c:\Windows\System32\timeout.exe /t 5
cd %CURRENT_DIR%\git\WP5\ACSmI_discovery\eu.decideh2020.acsmi.backend.services.server
start "Services" "mvnw"


echo Start Frontend
c:\Windows\System32\timeout.exe /t 5
cd %CURRENT_DIR%\git\WP5\ACSmI_discovery\eu.decideh2020.acsmi.frontend.server
start "Frontend" "mvnw"

echo Load database, requires the components to be running
:while1
wget -qO- http://localhost:8083 > nul
if %ERRORLEVEL% NEQ 0 (
    echo services backend down at http://localhost:8083 waiting 10 segs
    c:\Windows\System32\timeout.exe /t 10
    goto :while1
)

mysql -u root -f -D acsmi_backend_services_server <
%CURRENT_DIR%\git\acsmi.devops\eu.decideh2020.acsmi.int.backend.services.server.test.00.
src.dvp\src\main\docker\sql\test.sql

pause
```

Figure 17. Code to launch ACSmI

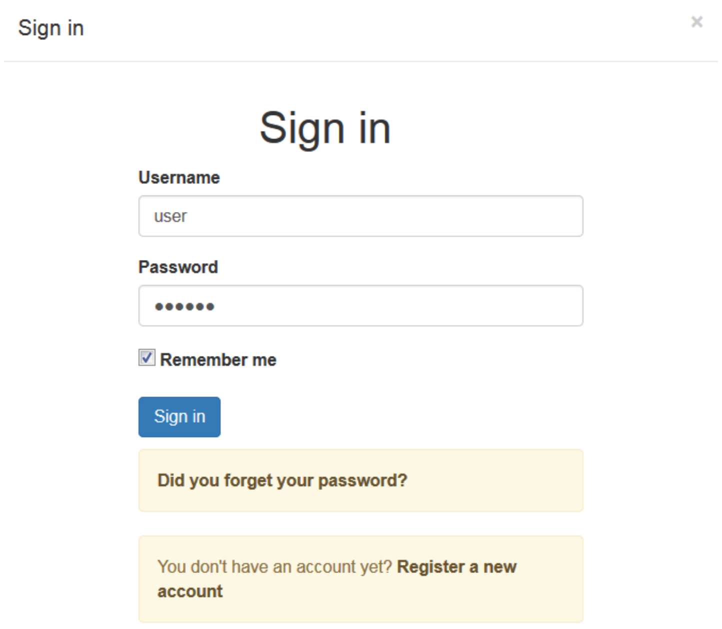
As a prerequisite, it is required to install MySQL database with a "root" user without password. After this, it is required to execute "mvnw" in each of the acsmi.platform folder.

3.3.3 User Manual

The first step to start with ACSmI discovery is to login. Several default users have been set up with different roles for ACSmI discovery.

Table 3. Users and passwords

Role	User	Password
User	user	decide
User_owner	userowner	decide
CSP	csp	decide
CSP_owner	cspowner	decide



Sign in

Sign in

Username

user

Password

••••••

☑ Remember me

Sign in

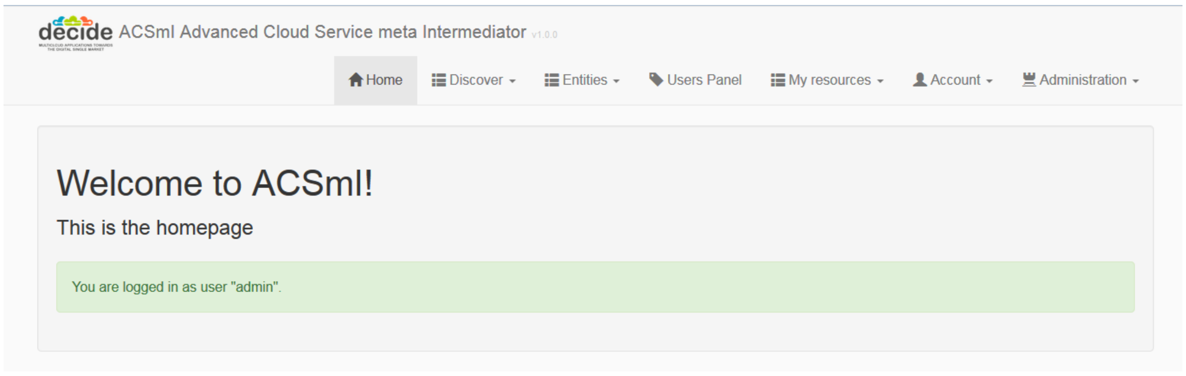
Did you forget your password?

You don't have an account yet? Register a new account

Figure 18. ACSmI login interface

Then depending on the role, it will be allowed to do different tasks.

The administrator will have access to all tasks:



ACSmI Advanced Cloud Service meta Intermediator v1.0.0

Home | Discover | Entities | Users Panel | My resources | Account | Administration

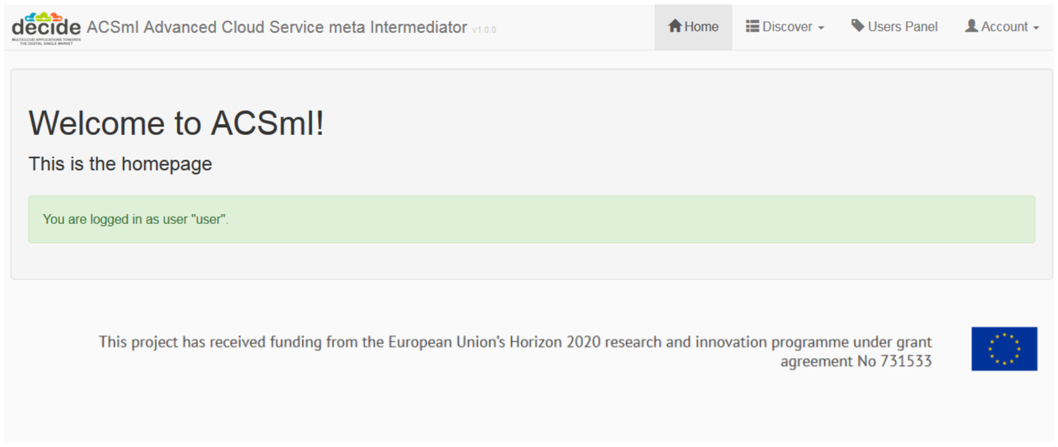
Welcome to ACSmI!

This is the homepage

You are logged in as user "admin".

Figure 19. Dashboard for admin

If the access is done as a user or a CSP, the allowed tasks are different.



ACSmI Advanced Cloud Service meta Intermediator v1.0.0

Home | Discover | Users Panel | Account

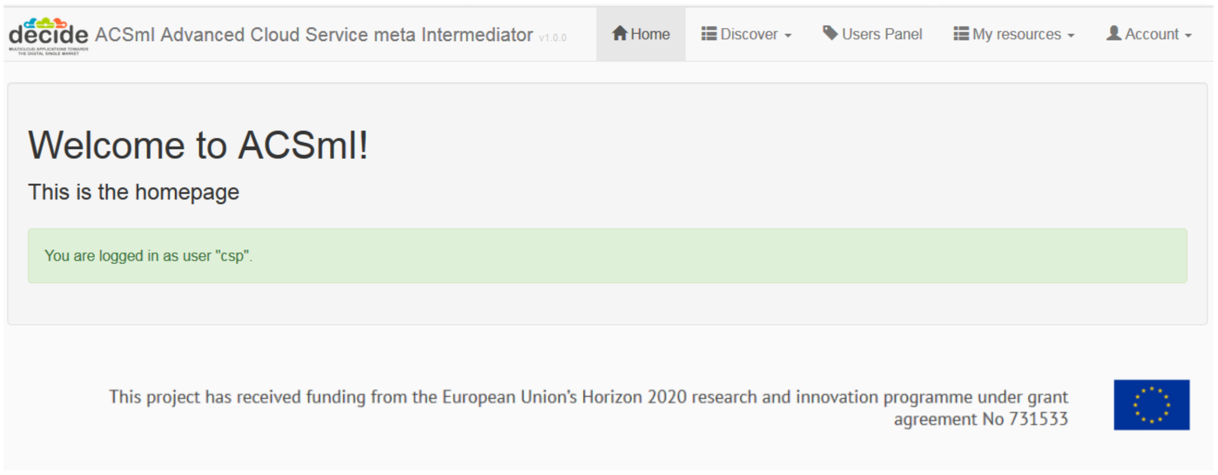
Welcome to ACSmI!

This is the homepage

You are logged in as user "user".

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731533

Figure 20. Dashboard for users


decide ACSmI Advanced Cloud Service meta Intermediator v1.0.0
Home
Discover
Users Panel
My resources
Account

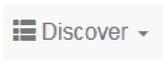
Welcome to ACSmI!

This is the homepage

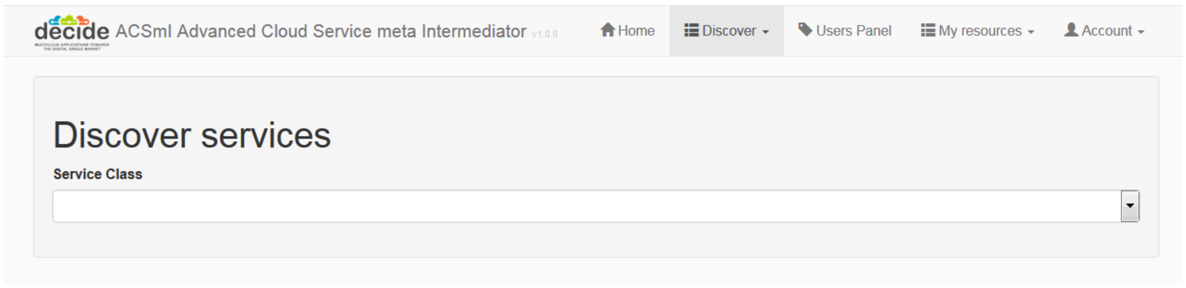
You are logged in as user "csp".

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731533

Figure 21. Dashboard for CSPs

The Discovery tab  will access to two different functionalities depending on the role.

- Role User: This tab will guide to the discovery functionality

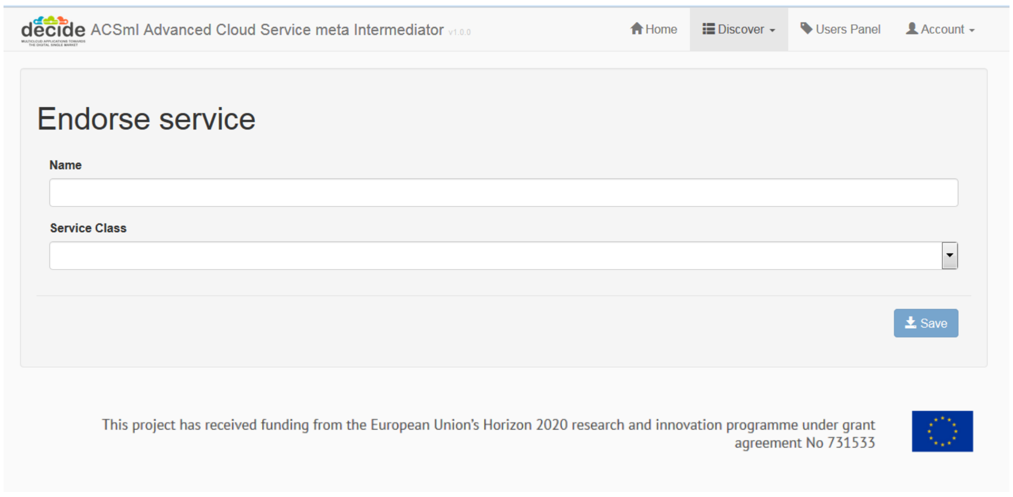

decide ACSmI Advanced Cloud Service meta Intermediator v1.0.0
Home
Discover
Users Panel
My resources
Account

Discover services

Service Class

Figure 22. Discovery functionality

- Role CSP: This tab will guide to the endorse functionality


decide ACSmI Advanced Cloud Service meta Intermediator v1.0.0
Home
Discover
Users Panel
Account
Save

Endorse service

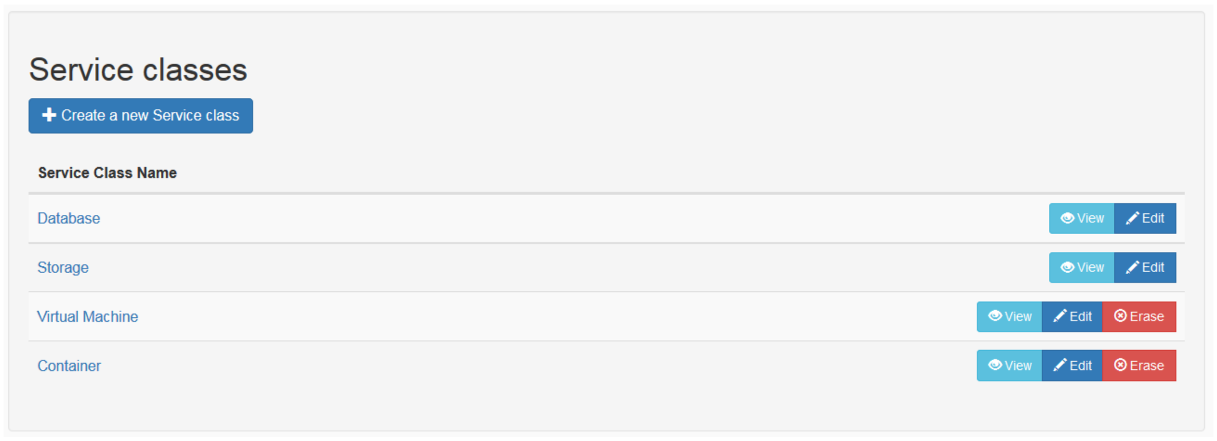
Name

Service Class

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731533

Figure 23. Endorse functionality

The ACSmI administrator role is defined in the entity Service Types. At this point of the DECIDE project, four service types are defined.

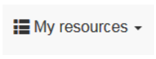


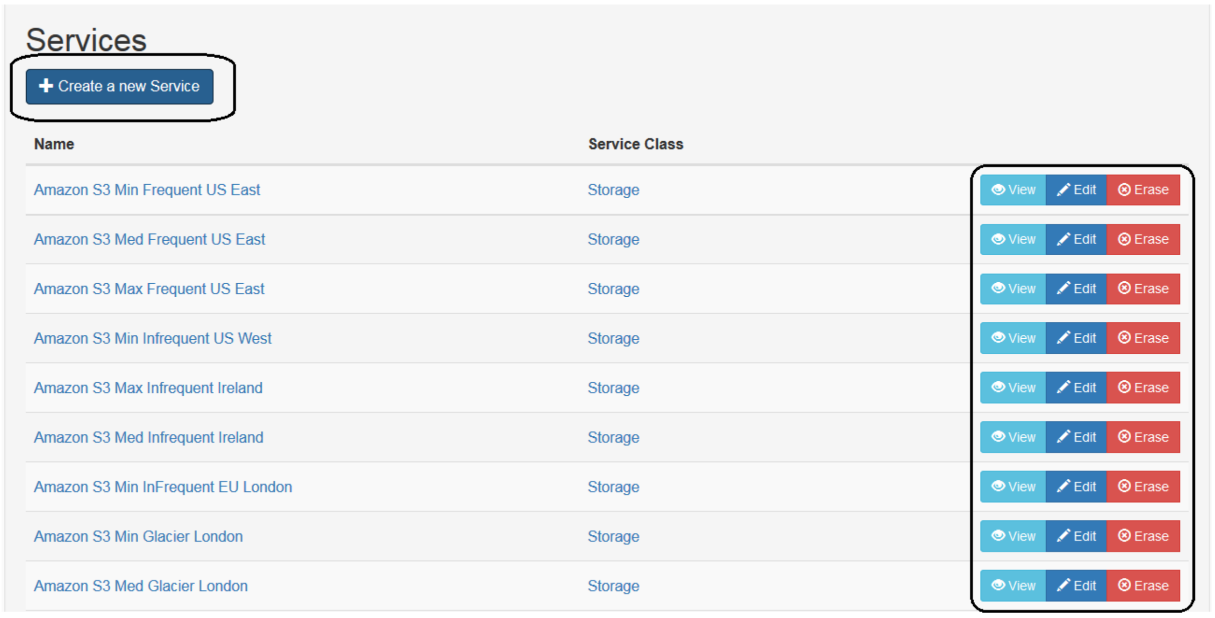
Service classes

+ Create a new Service class

Service Class Name	
Database	View Edit
Storage	View Edit
Virtual Machine	View Edit Erase
Container	View Edit Erase

Figure 24. Service type creation

From the My resources tab  the functionalities related to create, read, update or delete services can be accessed.



Services

+ Create a new Service

Name	Service Class	
Amazon S3 Min Frequent US East	Storage	View Edit Erase
Amazon S3 Med Frequent US East	Storage	View Edit Erase
Amazon S3 Max Frequent US East	Storage	View Edit Erase
Amazon S3 Min Infrequent US West	Storage	View Edit Erase
Amazon S3 Max Infrequent Ireland	Storage	View Edit Erase
Amazon S3 Med Infrequent Ireland	Storage	View Edit Erase
Amazon S3 Min InFrequent EU London	Storage	View Edit Erase
Amazon S3 Min Glacier London	Storage	View Edit Erase
Amazon S3 Med Glacier London	Storage	View Edit Erase

Figure 25. CRUD of services

3.3.4 Licensing information

This component is offered under Apache 2.

```
Copyright (c) 2017 Tecnalia.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

The code has been contributed over a stub generated with the JHipster
Application generator http://www.jhipster.tech/.

Contributors (in alphabetical order):

Gorka Benguria                      Tecnalia
Iñaki Etxaniz                       Tecnalia
Juncal Alonso                       Tecnalia
Leire Orue-Echevarria               Tecnalia
Maria Jose lopez                    Tecnalia
Marisa Escalante                    Tecnalia

Initially developed in the context of DECIDE EU project www.DECIDE-h2020.eu
```

Figure 26. Copyright header

3.3.5 Download

The source code is available in the EC portal for deliverables, included in the zip file for D5.2.

The first release is available in the DECIDE open git repository, more precisely at the following address:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/M12/ACSmI/Discovery

4 ACSmI Contract management

4.1 Functional description

ACSmI contract management shall be performed via the ACSmI contracting component that is to be designed and developed by the CloudBroker team. The Contract Management module is in charge of the execution and management of the core functions with respect to the service contracts. It manages mainly two different types of contracts: The contracts between the user and the ACSmI and the contracts between the CSPs (service providers) and the ACSmI.

The component is meant to be used by final solution users who would like to get an access to the cloud resources in order to deploy their software.

Requirements covered by the prototype:

Table 4. Requirements covered by the M12 ACSmI Contract management prototype.

Req. ID	Req. Description	Requirement coverage by the prototype
WP5-BUS07	This requirement shall allow contracting a service or services in the ACSmI for a certain multi-cloud application owner. After making a contract with the multi-cloud application owner, ACSmI handles the appropriate contracting with required SCP.	In this release, a multi-cloud application owner will be able to contract services available in the ACSmI by providing to the ACSmI Contract Management module his or her data needed to get a service contract through UI Interaction Controller component. In this way ACSmI will handle ACSmI-User contracting procedure.
WP5-BUS08	This requirement shall allow developer to contract a service or services directly with the CSP. ACSmI will require the information for the contracted services (SLAs) to be included in the registry and to be monitored.	In this release, it will be possible to contract services available for ACSmI users from the Contract Management UI interaction controller. The users having no credentials required for the service usage will be able to: • Get such credentials by autoregistration • Get the cloud account from the pool provided by ACSmI • If the above options are not possible – will be able to use ACSmI account for service usage. If a user already has cloud credentials, he or she will be able to use the credentials via ACSmI to get access to the service. ACSmI will collect the required contracted services information in the Contract Registry so that it could be retrieved by other DECIDE and monitored accordingly.

4.2 Technical description

The ACSmI Contract management module will handle two use-case scenarios to access the cloud resources by the user. First scenario provides that the user has his/her own credentials for the cloud resource he/she is willing to use. The second scenario provides that the user has no such credentials and in such a case all the further contracting is to be done in 2 stages: ACSmI to User contracting and ACSmI to Resource contracting.

Both scenarios are to be described in more detail in section 4.2.2. "Components description".

4.2.1 Prototype architecture

Below is the diagram showing the Contract management module's internal structure and its outer interactions.

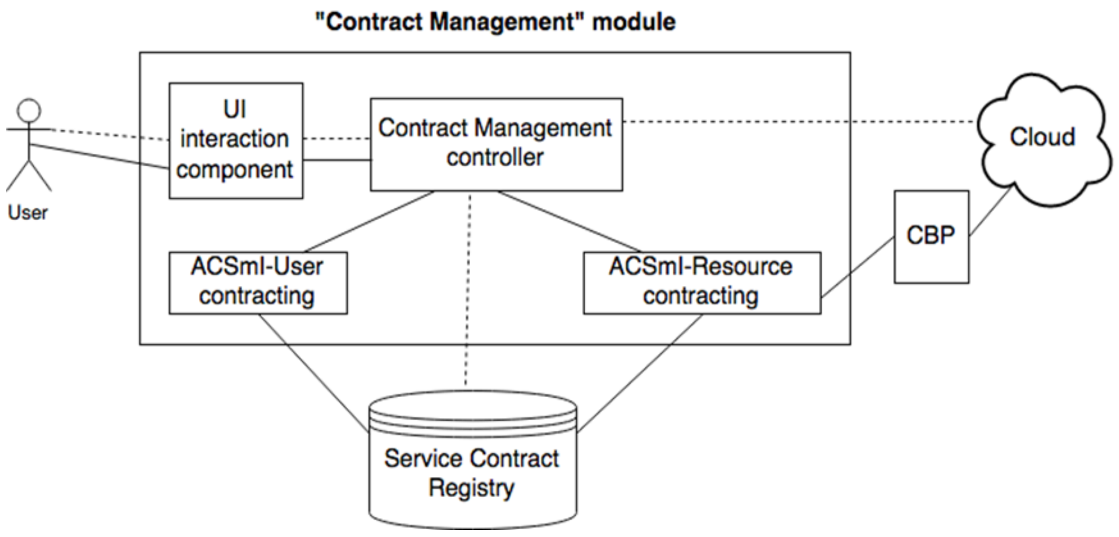

"Contract Management" module
UI interaction component
Contract Management controller
Cloud
User
CBP
ACSmI-User contracting
ACSmI-Resource contracting
Service Contract Registry

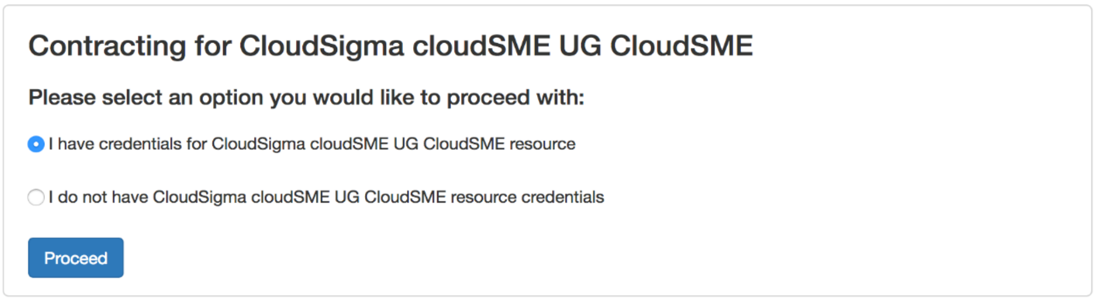
Figure 27. Contract management module

The major internal components of the Contract management module are:

- *UI interaction component*, that is responsible for interaction between user and the module itself.
- *Contract Management controller*, which is responsible for contracting depending on the presence or absence of user's cloud resource credentials: scenario 1 (dashed-line path) is performed in case the cloud credentials are available and scenario 2 (solid-line path) is performed in case the user does not have them.
- *ACSmI-User contracting* and *ACSmI-Resource contracting* components are involved in scenario 2.
- Module's outer interactions include storing all contracts within the *Service Contract Registry* and accessing the Resource through the CloudBroker Platform that acts as a proxy in scenario 2.

4.2.2 Components description

As mentioned above, there shall be **two scenarios** for accessing the cloud resources by the user, depending on the selection made in the form below, which is an entry point for contracting (Figure 28).


Contracting for CloudSigma cloudSME UG CloudSME

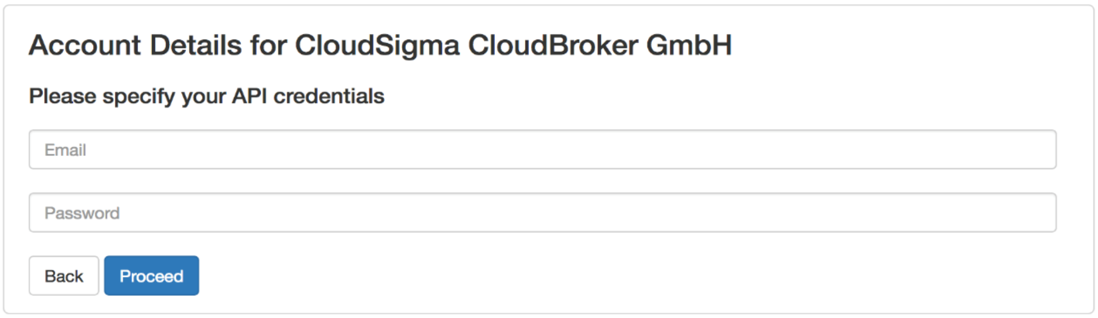
Please select an option you would like to proceed with:

[✓ I have credentials for CloudSigma cloudSME UG CloudSME resource]

I do not have CloudSigma cloudSME UG CloudSME resource credentials

Figure 28. Entry point for contracting

In the **first scenario** (given the user has own credentials for the selected cloud resource) the user is prompted to enter his/her cloud credentials (Figure 29).


Account Details for CloudSigma CloudBroker GmbH

Please specify your API credentials

Email

Password

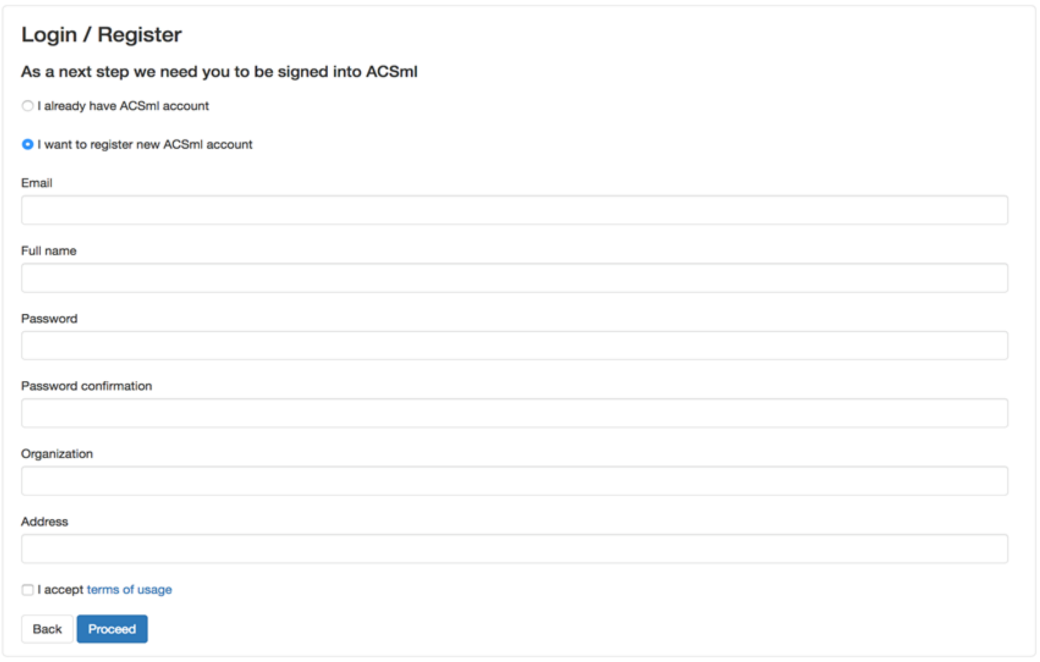
Figure 29. User has his/her own credentials

The credentials entered by the user are then stored within the ACSmI contracting component. The session identifier is returned back to the component requesting this information (either Application Controller, or ADAPT, or any other component if needed).

In the **second scenario** (given the user does not have own credentials for the selected cloud resource) the further contracting is done in 2 stages:

- Stage 1: ACSmI to User contracting.

At this stage user is required to provide necessary information for registering a new ACSmI account (see Figure 30) or (in case he/she already has own ACSmI account) to provide already existing credentials for logging in into ACSmI (see Figure 31):



Login / Register

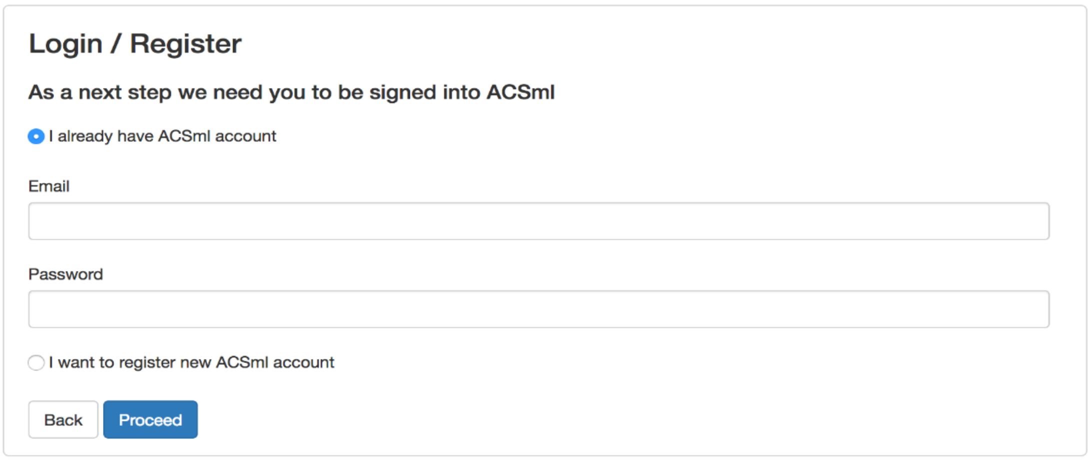
As a next step we need you to be signed into ACSmI

I already have ACSmI account [✓ I want to register new ACSmI account]

Email	
Full name	
Password	
Password confirmation	
Organization	
Address	

☐ I accept terms of usage

Figure 30. Register new account within ACSmI



Login / Register

As a next step we need you to be signed into ACSmI

[✓ I already have ACSmI account] I want to register new ACSmI account

Email	
Password	

Figure 31. Sign in to ACSmI with existing credentials

- Stage 2: ACSmI to Resource contracting.

Once the contract between user and ACSmI is set, the Component is to set up a contract between ACSmI and the Resource. Whenever the cloud adapter of the selected resource supports the automatic registration, ACSmI is to automatically create an account for the User. Please note that real user email address is not to be used. An internal ACSmI email/username and automatically generated password are to be provided as a resource credentials. This way the contracting between ACSmI and Resource will remain transparent for the user.

In case it is not technically possible to create an account automatically, the account from the Resource Account Pool is to be assigned to this specific user. If there are no resource accounts left in the pool, the user is to be contracted through the ACSmI-owned account.

As a result of the contracting operations above, it is suggested to return a Session ID back to the Application Controller. The Session ID is to be stored within Application Description file. ACSmI component is to associate certain session ID with either

- user-specified cloud credentials, or

- ACSmI user and resource that was contracted.

If the user has specified his own cloud credentials, they are to be stored within ACSmI component. ADAPT (or other DECIDE components) is to be able to retrieve the credentials via API using the Session ID.

If the contracting is done through the ACSmI, there is no way foreseen to obtain access to the cloud by other DECIDE components. The credentials obtained during the automatic account registration, credentials of the pooled-account or the ACSmI-owned account credentials should not leave the ACSmI component for billing and security reasons.

This means the whole communication between ADAPT (and other DECIDE components, if necessary) and the Resource is to be performed through the CloudBroker Platform (CBP). Credentials of the CBP user created for this operation can be obtained via the API call to ACSmI.

4.2.3 Technical specifications

The component is suggested to be implemented as a standalone application. It should be possible to easily setup an ACSmI Contracting component by external users. The solution is to be open-source. Possible consideration for the later versions: the solution can be "dockerized".

The following technical solutions are suggested for implementation:

- **Framework.** The component is to be implemented using Ruby on Rails framework [16]. The technology should allow fast prototype implementation, easy connectivity to other components (due to native API support), performance as well as rapid setup.
- **Prototype DB solution.** It is suggested to use sqlite [17] as a Prototype DB solution due to light load expected at initial stages.
- **Server setup.** Nginx [18] plus passenger [19] is the recommended server setup, however the application can be setup within any other rails-supporting server.

In order to be able to perform contracting, an ACSmI Component needs to be connected to the CloudBroker Platform instance. It is not needed to setup the Platform separately. The Platform API URL together with user credentials is to be specified.

All users registered on the ACSmI are to be created as standard users with limited access permissions within the organization of the user, whose credentials were configured in the step above.

The component database scheme can be found on the figure below:

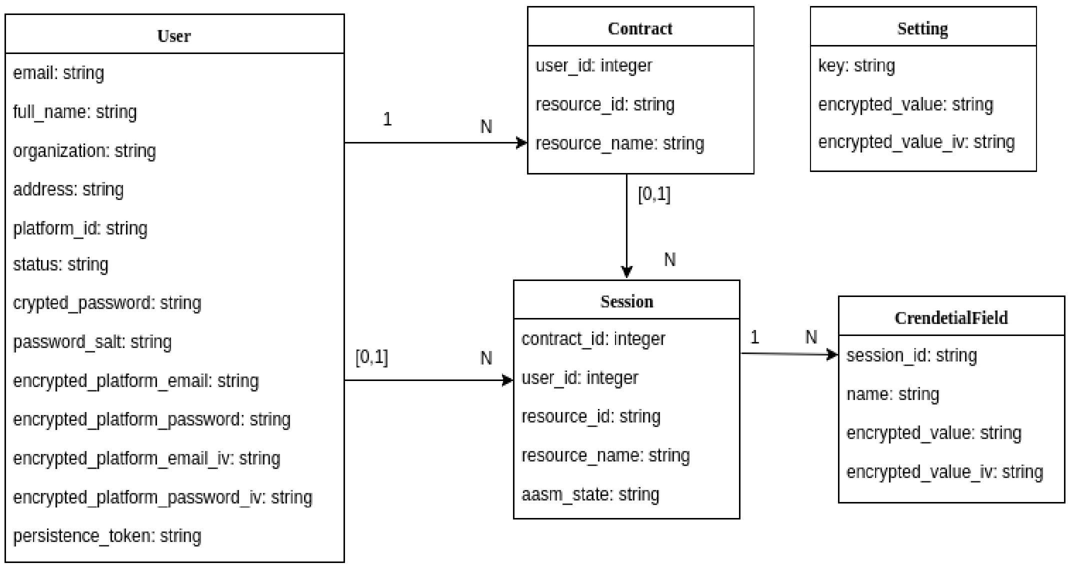
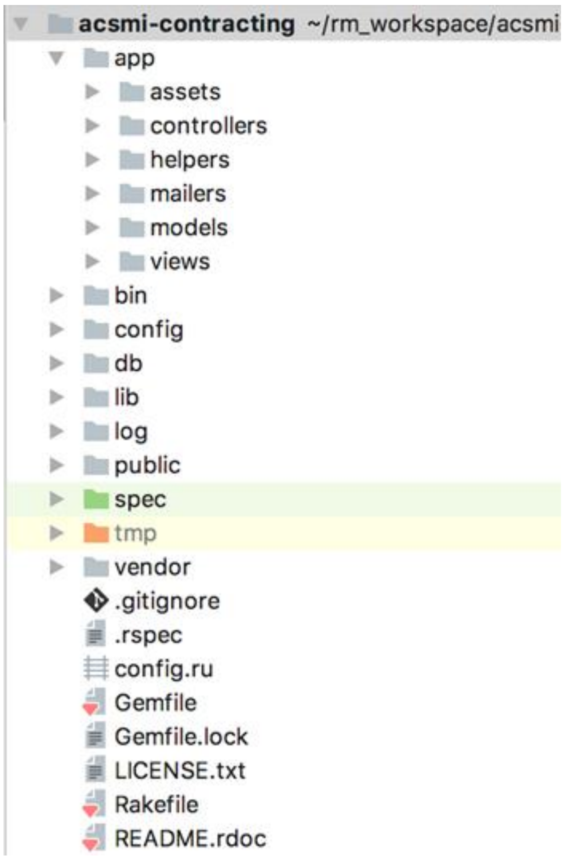


Figure 32. Component database structure

4.3 Delivery and usage

4.3.1 Package information

The delivered archive contains the acsmi-contracting project with the structure as shown on the figure below:



```
▼ acsmi-contracting ~/rm_workspace/acsmi
  ▼ app
    ▶ assets
    ▶ controllers
    ▶ helpers
    ▶ mailers
    ▶ models
    ▶ views
  ▶ bin
  ▶ config
  ▶ db
  ▶ lib
  ▶ log
  ▶ public
  ▶ spec
  ▶ tmp
  ▶ vendor
    .gitignore
    .rspec
    config.ru
    Gemfile
    Gemfile.lock
    LICENSE.txt
    Rakefile
    README.rdoc
```

Figure 33. Project structure

The structure of the project corresponds to the generic Ruby on Rails project structure. The app folder contains assets (javascript and stylesheets files), controllers (both UI and API), models and views (haml and erb templates).

Bin folder includes rails framework related files, that are necessary for application execution.

Config folder contains configuration files.

DB folder contains db schema and migrations needed to re-create a database.

Lib folder contains the tool to connect to the CloudBroker Platform via API (created in a scope of the project).

Log tool is a folder to contain server logs.

Public folder contains images and static html pages.

4.3.2 Installation instructions

1. Install ruby version 2.1.0 (https://www.ruby-lang.org/en/downloads/ for Windows or rvm.io for Unix). Please make sure ‘ruby -v’ returns the correct version in console.

2. Install dependencies. Go to project folder and run the following: ‘gem install bundler; bundle install’. Wait till all the gems are installed

3. Configure the access to the CloudBroker Platform. Go to db/seeds.rb and provide your Platform account details under 'platform_email' and 'platform_password' settings (please change value, not the key)

4. Create and configure the database. Go to project folder and run the following: 'rake db:create; rake db:migrate; rake db:seed'

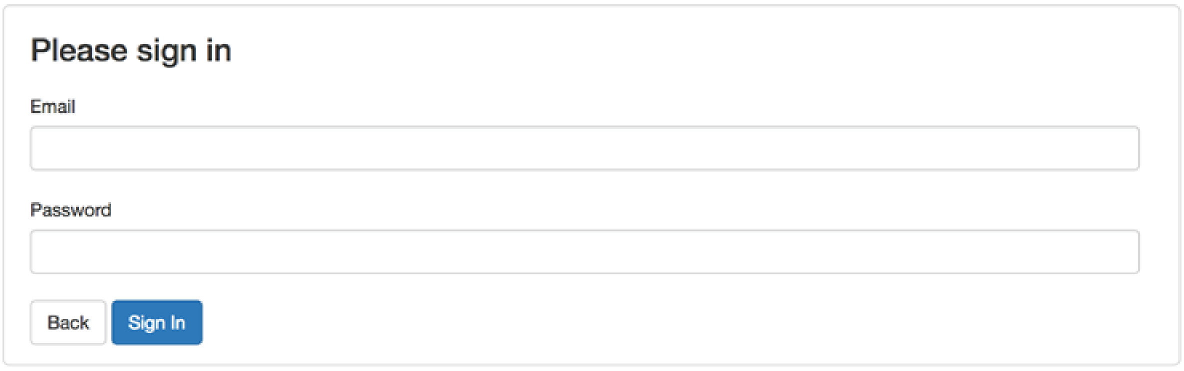
5. Start the server. Run "bin/rails server -b 0.0.0.0 -p 3000 -e development"

6. Open your browser, navigate to http://0.0.0.0:3000

4.3.3 User Manual

4.3.3.1 Logging in

User can sign in into his/her existing ACSmI account by clicking on the "Login" link in the navigation menu. The following login form will then be displayed (see Figure 34):


Please sign in

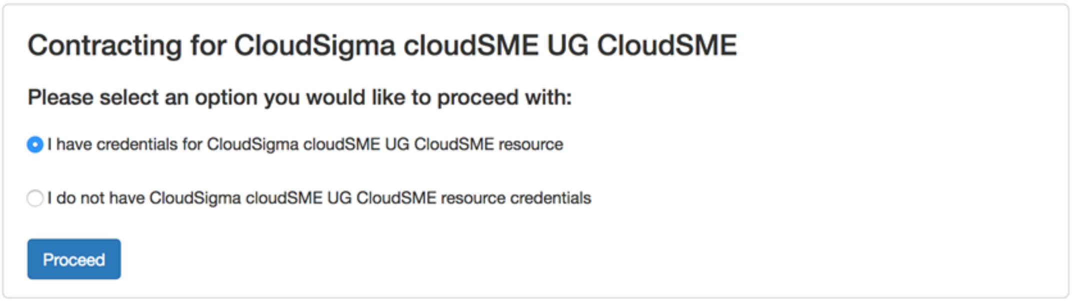
Email	
Password	

Figure 34. Login form

User has to enter his/her email and password and then, in case the provided credentials are valid and correct, user is logged into ACSmI.

4.3.3.2 Resource contracting

In order to communicate with the ACSmI Contracting component an API call to /decide/acsmi/contracting/api/v1/sessions with: resource_id as a request parameter should be made. The API response will contain a link. By following this link, user will be redirected to the form that serves as an entry point for contracting. In this form user is prompted to select one of the two options depending on presence or absence of resource credentials. Once user selects the appropriate option, the "Proceed" button is to be clicked (see Figure 35).


Contracting for CloudSigma cloudSME UG CloudSME

Please select an option you would like to proceed with:

● I have credentials for CloudSigma cloudSME UG CloudSME resource

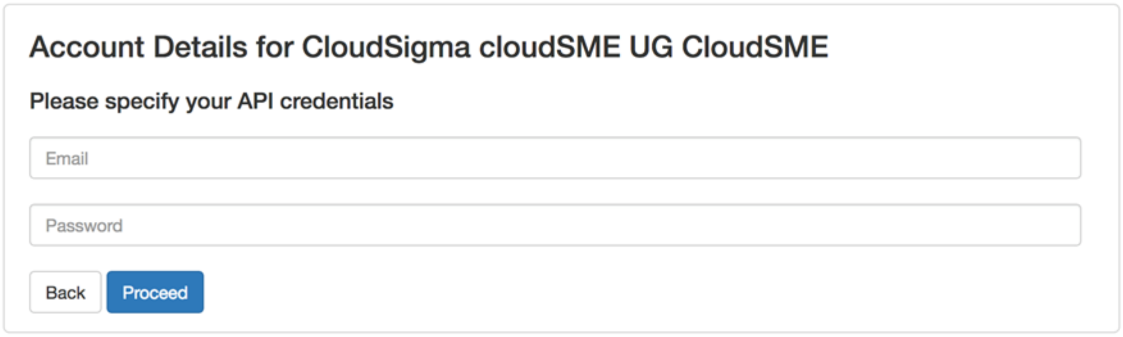
○ I do not have CloudSigma cloudSME UG CloudSME resource credentials

Figure 35. Entry point for contracting

The user's choice in the form above shall define the scenario to be applied.

Scenario 1: User selects the "I have credentials for CloudSigma cloudSME UG CloudSME resource".

Step 1.1: In this scenario, a form is displayed to the user where he/she has to enter already existing resource credentials, i.e. email and password (see Figure 36). User can also go back to the previous page by clicking on the "Back" button.

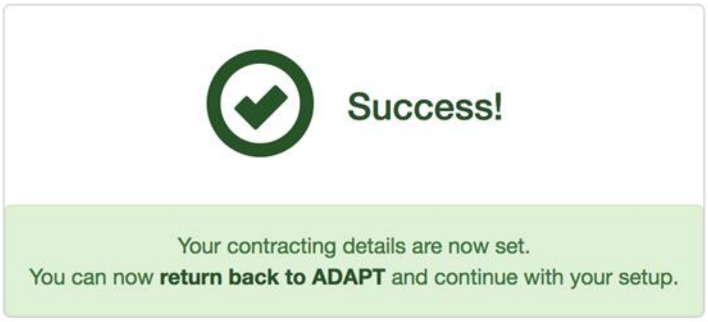

Account Details for CloudSigma cloudSME UG CloudSME

Please specify your API credentials

Email
Password

Figure 36. Existing resource account information

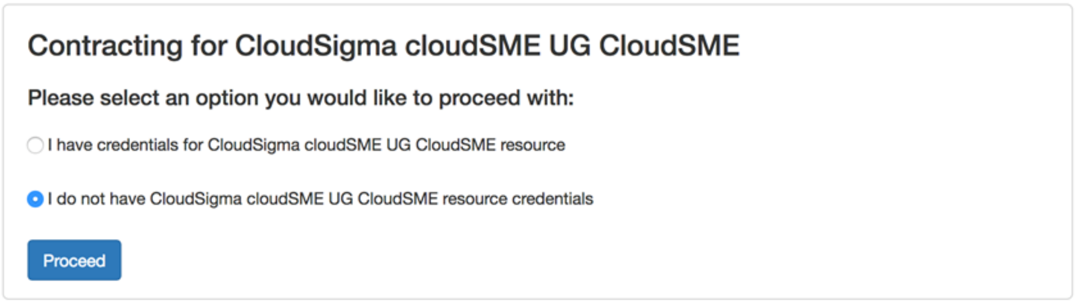
Step 1.2: The credentials are then stored within the ACSmI. Please note, that the credentials are not verified for validity at the moment (can be added in later versions). The contracting has been successfully performed, the following page shall be displayed (see Figure 37). The first scenario is then successfully completed.


Success!

Your contracting details are now set.
You can now **return back to ADAPT** and continue with your setup.

Figure 37. Successful contracting

However, the user can initiate the second scenario if the option "I do not have credentials for CloudSigma cloudSME UG CloudSME resource credentials" is selected on the entry point screen below:


Contracting for CloudSigma cloudSME UG CloudSME

Please select an option you would like to proceed with:

○ I have credentials for CloudSigma cloudSME UG CloudSME resource

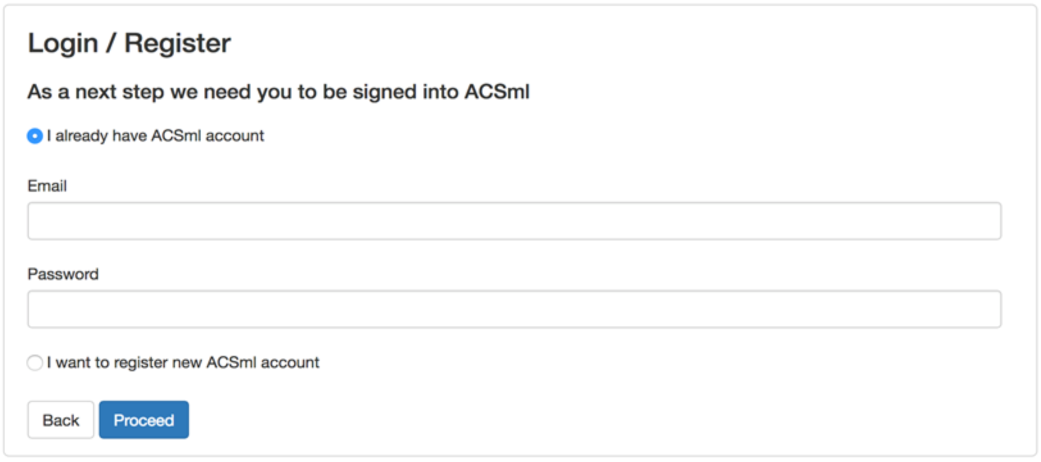
[✓ I do not have CloudSigma cloudSME UG CloudSME resource credentials]

Figure 38. Entry point for contracting

Scenario 2: User selects the "I do not have credentials for CloudSigma cloudSME UG CloudSME resource credentials".

Following the scenario 2, the first stage of this scenario is ACSmI to User contracting. To do this, user has to be signed in into ACSmI. In case user is already signed in, he/she is redirected to the credentials validation and contract setup process described under Steps 2.2 and 2.3.

Step 2.1.1: In case user is not signed in, but already has an ACSmI account, he/she needs to select the first option "I already have ACSmI account" and then provide the existing credentials, i.e. email and password, and click on "Proceed" (see Figure 39).


Login / Register

As a next step we need you to be signed into ACSmI

[✓ I already have ACSmI account]

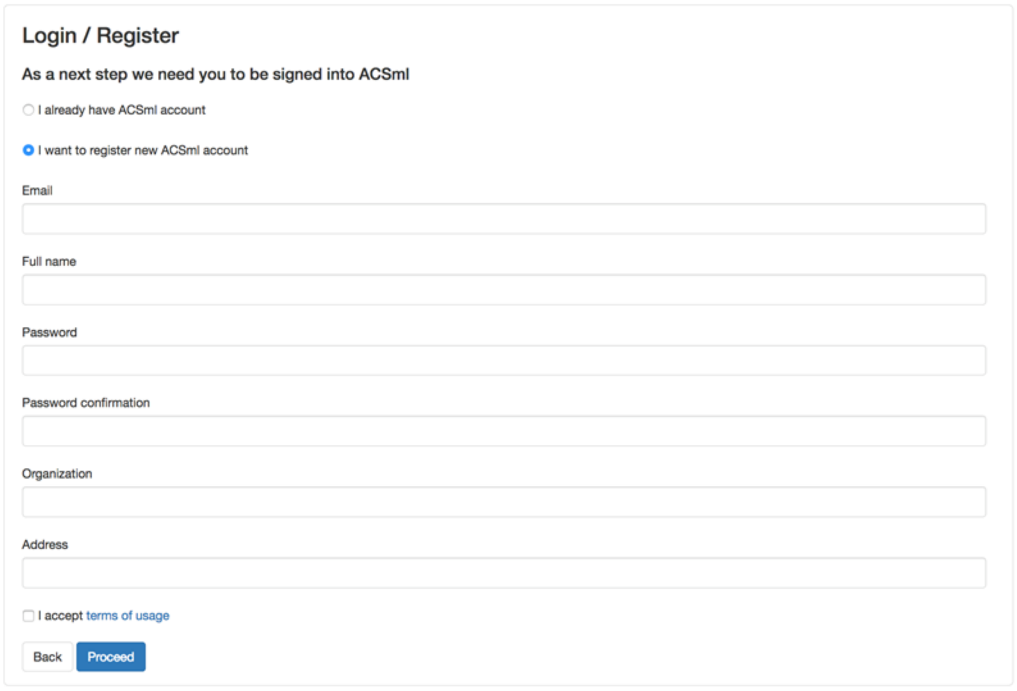
Email

Password

I want to register new ACSmI account

Figure 39. Form for providing already existing ACSmI account credentials

Step 2.1.2: In case user does not have an existing ACSmI account, the second option "I want to register new ACSmI account" is to be selected and the corresponding fields are to be filled in. After that, the "Proceed" button is to be clicked (see Figure 40).


Login / Register

As a next step we need you to be signed into ACSmI

I already have ACSmI account

[✓ I want to register new ACSmI account]

Email

Full name

Password

Password confirmation

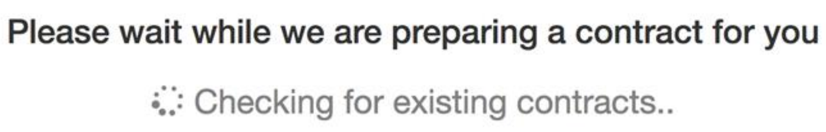
Organization

Address

I accept terms of usage

Figure 40. Form for registering a new ACSmI account

Step 2.2: Once the "Proceed" button is clicked, the user's credentials are being validated or, in case user had no existing ACSmI account, a new account is being set up and a contract is being prepared. Meanwhile, the following page is displayed (see Figure 41):

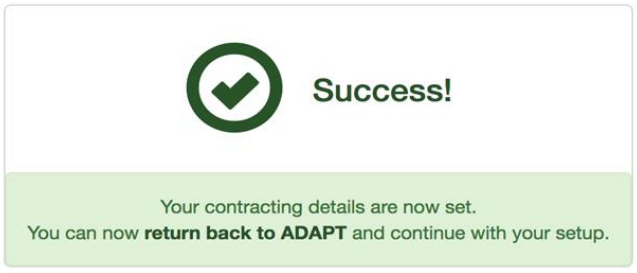


Please wait while we are preparing a contract for you

Checking for existing contracts..

Figure 41. Loading screen for data validation and contract preparation

Step 2.3: In case the data verification and contract preparation have been successfully performed, the following page shall be displayed (see Figure 42). The second scenario is then successfully completed.



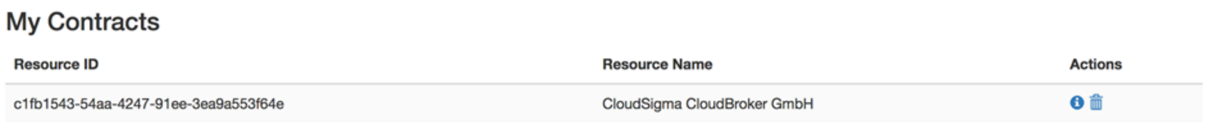
Success!

Your contracting details are now set.
You can now **return back to ADAPT** and continue with your setup.

Figure 42. Successful contracting

4.3.3.3 Contracts management

Once logged in, user is able to view and manage his/her existing contracts (see Figure 43).

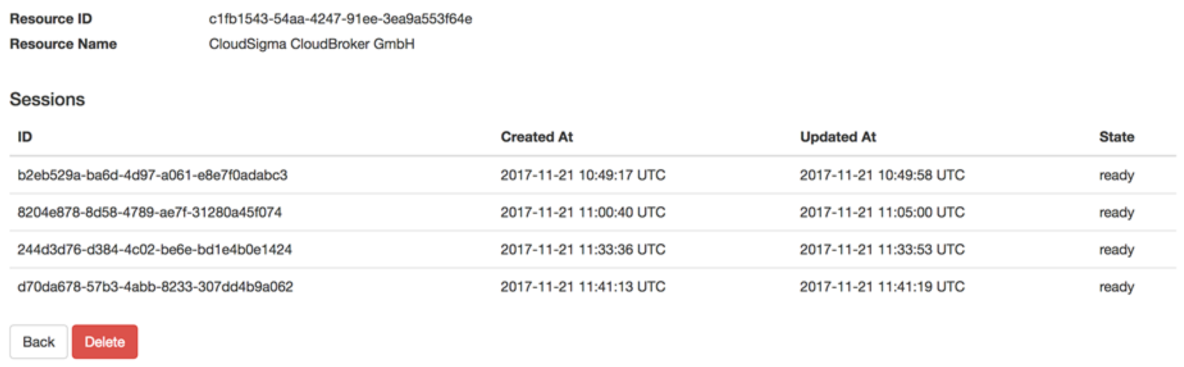


My Contracts

Resource ID	Resource Name	Actions
c1fb1543-54aa-4247-91ee-3ea9a553f64e	CloudSigma CloudBroker GmbH	

Figure 43. “My contracts” page

There are 2 actions available for each contract: view and delete. By clicking the “information” icon for a corresponding resource, user shall be able to view all resource sessions and their timeframe and state (see Figure 44). By clicking on “Delete” button user can delete the given resource contract. The “Back” button shall lead back to “My contracts” page.



Resource ID	c1fb1543-54aa-4247-91ee-3ea9a553f64e
Resource Name	CloudSigma CloudBroker GmbH

Sessions

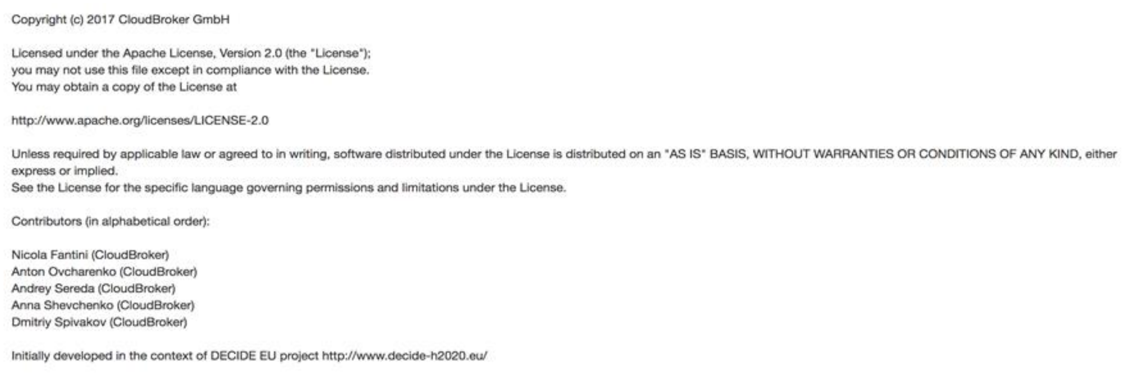
ID	Created At	Updated At	State
b2eb529a-ba6d-4d97-a061-e8e7f0adabc3	2017-11-21 10:49:17 UTC	2017-11-21 10:49:58 UTC	ready
8204e878-8d58-4789-ae7f-31280a45f074	2017-11-21 11:00:40 UTC	2017-11-21 11:05:00 UTC	ready
244d3d76-d384-4c02-be6e-bd1e4b0e1424	2017-11-21 11:33:36 UTC	2017-11-21 11:33:53 UTC	ready
d70da678-57b3-4abb-8233-307dd4b9a062	2017-11-21 11:41:13 UTC	2017-11-21 11:41:19 UTC	ready

Figure 44. Contract information page

On the “My contracts” page if the “Delete” icon is clicked in front of the corresponding resource contract; the given contract will be deleted. If user no longer has any resource contracts available, the following message will be displayed: “You do not have any active contracts at the moment”.

4.3.3.4 License

The component license can be found under the “License” link in the navigation menu (see Figure 45).


Copyright (c) 2017 CloudBroker GmbH

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.

Contributors (in alphabetical order):

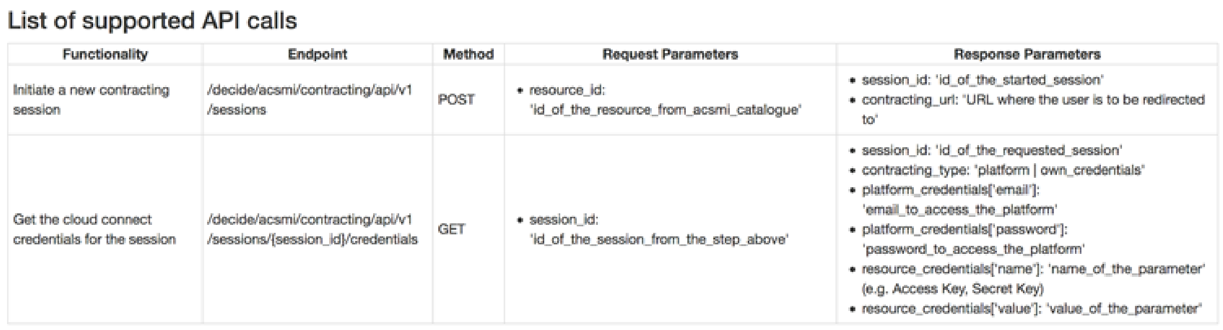
Nicola Fantini (CloudBroker)
Anton Ovcharenko (CloudBroker)
Andrey Sereda (CloudBroker)
Anna Shevchenko (CloudBroker)
Dmitriy Spivakov (CloudBroker)

Initially developed in the context of DECIDE EU project http://www.decide-h2020.eu/

Figure 45. License page

4.3.3.5 API

List of supported API calls can be found under the "API" link in the navigation menu (see Figure 46).

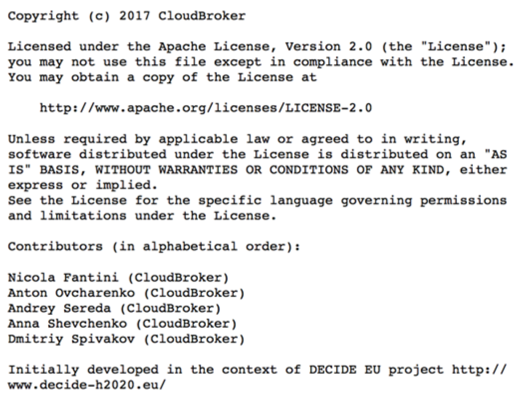

List of supported API calls

Functionality	Endpoint	Method	Request Parameters	Response Parameters
Initiate a new contracting session	/decide/acsmi/contracting/api/v1/sessions	POST	• resource_id: 'id_of_the_resource_from_acsmi_catalogue'	• session_id: 'id_of_the_started_session' • contracting_url: 'URL where the user is to be redirected to'
Get the cloud connect credentials for the session	/decide/acsmi/contracting/api/v1/sessions/{session_id}/credentials	GET	• session_id: 'id_of_the_session_from_the_step_above'	• session_id: 'id_of_the_requested_session' • contracting_type: 'platform | own_credentials' • platform_credentials['email']: 'email_to_access_the_platform' • platform_credentials['password']: 'password_to_access_the_platform' • resource_credentials['name']: 'name_of_the_parameter' (e.g. Access Key, Secret Key) • resource_credentials['value']: 'value_of_the_parameter'

Figure 46. List of supported API calls

4.3.4 Licensing information

The component shall be delivered under the "Apache 2.0" license. The license text can be found on the figure below.


```
Copyright (c) 2017 CloudBroker

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an "AS
IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
See the License for the specific language governing permissions
and limitations under the License.

Contributors (in alphabetical order):

Nicola Fantini (CloudBroker)
Anton Ovcharenko (CloudBroker)
Andrey Sereda (CloudBroker)
Anna Shevchenko (CloudBroker)
Dmitriy Spivakov (CloudBroker)

Initially developed in the context of DECIDE EU project http://
www.decide-h2020.eu/
```

Figure 47. Component license

4.3.5 Download

The first release is available in the DECIDE open git repository, more precisely at the following address:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/M12/ACSmI/Contracting

5 ACSmI Monitoring

5.1 Functional description

In order to monitor the SLAs (NFRs) of the service offered by the CSPs so as to detect any violation of the SLAs, it is required to select which are the parameters that need to be measured in the cloud services.

<u>Requirements covered by the prototype:</u>

Table 5. Requirements covered by the M12 ACSmI Monitoring prototype.

Req. ID	Req. Description	Requirement coverage by the prototype
WP5-MON03	The objective of this requirement is to relate the different SLA terms and NFRs, to the parameters to be monitored by ACSmI. This shall generate a generic list of parameters to be monitored for each NFR and SLA term.	In this release, the NFRs taken into account for deriving the associated parameters are: • Availability of the Cloud service
WP5-MON04	Based on the SLA contracted and the NFRs, the list of parameters to be monitored shall be selected from the generic list of parameters (WP5-MON03)	In this release, the relationship between the parameters (WP5-MON3) and the service types (WP5-DIS01) will be detailed

During this period, the NFR of Availability has been analysed. This NFR is not measured in the same way in all the services types. The following table summarises the metrics and which parameters are required to measure the NFR of Availability.

Table 6. NFR Availability

Type of services	Metric	Parameters[1]
Storage/Backup	Monthly Uptime Percentage [20] = Monthly Uptime % = (Maximum Available Minutes - Downtime) / Maximum Available Minutes	**Deployment Minutes:** it is the total number of minutes during which an item has been scheduled for Backup. **Downtime:** it is the total accumulated Deployment Minutes scheduled for Backup during which the Backup Service is unavailable for the item. The Backup Service is considered from the first Failure to Back Up or Restore until the initiation of a successful Backup or Recovery. **Maximum Available Minutes:** is the sum of all Deployment Minutes during a billing month.
Storage	Monthly Uptime Percentage [21] is calculated by subtracting from 100% the average of the Error Rates from each five-minute period in the monthly billing cycle.	**Error Rate:** it is the total number of internal server errors returned error status "**ServiceUnavailable**" divided by the total number of requests for the applicable request type during that five-minute period. For example: Service Unavailable= Database properties State different from ONLINE [22]

[1] When possible parameters from the Telegraf plug-in will be measured, if it is not possible we will extend Telegraf plug-ins.

Type of services	Metric	Parameters[1]
Database	Monthly Uptime Percentage [20] = Maximum Available Minutes-Downtime) / Maximum Available Minutes	**Maximum Available Minutes**: it is the total number of minutes that an instance of a Database has been deployed in a billing month. **Downtime**: it is the total accumulated minutes during a billing month for a Database is unavailable. A minute is considered **unavailable** for a given Database if all continuous attempts by Customer to establish a connection to the Database within the minute fail. **Unavailable** = STATE parameter different from ONLINE (From Telegraf plug-in)
Virtual Machine/ Container	Monthly Uptime Percentage [23] is calculated by subtracting from 100% the percentage of minutes during the month in which the virtual machine was in the state of "Region Unavailable."	**Region Unavailable:** means the region in which an instance is ran is "Unavailable". **Unavailable:** means all of running instances have no external connectivity. For example: **StatusCheckFailed_Instance** [24]: Reports whether the instance has passed the instance status check in the last minute. This metric can be either 0 (passed) or 1 (failed).

5.2 Technical description

The implementation of this component is not required for this release of the prototype. In this section, it is explained the vision that we have to implement it.

For the implementation of this component of ACSmI, a similar approach that the one followed in ADAPT monitoring [25] will be used.

Figure 48 represents the tentative architecture for this component.

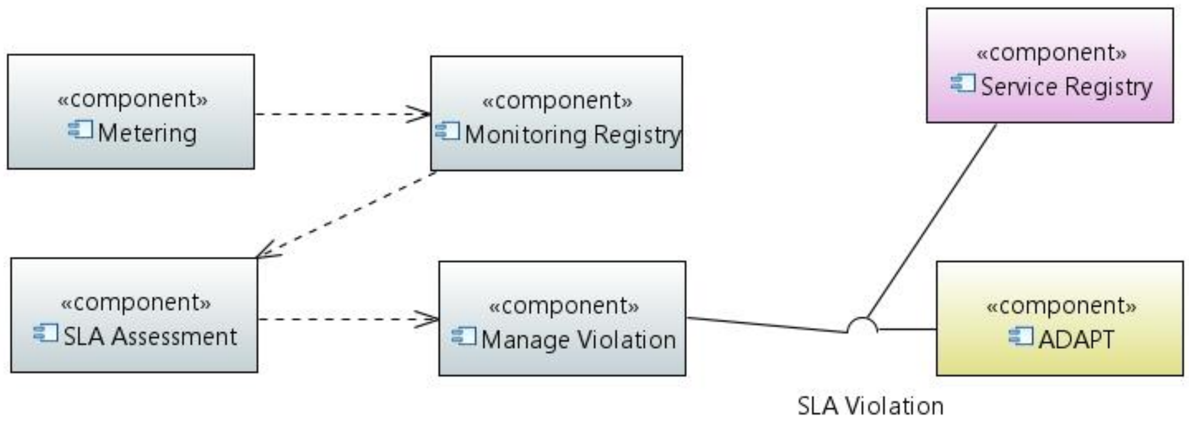

«component» Metering
«component» Monitoring Registry
«component» Service Registry
«component» SLA Assessment
«component» Manage Violation
«component» ADAPT
SLA Violation

Figure 48. ACSmI Monitoring Architecture

This component will be composed of four main subcomponents:

- **Metering** collects the different parameters defined in the Table 6 depending on SLA terms that will be monitored. To collect these parameters, the Telegraf [26] open source technology will be used.
- **Monitoring repository** will be in charge of storing the data collected in a DB. For the storage of the parameters, Influx DB storage technology has been implemented. InfluxDB is used as a data store for cases involving large amounts of timestamped, like monitoring applications or micro-services. The measurements are injected by the Telegraf plugins in the Influx DB database.
- **SLA Assessment** assesses the compliance of the SLA of the contracted services with respect to the values retrieved for the parameters by the Metering sub-component (contracted values vs. real values).
- **Manage Violation** is a sub-component in charge of managing and triggering alerts. This sub-component will alert in two ways, in one hand it will indicate to the service registry subcomponent of ACSmI that a specific cloud service is violated its SLAs, and also this sub-component will alert to ADAPT (Violation handlers) [25].

6 Conclusions

This deliverable presents the first prototype of ACSmI. This first prototype is composed of two main components: 1) *ACSmI discovery* whose objective is to discover services and also to allow the endorsement of cloud services to the service registry by the CSPs and 2) *ACSmI contracting* whose objective is the execution and management of the core functions with respect to the service contracts. In addition to these two main components, the first aspects (Functional and Technical) of the ACSmI monitoring component are detailed.

The details regarding the functional and technical aspects of the components comprising ACSmI are enumerated in this version of the deliverable. Moreover, the package information and manual for installation and for users are included.

The next version of this deliverable, D5.3, will include new functionalities and improvements for the three subcomponents explained, complemented with new subcomponents to cover the requirements elicited for ACSmI.

Future releases will not only focus on the implementation of new functionalities, but it will also be integrated with other DECIDE tools, like OPTIMUS and ADAPT.

References

[1] DECIDE Consortium, "D5.1 ACSmI requirements and technical design," 2017.

[2] DECIDE Consortium, “D3.4 – Initial profiling and classification techniques,” 2017.

[3] JHipster, “The JHipster Registry,” 2017. [Online]. Available: http://www.jhipster.tech/jhipster-registry/. [Accessed November 2017].

[4] JHipster, “JHipster,” [Online]. Available: http://www.jhipster.tech/. [Accessed November 2017].

[5] Spring, “Spring Boot,” 2017, [Online]. Available: https://projects.spring.io/spring-boot/. [Accessed November 2017].

[6] Angular, “Angular Docs,” 2017. [Online]. Available: https://angular.io/. [Accessed November 2017].

[7] Bootstrap , “Bootstrap: The most popular HTML, CSS, and JS library in the world.,” 2017. [Online]. Available: https://getbootstrap.com/. [Accessed November 2017].

[8] Netflix OSS, “Netflix Open Source Software Center,” 2017. [Online]. Available: https://netflix.github.io/. [Accessed November 2017].

[9] Elastic, “The Open Source Elastic Stack,” 2017. [Online]. Available: https://www.elastic.co/products. [Accessed November 2017].

[10] Docker, “Docker,” [Online]. Available: https://www.docker.com/. [Accessed November 2017].

[11] Yeoman, “Yeoman: The web's scaffolding tool for modern webapps,” [Online]. Available: http://yeoman.io/. [Accessed November 2017].

[12] Webpack, “Webpack Module Bundler,” [Online]. [Accessed November 2017].

[13] Gulp, “Gulp: Automate and enhance your workflow,” [Online]. Available: https://gulpjs.com/. [Accessed November 2017].

[14] Apache Maven Project, “Welcome to Apache Maven,” [Online]. Available: https://maven.apache.org/. [Accessed November 2017].

[15] Gradle Inc., “Gradle Build Tool,” [Online]. Available: https://gradle.org/. [Accessed November 2017].

[16] Rails, "Ruby on Rails: A web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern.," 2017. [Online]. Available: http://rubyonrails.org/. [Accessed November 2017].

[17] SQlite, "SQLite Home Page," 2017. [Online]. Available: https://www.sqlite.org/. [Accessed November 2017].

[18] nginx, "nginx," [Online]. Available: https://nginx.org/en/. [Accessed November 2017].

[19] Phusion B.V, "Passenger - Enterprise grade web app server for Ruby, Node.js, Python," 2017. [Online]. Available: https://www.phusionpassenger.com/. [Accessed November 2017].

[20] Windows Azure, "SLA summary for Azure services," Microsoft, 2017. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/summary/. [Accessed 08 November 2017].

[21] AWS, "Amazon S3 Service Level Agreement," [Online]. Available: https://aws.amazon.com/s3/sla/. [Accessed November 2017].

[22] Telegraf, "GitHub telegraf/plugins/inputs/sqlserver," [Online]. Available: https://github.com/influxdata/telegraf/tree/master/plugins/inputs/sqlserver. [Accessed November 2017].

[23] AWS, "Amazon EC2 Service Level Agreement," AWS, [Online]. Available: https://aws.amazon.com/ec2/sla/. [Accessed November 2017].

[24] AWS, "Amazon CloudWatch," AWS, [Online]. Available: https://aws.amazon.com/cloudwatch/. [Accessed November 2017].

[25] DECIDE Consortium, "D4.7 Initial multi-cloud application monitoring," 2017.

[26] Telegraf, "Telegraf is the Agent for Collecting & Reporting Metrics & Data," [Online]. Available: https://www.influxdata.com/time-series-platform/telegraf/. [Accessed November 2017].

APPENDIX 1: Cloud Services Attributes

Table 7. Explanation on the Services Attribute

Attribute Type	Description or possible Values
Name	Id of the services
Region:	Choose a region of the following list: United States, Canada, Europe, Russian Federation, United Kingdom, Ireland, France and Brazil.
Provider	Organisation who provides the services
Type (Storage)	This attribute could have two values at this moment: General and Backup
Type (DB)	This attribute could have one the DB Value at this moment
Subtype(Storage):	The value of this attribute should be chosen from the following list: Objects, Tables, files, queues and disk
Subtype(DB)	The value of this attribute should be chosen from the following list: Relational, Non-Relational, Elastic cache, Data migration.
Availability Zone	The value of this attribute should be chosen from the following list: Single and Multiple
Technology	The value of this attribute should be chosen from the following list: R-Oracle, R-MSQLServer, R-PostgreSQL, R-MariaDB, R-MySQL, NR-Amazon Dynamo, NR-TableStorage, EC-AmazonEC, EC-AmazonRD, EC-MicorsoftRC, AmazonMigration, and DataFactory.
License	The value of this attribute should be chosen from the following list: Included and Not included
Instance Type	The value of this attribute should be chosen from the following list: Reserved and On demand
Size	The value of this attribute should be chosen from the following list: Storage Size and Cache size
CPU	This attribute is a number field.
Capacity	The value of this attribute should be chosen from the following list: Minimum, medium and maximum
Access type	The value of this attribute should be chosen from the following list: Frequent and Infrequent
Data Redundancy	The value of this attribute should be chosen from the following list: locally redundant storage (LRS), Zone-redundant storage (ZRS), Geo-redundant storage (GRS) and Read-access geo-redundant storage (RA-GRS)
CPU cores	It is the number of cores that are available in each of the instance type´s CPU sockets defined in CPU.
Total Cores	This is the total number of cores that will be used by the platform for this service. It is computed from the number of CPUs times the number of CPU cores times two in case of hyper-threading
Hyper-threading	It specifies if hyper-threading is turned on for this instance type and should be used by the platform for counting the total number of cores. The total number of cores is then doubled compared to the number without hyper-threading in place
MHz per core	This attribute is a number field.
Memory	It specifies the amount of memory (in GB)
Storage (GB per month)	This attribute is a number field.
Data transfer out (GB per month)	This attribute is a number field.