Deliverable D5.3

Intermediate Advanced Cloud Service meta-Intermediator

Editor(s):	Marisa Escalante
Responsible Partner:	TECNALIA
Status-Version:	Final – v1.0
Date:	30/11/2018
Distribution level (CO, PU):	CO

Project Number:	GA 726755
Project Title:	DECIDE

Title of Deliverable:	D5.3 – Intermediate Advanced Cloud Service meta-Intermediator
Due Date of Delivery to the EC:	30/11/2018

Workpackage responsible for the Deliverable:	WP5 – Continuous cloud services mediation
Editor(s):	TECNALIA
Contributor(s):	Vitalii Zakharenko, Andrey Sereda (CB) Maria Jose Lopez, Marisa Escalante Gorka Benguria Leire Orue-Echevarria, Juncal Alonso, Iñaki Etxaniz, Alberto Molinuevo (TECNALIA)
Reviewer(s):	Manuel León; 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	01/10/2018	ToC	TECNALIA
V0.1	5/11/2018	ACSmI discovery, ACSmI monitoring and ACSmI Legal sections first version	TECNALIA & Timelex
V0.2	12/11/2018	ACSmI monitoring and Discovery section second version	TECNALIA
V0.3	16/11/2018	ACSmI contracted and Billing sections and updates in the other section	CB &TECNALIA
V0.4	19/11/2018	Version sent to internal review	TECNALIA & CB & timelex
V0.5	21/11/2018	Reviewed	ARSYS
V0.6	22/11/2018	Implemented the changes suggested by the reviewer	TECNALIA
V1.0	26/11/2018	Ready for submission	TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

ABC	ACSmI Business Component
ACC	ACSmI Contracting Component
ACSmI	Advanced Cloud Service meta Intermediator
ADAPT DO	ADAPT Deployment Orchestrator
API	Application Programming Interface
CRUD	Create, Read, Update and Delete
CS	Cloud Service
CSP	Cloud Service Provider
DB	Database
EC	European Commission
NFR	Non-Functional Requirement
SLA	Service Level Agreement
SLO	Service Level Objective
UI	User Interface

Executive Summary

This document describes the M24 prototype of ACSmI. The general objectives of this M24 prototype are to discover and benchmark services based on a set of requirements, to monitor the contracted cloud services to assess if they fulfil the SLAs recorded in the registry, to facilitate the contract of the cloud services selected and to manage the connectors to access to these cloud services to deploy the application. This is realized by three main components: ACSmI discovery, ACSmI monitoring and ACSmI contracting and complemented with the first design of ACSmI billing and the approach to be followed to assess the legal compliance.

The objective of ACSmI discovery is to discover and benchmark the services stored in the ACSmI registry 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. The objective of ACSmI monitoring is to assure that the SLAs of the NFRs for each of the cloud services are fulfilled and if not, to alert the non-fulfilment to the relevant stakeholders.

This deliverable is an update of deliverable D5.2 – "Initial Advanced Cloud Service meta-Intermediator" [1] and therefore reuses and updates the previous results and content.

This M24 prototype includes:

- New functionalities to the previous prototype in ACSmI discovery and ACSmI contracting components such as the benchmarking of the services and new CSP connectors
- Improvements to the existing functionalities in the M12 components
- Implementation of ACSmI monitoring that allows to assess the cloud service SLAs of the NFRs such as availability, performance and location and to alert if a violation occurred to ADAPT Violation Handler and ACSmI registry.
- Definition of the approach to be followed in the implementation of the billing component and the legislation compliance.

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 next release of ACSmI, which is due in M30, including the new functionalities implemented in each component.

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 addressing the implementation and usage details of the ACSmI Discovery, ACSmI contracting and ACSmI monitoring prototypes and the design of the legislation compliance and billing functionalities. It is divided into six main sections describing the general implementation of ACSmI, the five components of the prototype, the architectural and implementation details of ACSmI discovery, contracting and monitoring components and how to use and download them. Section 6 and section 7 describe the first approach and design to be implemented for M30 for the ACSmI billing and ACSmI legal compliance.

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

This section introduces the functional (section 5.2) and technical (section 5.3) description of the ACSmI monitoring prototype while sub-section 5.4 provides a description of the delivery and usage of the prototype, including installation and downloading instructions.

Section 6 ACSmI Billing

This section introduces the main functionalities and the first approach for implementing them for the ACSmI billing component.

Section 7 ASCmI Legal

This section introduces the approach to be followed to carry out the legal compliance of the services in the ACSmI registry.

To finalize the deliverable, the conclusions are presented, and several annexes are included.

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.

As explained in [1], 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 registration 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.

Functionality that this prototype offers:

The delivered prototype in the second version of ACSmI contains the following functionality:

F1. **Discovery of services** stored in the service registry. This release of the component implements the discovery of the services based on NFRs like availability, performance, location and cost. In this prototype, several APIs have been implemented to be integrated with:

- OPTIMUS [2]: ACSmI provides OPTIMUS with the services that are in the registry that fulfil the requirements. Not only the services that fulfil completely all requirements are selected, but rather ACSmI presents a short list with the information of the requirements fulfilled and the coverage degree.
- MCSLA [3]: ACSmI provides MCSLA with the information regarding the SLOs of the NFRs of the service that has been selected.
- ACSmI contracting collects the information that it requires to contract the services through and API exposed by ACSmI Discovery.

F2. **Endorsement of services** in the service registry. The fields to be completed are adapted depending on the type of service to be endorsed. This release of ACSmI improves the endorsement process collecting the information required to complete the SLOs for NFRs like availability and performance but not only, it also allows to provide information (attaching contracts and other information) that enables the legal experts to assign a legal level based on the analysis of this information. The registry also records the SLA violation of each service, as this information is provided by ACSmI monitoring

F3. 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. This release allows to contract in an automatic way services of the following CSPs: Amazon, Cloud Sigma and Arsys. Also, it has included the connectors needed to facilitate the start-up of the contracted services.

F4. **Monitoring of the CSPs**. This component implements the detection of any violation according to the SLOs of the NFRs and the alerting mechanism. Two types of alerts are implemented: 1) To the violation handler (of ADAPT), and to the ACSmI registry. In order to detect the violations, this component meters the behaviour of the services, records these metrics in a Time series database (see more detailed information in section 5) and assesses the fulfilment against the SLA of the NFRs.

This document also presents a design for the implementation of the billing component that will be released in M30.

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 mainly in the phase of multi-cloud applications continuous delivery (Figure 1):

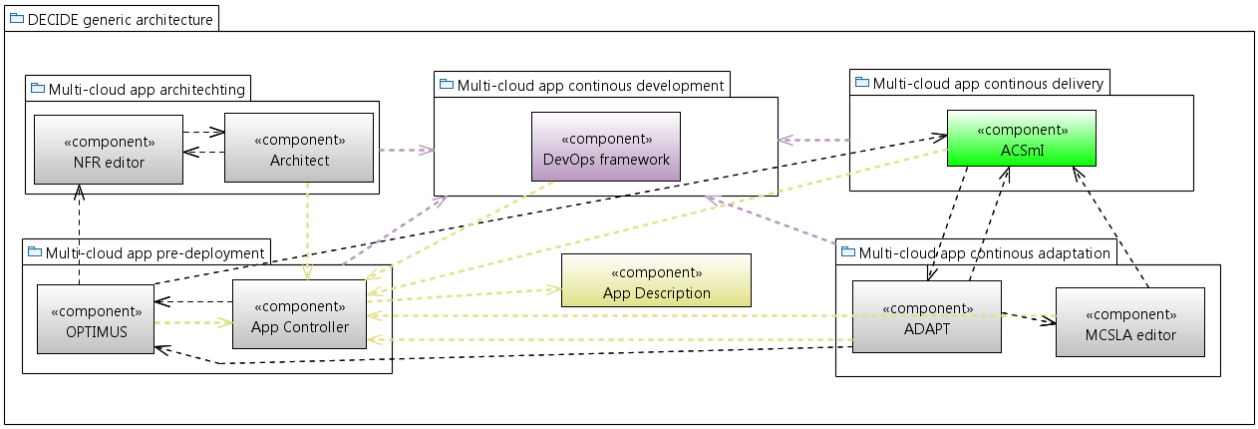


Figure 1. ACSmI in DECIDE architecture

ACSmI interacts with other tools in the DECIDE ecosystem (see Figure 2). To this respect,

- ACSmI communicates with OPTIMUS to gather the non-functional requirements that the discovered services have to fulfil. ACSmI will return OPTIMUS the list of services available with the information of the fulfilment (percentage and covered requirements) in the service registry fulfilling the entered requirements.
- ACSmI informs MCSLA [3] about the SLOs for the NFRs of the services selected for the deployment.
- ACSmI receives information from the Application Controller [4] regarding the services that will have to be contracted and will return the Application Controller the final information about the contracted cloud services.
- DevOps framework interacts with ACSmI to support the user management functionalities for the profile of developers and operators. ACSmI manages its own users for the profiles of CSP.
- Finally, ACSmI will interact with ADAPT with the following objectives:
  1. to provide the information of the cloud services to allow ADAPT deploying the multi-cloud application [5], and
  2. to interchange information regarding the violation of the SLAs [6].

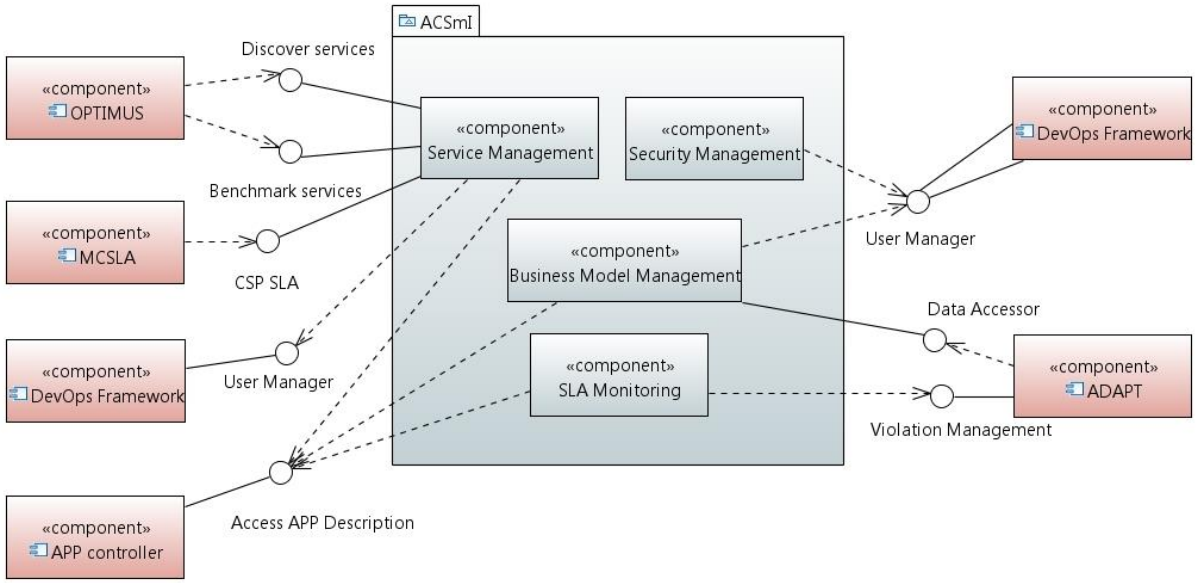


Figure 2. ACSmI Interfaces with other DECIDE tools

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.

From the functionalities detailed in section 2.1, this component implements 'F1. Discovery of services' and 'F2. Endorsement of services.'.

Requirements covered by the prototype:

The following table details the requirements covered in this prototype based on the ACSmI requirements listed in the D2.2 [1].

Table 1. Requirements covered by the M24 ACSmI Discovery prototype.

Req. ID	Req. Description	Requirement coverage by the prototype
WP5-DIS01	CSPs or the ACSmI administrator (for Large CSPs) register(s) its 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.	This prototype allows the endorsement of three different types of service classes: • Virtual machine • DB • Storage The prototype will request certain information, depending on the service class to the CSP. The information collected in this step is that information required from other components or tools to carry out their activities. The information collected by the endorsement functionality is used by OPTIMUS, MCSLA, other components of ACSmI.
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 defined for the modelling of the CSPs and their services. This allows for an automatic comparison of the requirements with the services stored in the registry. The communication with OPTIMUS will be done through an API provided by OPTIMUS	ACSmI provides a REST API to OPTIMUS that enables to detail the characteristics that the cloud services should have in order to be used. This API allows to look for FR and NFR like availability or certification schemes if these are the requirements of the developer. The terms for the modelling of the cloud services are specified in D3.5 [2] but it is important to have in mind that this approach has been implemented to allow to add new terms to the model in an easy way. ACSmI also provides an UI to allow the user to search himself.

Req. ID	Req. Description	Requirement coverage by the prototype
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 developer and operator.	In this prototype, the discovery functionality has been updated considered the comparison of the units of the terms, or the nature of the term, (i.e 500 MB= 0,5 GB; or if required availability is 95%, all the services with the availability major than 95% are presented).
WP5-DIS05	The discovered services (WP5-DIS03) shall be prioritized. Depending on the level of fulfilment of the NFRs expressed by the DECIDE operator, the discovered services will be sent back to DECIDE operator in the form of a sorted list, indicating the degree of fulfilment.	In this release, the discovered services are shown or sent to OPTIMUS with a percentage of the fulfilment and a list of the NFRs that are fulfilled
WP5-DIS06	The objective is to provide means to create, read, update and delete (CRUD) the users´ registry. When creating a new user, a role shall be assigned to him, and based on this role, the allowed activities to be performed shall be associated to this user.	The management of the user with the role of developer or multi-cloud application owner is going to be carried out by the DECIDE Framework. ACSmI takes care the management of the CSPs users
WP5-DIS07	The 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.	ACSmI registry records information about the SLA violation: Time and type of the violation and allows to upload the required information to be analysed by a legal expert to assign the legal level.
WP5-DIS08	The objective is to handle the dashboard that shall be personalised depending on the role of the ACSmI users. ACSmI shall customise the dashboard to show users only the allowed tasks to be performed.	ACSmI allows to perform different tasks depending the role that the user has assigned. At this moment, there are 4 roles identified: Admin, CSP, Developer/operator/user and Legal expert.
WP5-LEG01	ACSmI shall be able to show what the legal level is of the cloud service in question. The legal level will be defined through assessing all the legally relevant aspects when initiating a service, in particular in regard to location of data, data security level, location of the service provider etc., in order to enable the cloud consumer to assess the legal impact of initializing and operating the proposed service.	This ACSmI Discovery prototype allows when endorsing a service, on one hand to indicate the accreditations that the CSP has like, ISO 27001, EU – Wide Certification schemes, and the other hand, when a CSP is endorsing a cloud service, he should upload the following documents that will allow the legal expert to assess the legal level. These documents are: • The service contract applicable to the service

Req. ID	Req. Description	Requirement coverage by the prototype
	It will also specifically focus on the terms that regulate the termination of a service, e.g. data format on exit, data portability, etc. This will also be factored into the legal level of the service.	• The SLA applicable to the service • The Data processing agreement governing the service The questionnaire to be answered by the legal expert and the logic to derive the legal level will be ready in the next version.

3.2 Technical description

The discovery and endorsement of the services will be performed based on common attributes for each service type class. Next figure shows the class model of the ACSmI discovery component.

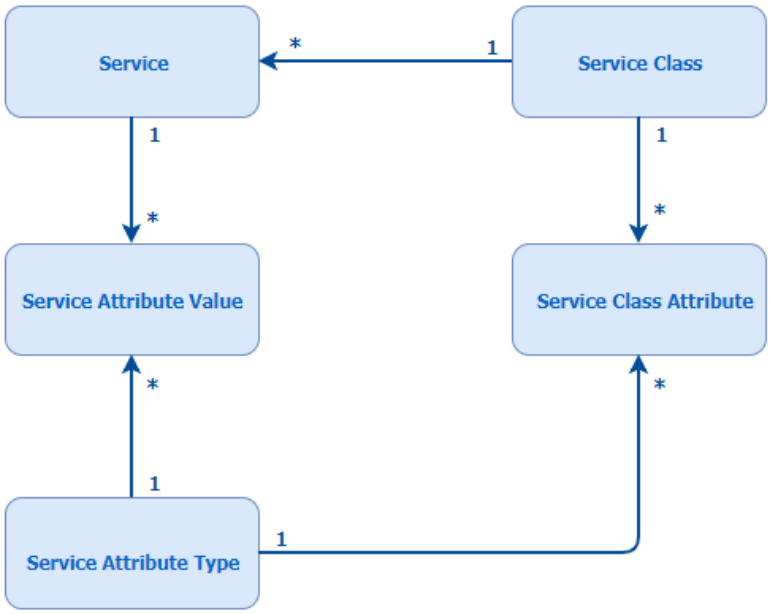


Figure 3. ACSmI discovery class service model

This model is a bit different from the one used in the M12 prototype, as in this one, the Service Class Attribute has been included. The Service Class Attribute entity determines what attribute types belong to each Service Class.

For this M24 prototype, three service classes have been defined with the corresponding service attributes, namely “Storage”, “Database”, and “Virtual machine”. The table below shows the relationship between the service type and the attributes. Any CSP that wants to endorse their cloud services in the service registry is required to provide this information.

Table 2. Attributes vs. Cloud services Classes

Storage	DataBase	Virtual Machine
Region *	Region *	Region *
Zone	Zone	Zone
Provider*	Provider *	Provider *
Storage type *	Database type *	Virtual CPU cores *
Storage subtype *	Database technology *	Frequency per core

Storage	DataBase	Virtual Machine
Storage capacity *	Data transfer IN *	Memory *
Storage data redundancy *	Data transfer OUT *	Instance storage
Availability *	Virtual CPU cores *	Optimized for
Request – Response time: Storage Performance *	Database storage capacity *	Public IP
Legal certifications/ accreditations *	Availability *	Underpinning technology
Cost/Currency	Transaction Unit (DTU): Database performance *	Availability*
	Legal certifications/ accreditations *	Response time: Virtual Machine Performance *
	Cost/Currency	Legal Level *
		Cost/Currency

* Means mandatory field

3.2.1 Prototype architecture

In the current prototype (M24), the discovery component is structured into four main components, as shown in the next figure.

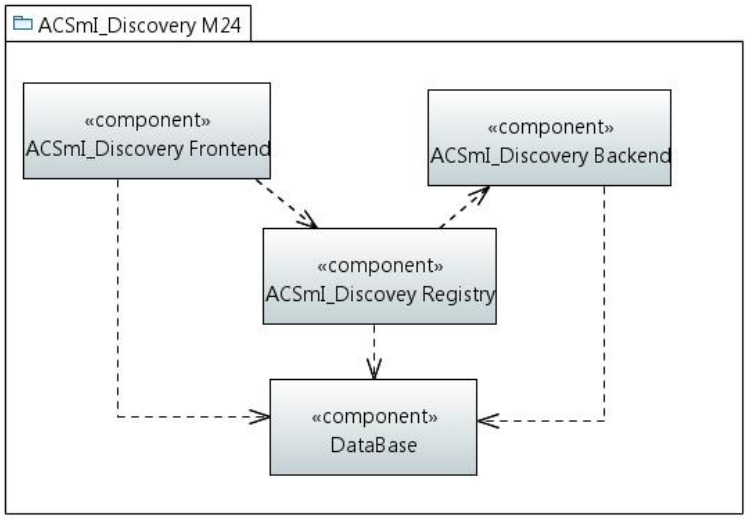


Figure 4. ACSmI Discovery M24 High-Level Architecture

- *Frontend*. This component is in charge of managing all the aspects related to the ACSmI interface as well as to the user management. Four roles have been defined: Admin, User, CSP and Legal expert, although when ACSmI Discovery is used together with the DevOps framework the management of the User role for developers and operators is done by DevOps Framework. 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, benchmark 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 and all information related to them.

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.

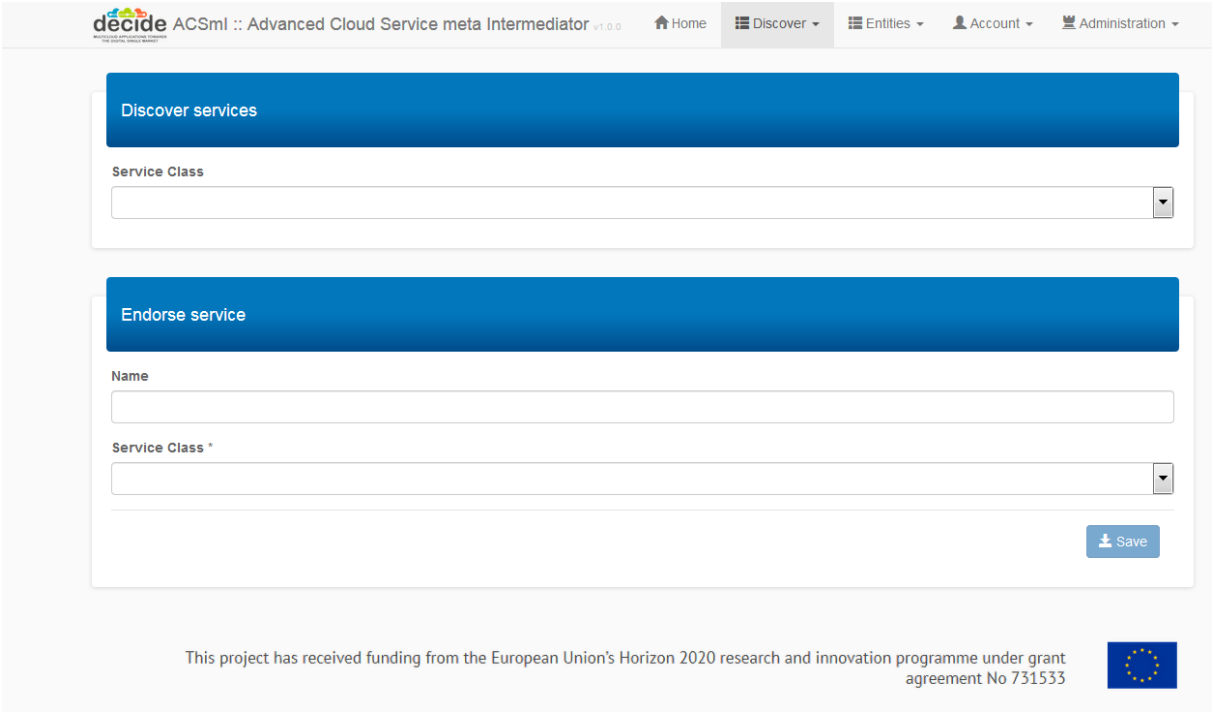


Figure 5. Discover and Endorse services functionalities – Admin view

2. To manage users. This component will enable to create, read, update and delete users in ACSmI discovery. At this moment, four different user roles for the user have been identified:
    - Admin. It is the responsible of the management of the ACSmI.
    - User. This role will be able to carry out the discovery of the services. The management of this role is done by the DevOps framework when ACSmI is used through it.
    - CSP. This role will be able to carry out the endorsement of the services providing all the information required
    - Legal Expert. This role is responsible to assess the documents provided by the CSP and to assign a legal level.

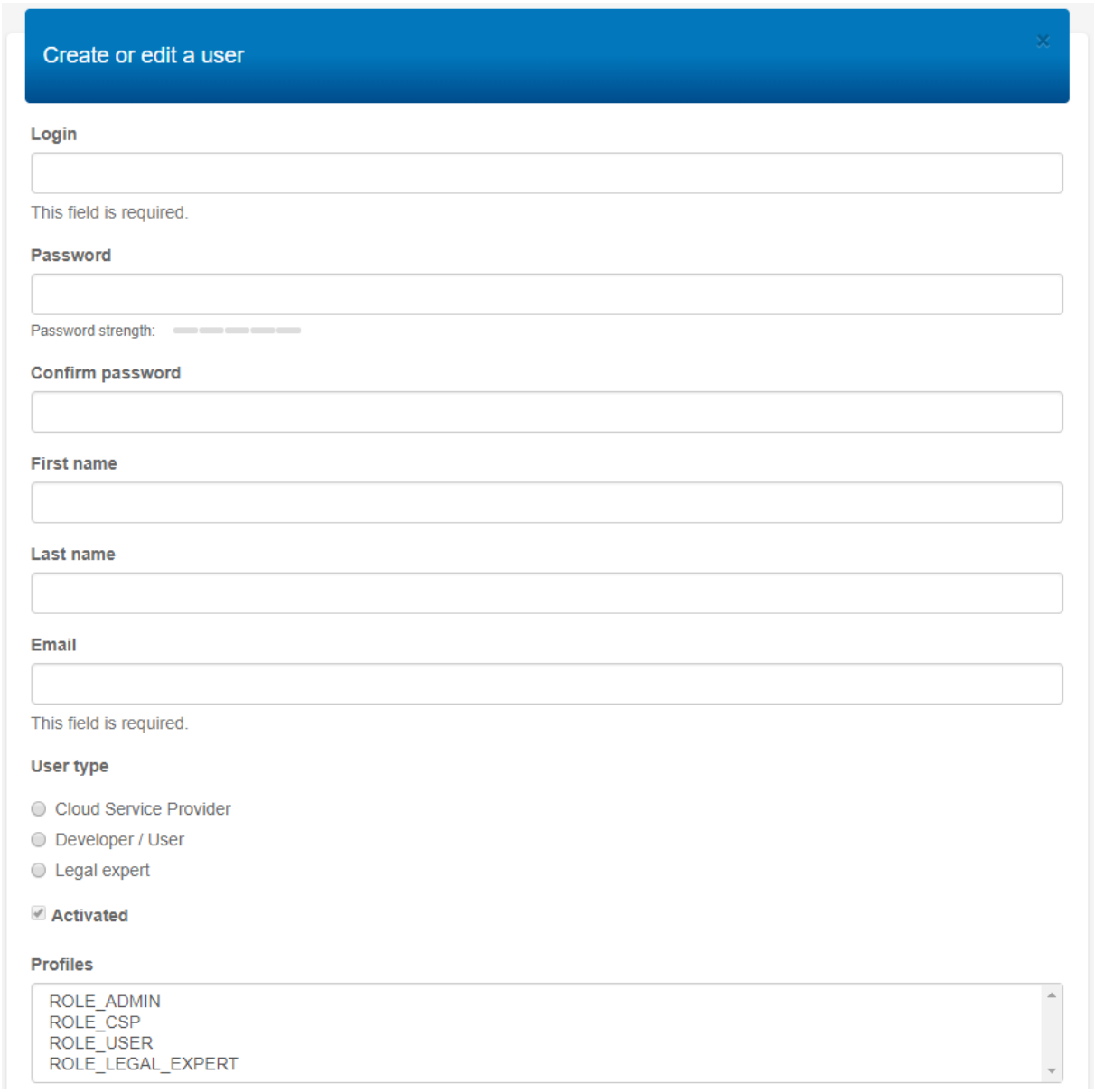


Figure 6. Create or edit a user

In addition to the graphical interface, this component offers the interfaces shown in Figure 7 that allows automatizing some functionalities and allow other tools to interact with this component. These tools are mainly OPTIMUS and MCSLA. These interfaces also allow to interact with ACSmI contracting and ACSmI monitoring.

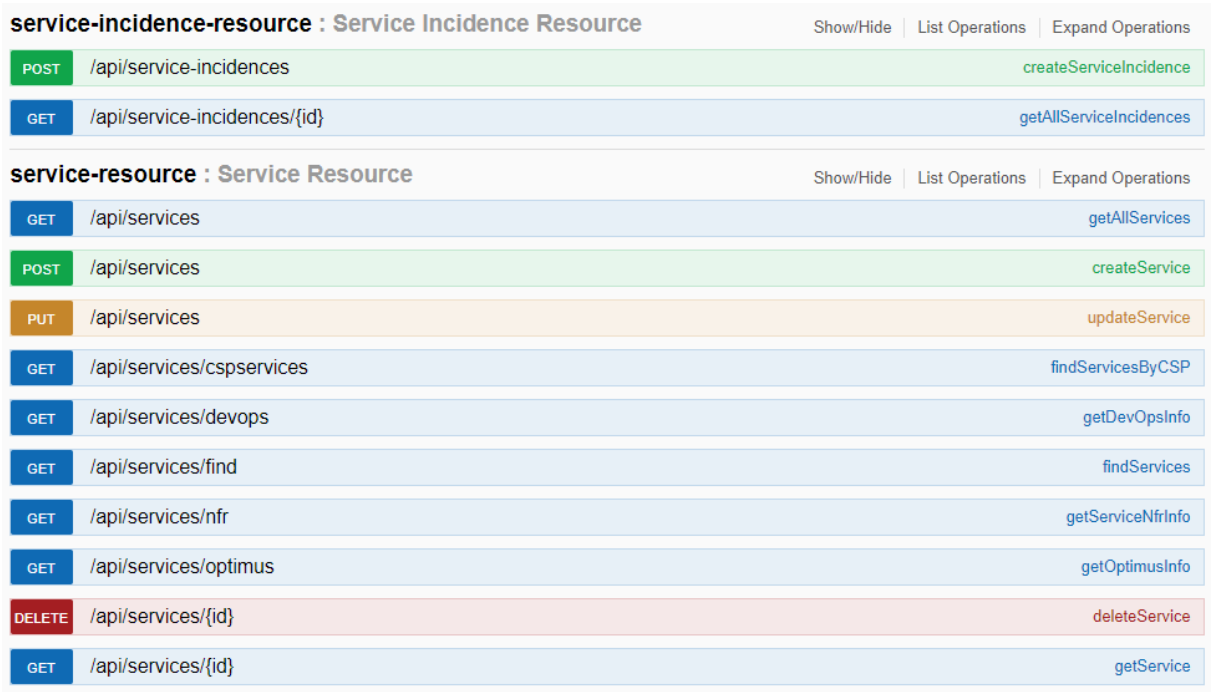


Figure 7. Frontend API Information.

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, 2) benchmarking each cloud service indicating the degree of fulfilment and the attributes that are fulfilled and 3) the endorsement of the services based on the information provided by the CSPs 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. There are two ways to introduce the information for the discovery: 1) through the graphical interface in order to be able to access to this functionality the developer should have assigned a “user” role or 2) through OPTIMUS using the interface “getOptimusInfo”.

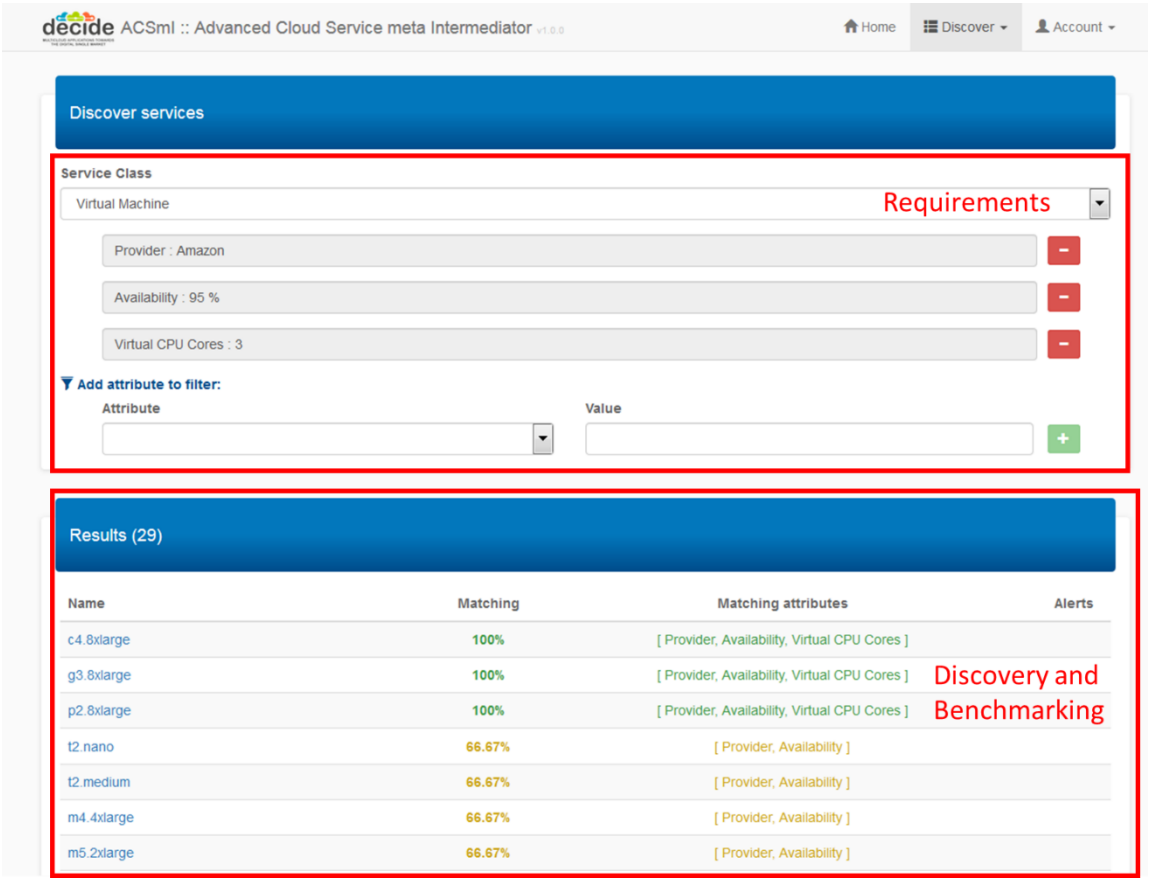


Figure 8. Example result of a discovery query.

Endorsement of services

This functionality allows to insert all the information required to add a service into the ACSmI service registry. The registry will have some mandatory information complemented with additional information, which is used to benchmark the different services. The registry follows the model explained previously in this section. The attributes to be collected have been classified in three types:

- Common attributes
- Functional Requirements (FR) attributes
- Non-functional Requirements (NFR) attributes

In addition, service contracts are collected. There are three types of contracts:

- The overarching services contract (+ volume order).
- The SLA for the specific service.
- The data processing agreement (which might be integrated in another contract).

In order to be able to access to this functionality, it is required to have a user with the “CSP” or “Admin” role.

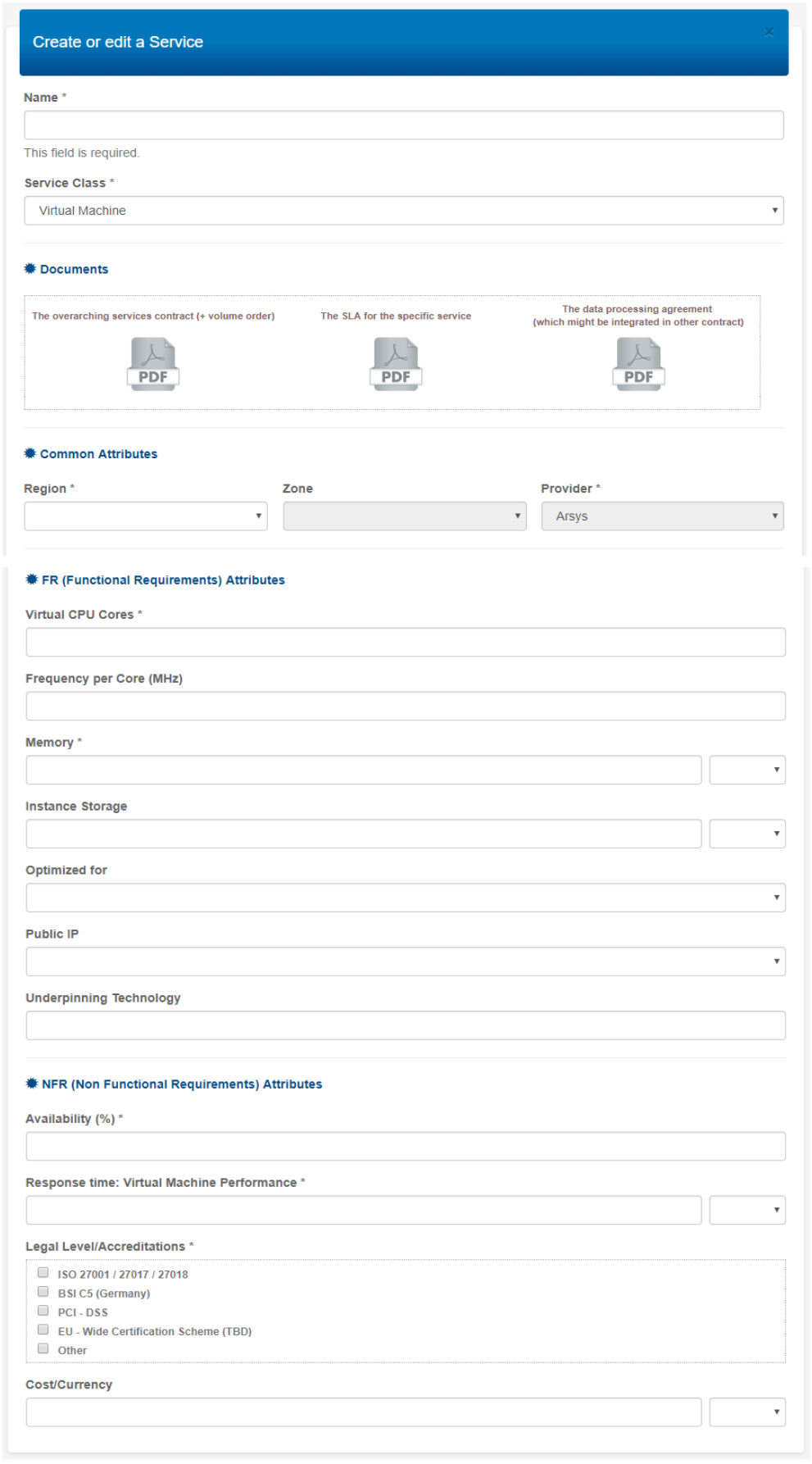


Figure 9. Example of the endorsement of a service.

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.

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 provides 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 an 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 complete web application or micro-service architecture, unifying:

- A high-performance Java stack on the server side with Spring Boot [5]
- A sleek, modern, mobile-first frontend with Angular [6] and Bootstrap [7]
- A micro service architecture with JHipster Registry [3], Netflix OSS [8], ELK stack [9] and Docker [10]
- A 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.

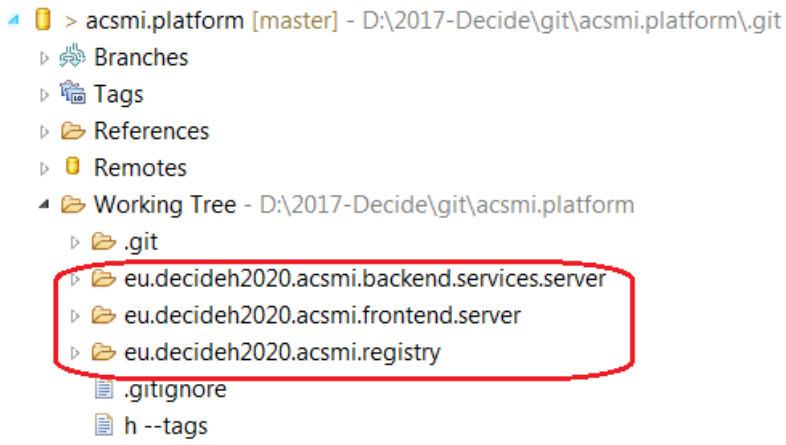


Figure 10. Package Structure

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.

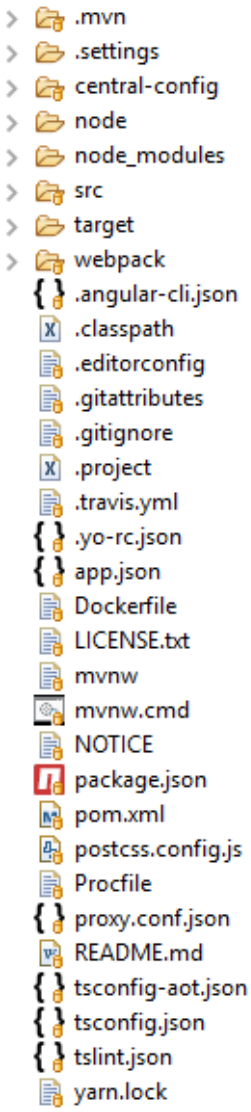


Figure 11. Registry component packages structure

- *eu.decideh2020.acsmi.frontend.server.* This component is composed of one Maven project, and the main technologies used in this component are AngularJS + Spring REST. The main structure of this project is:

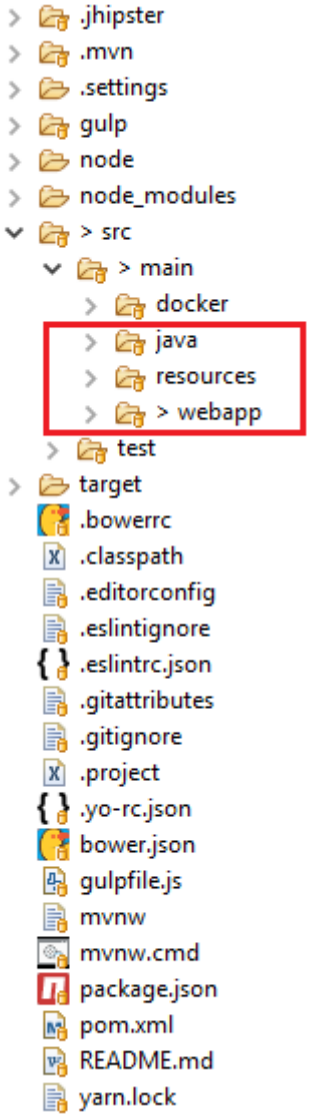


Figure 12. Frontend component packages structure

The “java” folder contains the code to manage the REST API supplied:

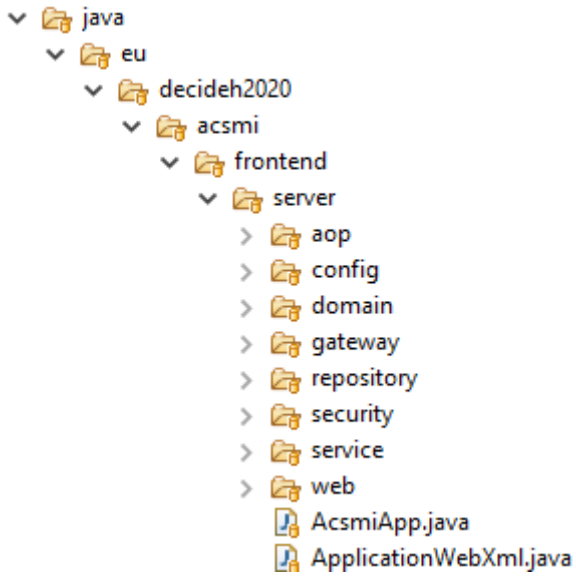


Figure 13. The "java" folder structure

The "resources" folder contains the configurations for the application and the database. This database controls the users and their roles:

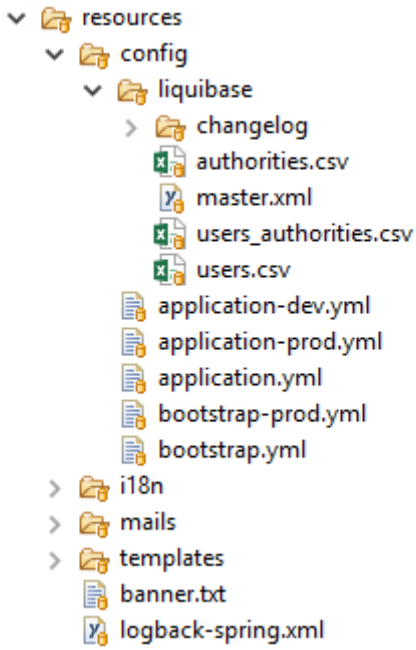


Figure 14. The "resources" folder structure

Finally, the "webapp" folder contains the HTML pages and their logic in AngularJS Javascript files:

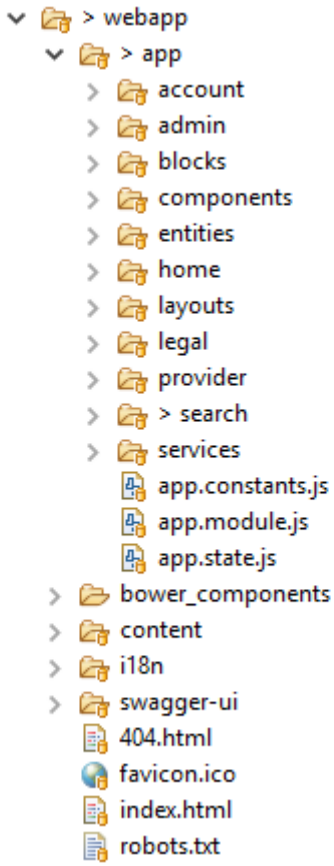


Figure 15. The "webapp" folder structure

Each action is organized in a folder. For example, the Service Discovery Action has the "search" folder.

These packages are accomplished with a database server, detailed in Figure 20.

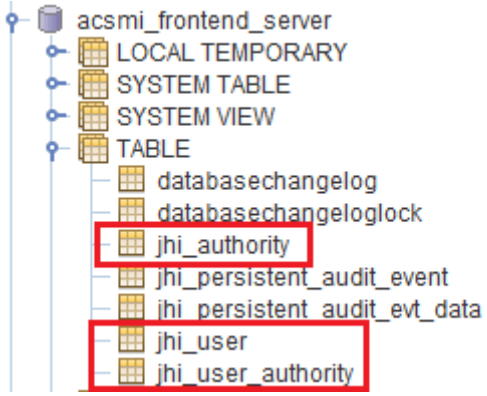


Figure 16. Front-end project database

- *eu.decideh2020.acsmi.backend.services.server*. This project is in charge of the core functionalities of the prototype and contains the source code for the implementation of these functionalities. It is composed of one Maven project, and the main technology used in this component is Spring REST. The main structure of this project is:

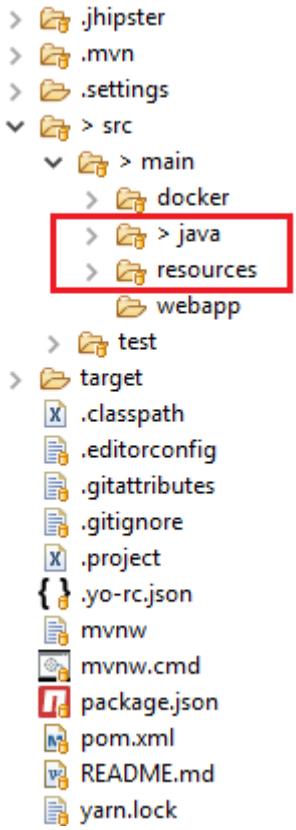


Figure 17. Back-end component packages structure

The “java” folder contains the code to manage the REST API supplied:

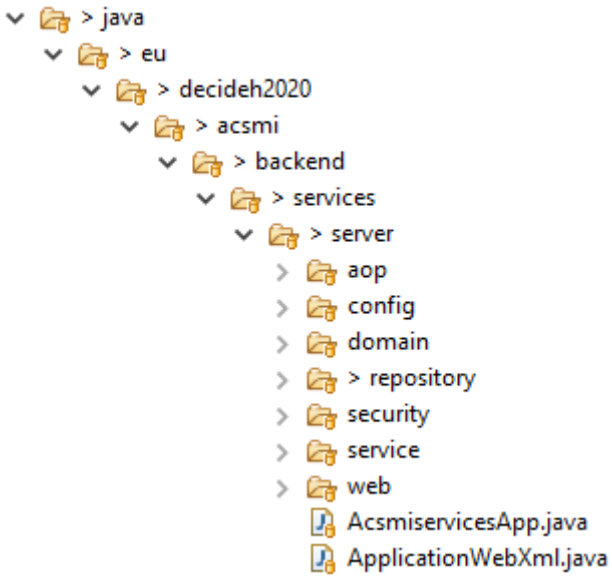


Figure 18. The “java” folder structure

The “resources” folder contains the configurations for the application and the database. This database is used to manage the services and their attributes and incidences:

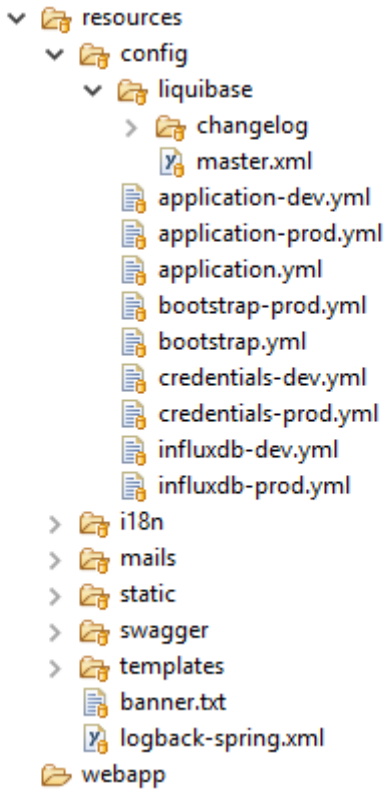


Figure 19. The "resources" folder structure

These packages are accomplished with a database server, detailed in Figure 20.

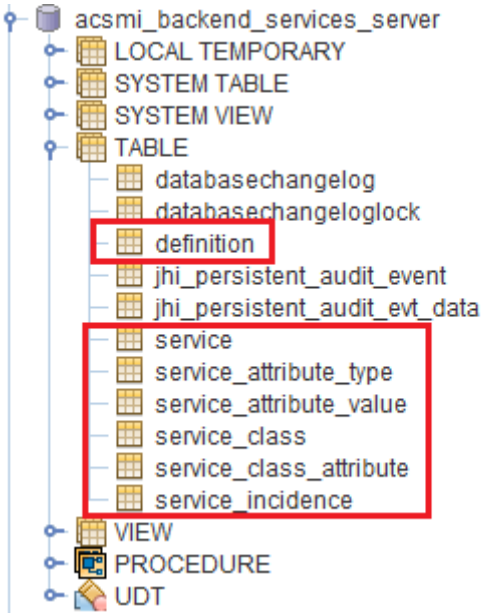


Figure 20. Back-end project database

3.3.2 Installation instructions

The prototype has been installed and tested using the following software (see Pre-requirements section):

- Docker to create the containers of the different sub-components.

The following steps need to be executed to get the prototype up and running:

1. Download the src from the DECIDE repository.

```
git clone
https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git
```

2. Each of the containers of the different sub-components needs to be built, in the following order:
   - MySQL container.
   - ACSmI Discovery Registry container.
   - ACSmI Discovery Backend container.
   - ACSmI Discovery Frontend container.
   - Database Test container.

a. MySQL container

i. Go to the folder where the docker file for MySQL is located:

```
cd <ACSmI discovery folder>\
eu.decideh2020.int.acsmi.mysql.server.src.dvp\src\main\docker
```

ii. Build the docker image for MySQL with the following arguments:

```
docker build -f /docker.acsmi.mysql.server -t
tecnalia/eu.decideh2020.int.acsmi.mysql.server
```

iii. Run the docker image with the following arguments:

```
docker run -d -p 3306:3306 --name
eu.decideh2020.int.acsmi.mysql.server --env
MYSQL_ALLOW_EMPTY_PASSWORD=yes
tecnalia/eu.decideh2020.int.acsmi.mysql.server
```

This container will support the main databases of the application. It is needed to pass as environment variable the following key / value pair: "MYSQL_ALLOW_EMPTY_PASSWORD=true".

b. ACSmI Discovery Registry container

i. Go to the folder where the docker file for ACSmI Discovery Registry is located:

```
cd <ACSmI discovery
folder>\eu.decideh2020.acsmi.registry.server.src.dvp\src\main\do
cker
```

ii. Build the docker image for ACSmI Discovery Registry with the following arguments:

```
docker build -f /docker.acsmi.registry.server -t
tecnalia/eu.decideh2020.acsmi.registry.server
```

iii. Run the docker image with the following arguments:

```
docker run -d --restart=always -p 8761:8761 --name
eu.decideh2020.acsmi.registry.server
tecnalia/eu.decideh2020.acsmi.registry.server
```

c. ACSmI Discovery Backend container

i. Go to the folder where the docker file for ACSmI Discovery Backend is located:

```
cd <ACSmI discovery folder>\
eu.decideh2020.acsmi.backend.services.server.src.dvp\src\main\do
cker
```

ii. Build the docker image for ACSmI Discovery Backend with the following arguments:

```
docker build -f /docker.acsmi.backend.services.server -t
tecnalia/eu.decideh2020.acsmi.backend.services.server
```

iii. Run the docker image with the following arguments:

```
docker run -d --restart=always --name
eu.decideh2020.acsmi.backend.services.server --add-host
mysql:172.18.0.1 --add-host registry:172.18.0.1
tecnalia/eu.decideh2020.acsmi.backend.services.server
```

ACSmI Discovery Backend needs to know the host of MySQL and the host of ACSmI Discovery Registry. They are specified using "--add-host" parameter.

d. ACSmI Discovery Frontend container

i. Go to the folder where the docker file for ACSmI Discovery Frontend is located:

```
cd <ACSmI discovery folder>\
eu.decideh2020.int.acsmi.frontend.server.src.dvp\src\main\docker
```

ii. Build the docker image for ACSmI Discovery Frontend with the following arguments:

```
docker build -f /docker.acsmi.frontend.server -t
tecnalia/eu.decideh2020.acsmi.frontend.server
```

iii. Run the docker image with the following arguments:

```
docker run -d -p 12080:8080 --restart=always --name
eu.decideh2020.acsmi.frontend.server --add-host
mysql:172.18.0.1 --add-host registry:172.18.0.1
tecnalia/eu.decideh2020.acsmi.frontend.server
```

ACSmI Discovery Frontend needs to know the host of MySQL and the host of ACSmI Discovery Registry. They are specified using "--add-host" parameter.

e. Database Test container

i. Go to the folder where the docker file for Database Test is located:

```
cd <ACSmI discovery folder>\
eu.decideh2020.int.acsmi.backend.services.server.test.00.src.dvp
\src\main\docker
```

ii. Build the docker image for Database Test with the following arguments:

```
docker build -f /docker.acsmi.backend.services.server.test.00 -
t tecnalia/eu.decideh2020.acsmi.backend.services.server.test.00
```

iii. Run the docker image with the following arguments:

```
docker run -d --name
eu.decideh2020.acsmi.backend.services.server.test.00 --
restart=no --env MYSQL_DB_USER=root --env
MYSQL_DB_NAME=acsmi_backend_services_server --env
```

```
MYSQL_DB_HOST=172.18.0.1
tecnalia/eu.decideh2020.acsmi.backend.services.server.test.00
```

It is needed to pass as environment variables the following key / value pairs: "MYSQL_DB_USER=root", "MYSQL_DB_NAME=acsmi_backend_services_server" and "MYSQL_DB_HOST=172.18.0.1".

3.3.2.1 Pre-Requirements

The following lists the minimum requirements to get the component working.

- Machine or virtual machine with the O.S (Ubuntu Xenial 16.04) and the Docker server.
- JRE

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
Administrator	admin	decide
CSP	csp	decide
Legal Expert	legal	decide

1. Clicking on "Account → sign in" on the navigation bar, the following form is shown:

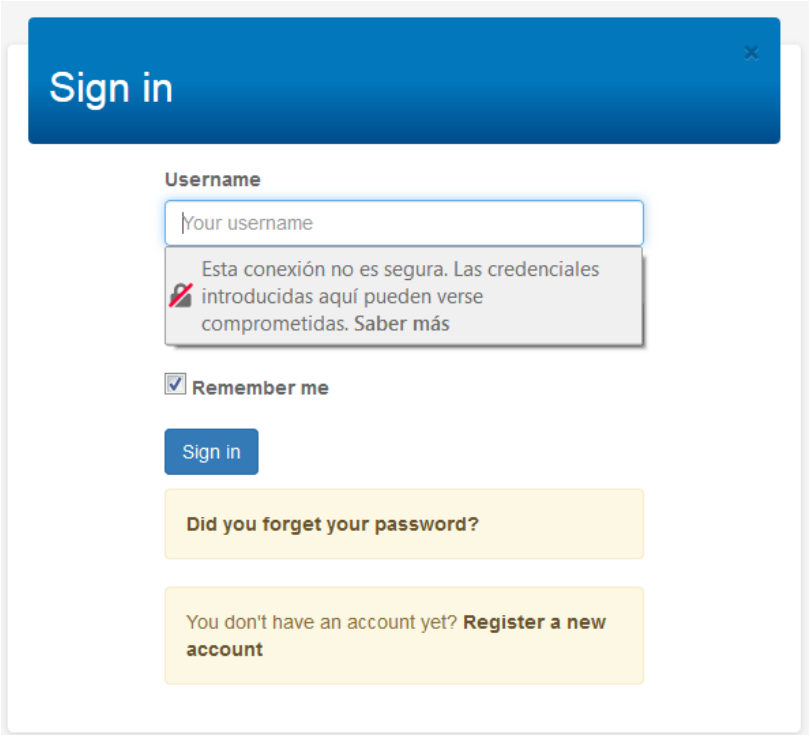


Figure 21. ACSmI login interface

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

The administrator will have access to all tasks:

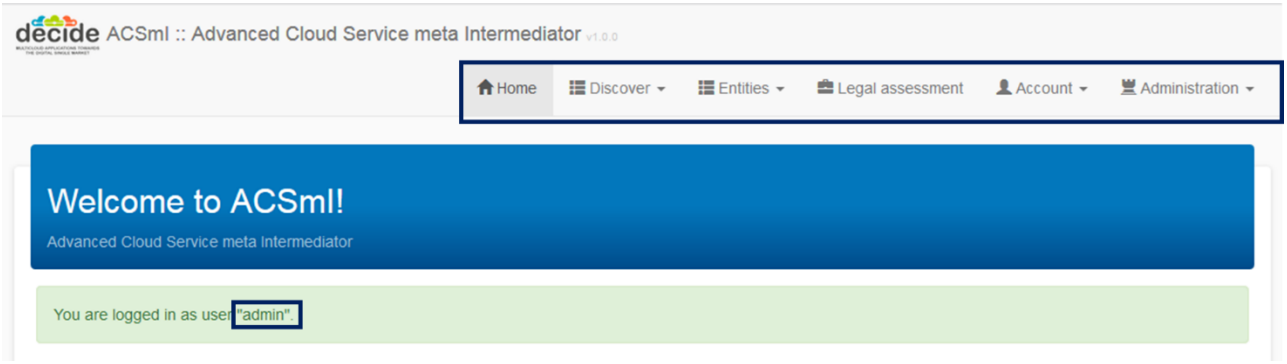


Figure 22. GUI for admin

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

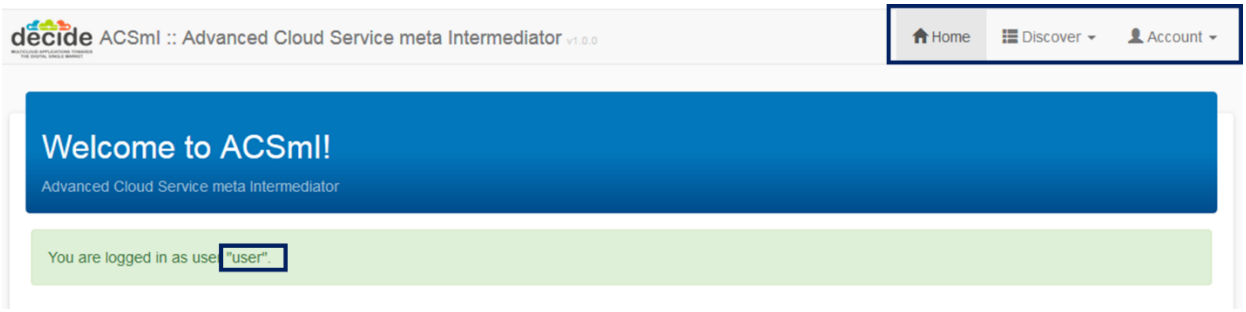


Figure 23. GUI for users

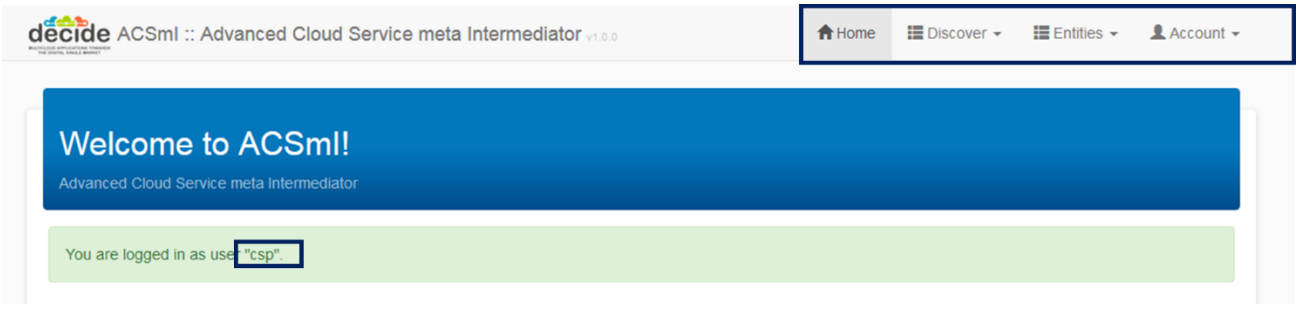


Figure 24. GUI for CSPs

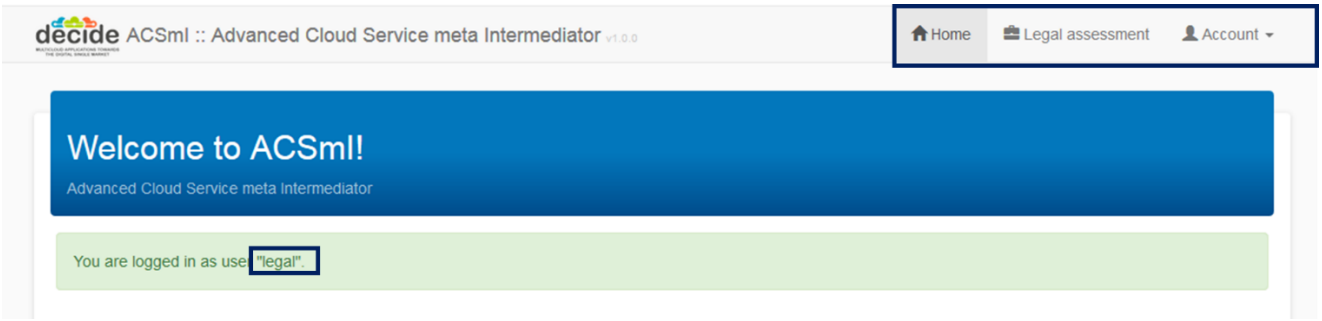
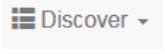


Figure 25. GUI for Legal experts

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

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

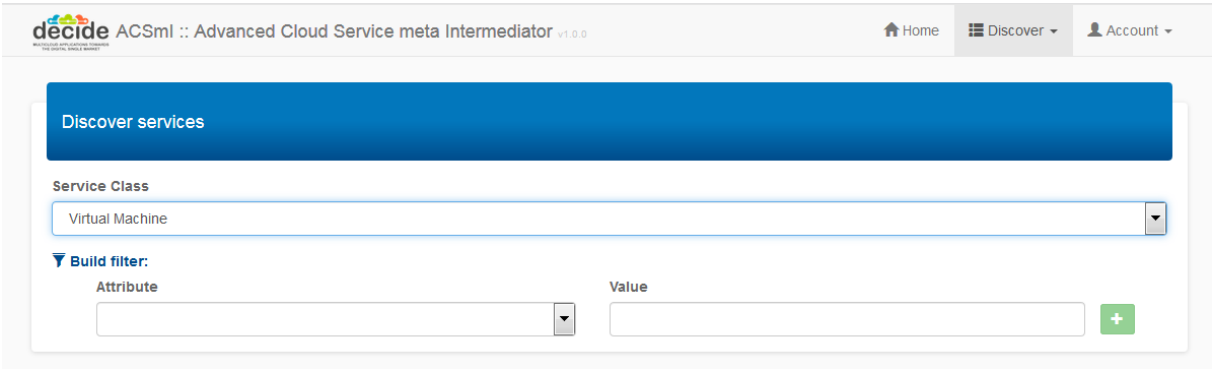


Figure 26. Discovery functionality

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

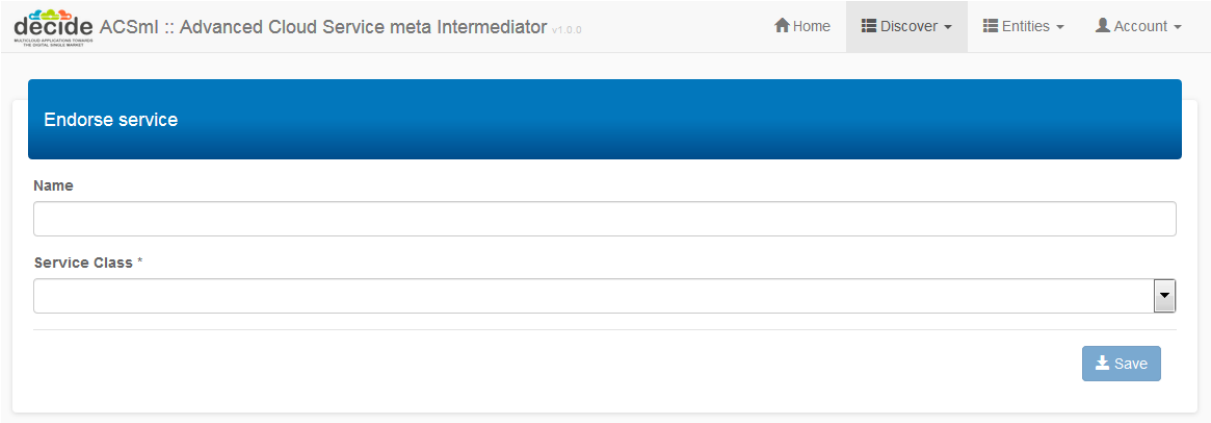
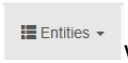


Figure 27. Endorse functionality

- The tab 

when CSP is log-in allow to create, read, update and delete its services.

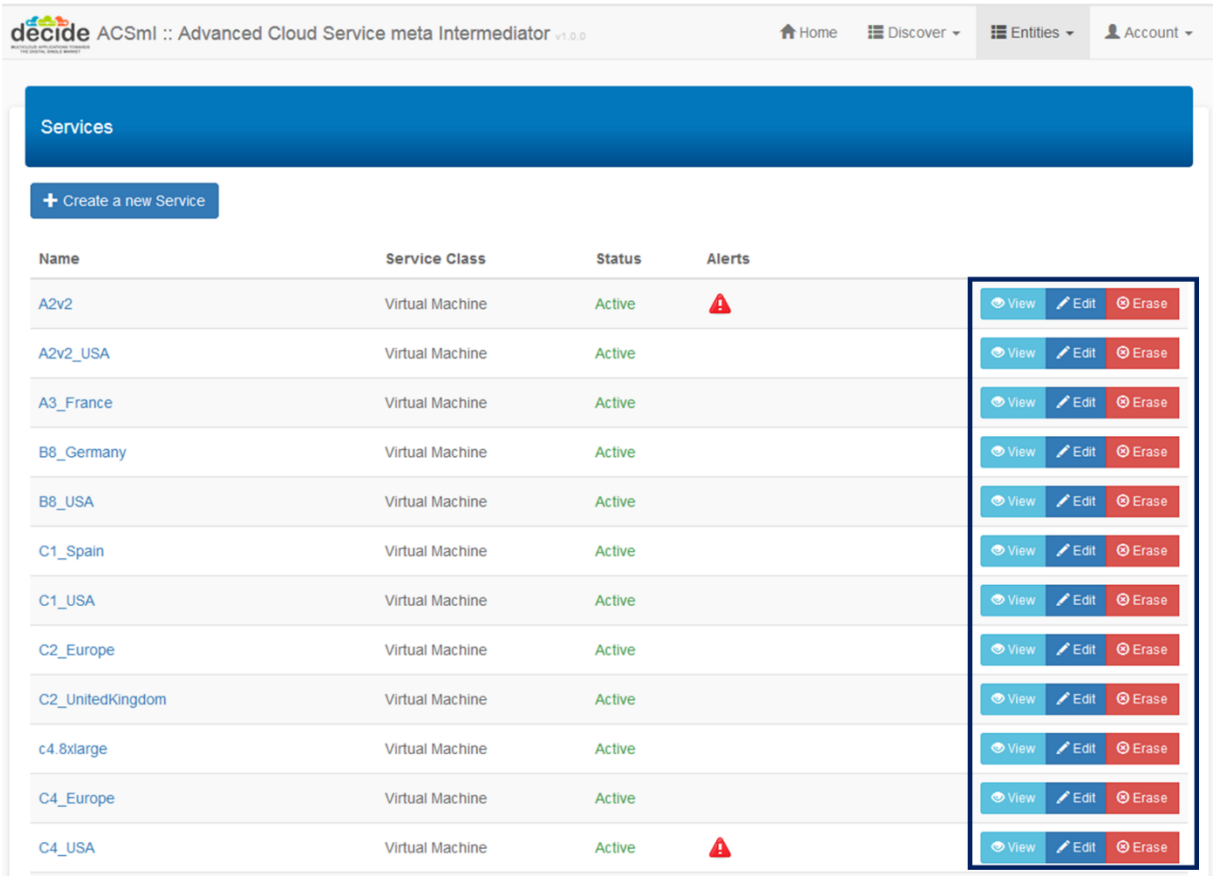


Figure 28. CRUD in the services registry

The information that both CSP and user can see for each service is shown in the figure below.

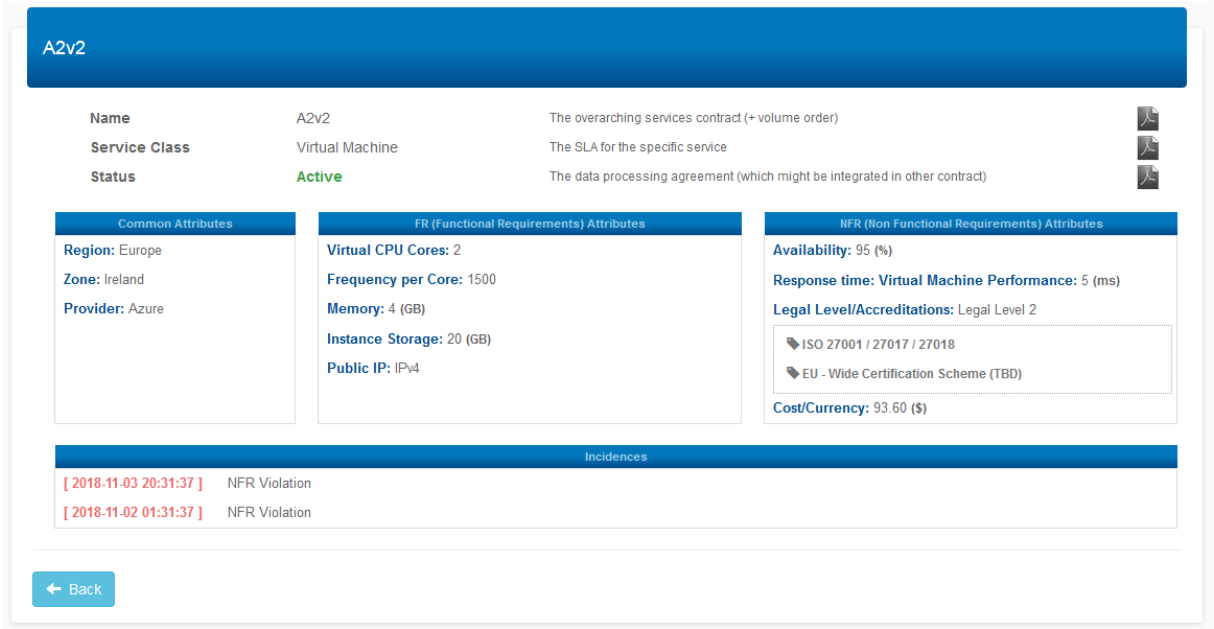


Figure 29. Cloud service information

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):
 *
 * Alberto Molinuevo                    Tecnalia
 * Gorka Benguria                       Tecnalia
 * IÃ±aki 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 30. Copyright header

3.3.5 Download

The source code is available in the EC portal for deliverables, included in the zip file for D5.3.

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

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

4 ACSmI Contract management

4.1 Functional description

ACSmI Contract Management component, described in this section, is responsible for the execution and management of the core functions with respect to the service contracts.

This component is a part of the Business Model Management component of the ACSmI and implements the following functionalities explained in the section above:

F1. **Service Contracting.** The prototype allows to establish contracts with cloud providers to use their resources through the DECIDE framework.
F2. **Manage CSPs**. Access Management to deploy the software onto an infrastructure user needs to have an access. The prototype offers the possibility to the user to provide their own credentials or to establish a new contract. Once established, the existing contract can be reused.

Requirements covered by the prototype:

Information on the requirements covered by the M24 prototype is presented in the table 4.

Table 4. Requirements covered by the M24 ACSmI Contracting 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.	The requirement is covered by the prototype.
WP5-BUS08	This requirement shall allow developer to contract a service or services directly with the CSP.	By M24 the contracting is performed through the CloudBroker Platform. Direct contracts with the CSPs are to be implemented.
WP5-BUS09	This requirement shall generate the APIs required to contract the services and monitor them in different CSPs.	The prototype allows to contract the services. Still, a small amount of API calls is to be extended (e.g. API call to terminate the contract is to be added.)

The current prototype of the ACSmI Contracting tool supports the situations when contracting is requested for more than one resource. General flows for both cases are presented in the Figure 31 and the Figure 32 .

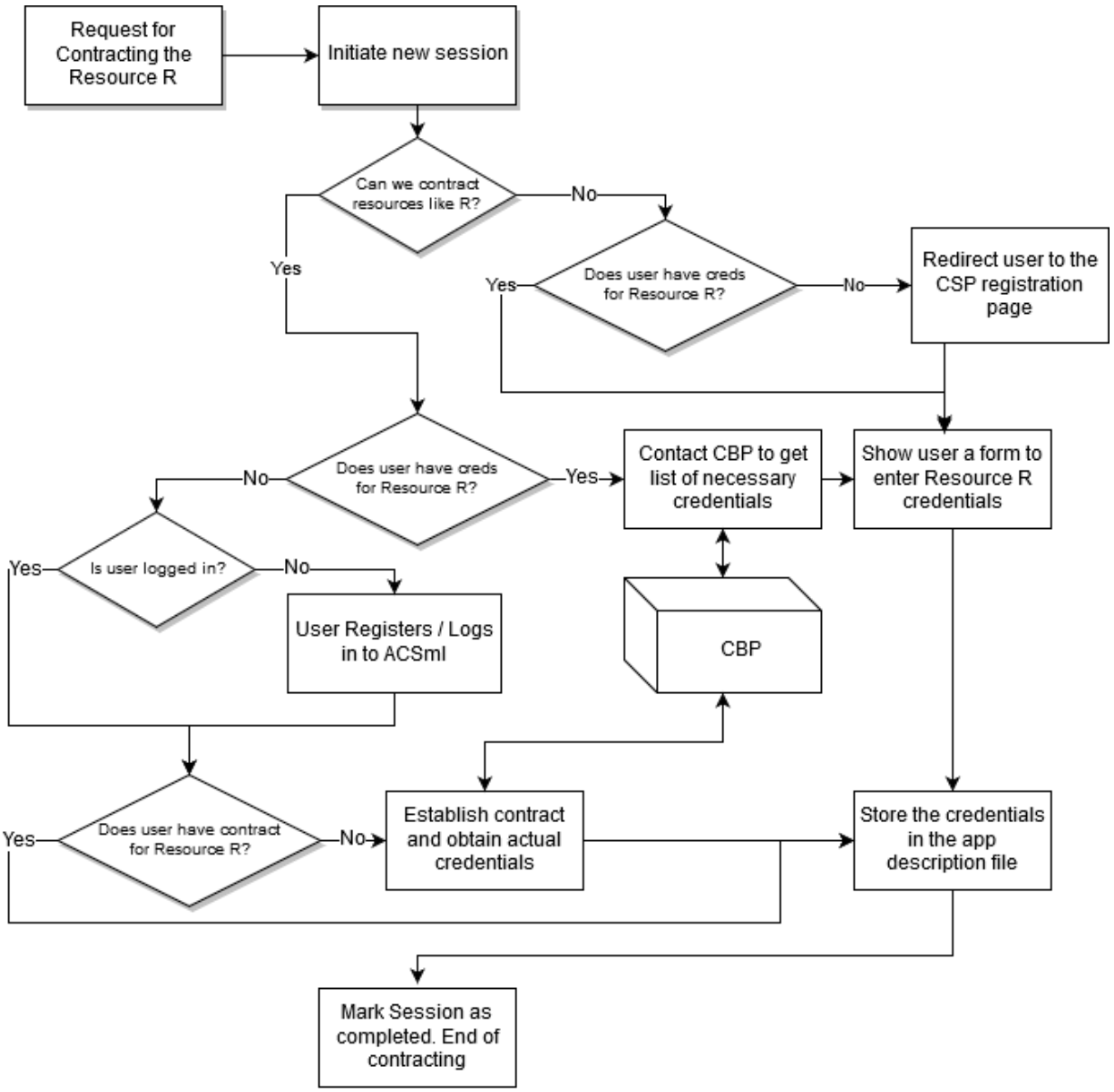


Figure 31. Single contracting Flow

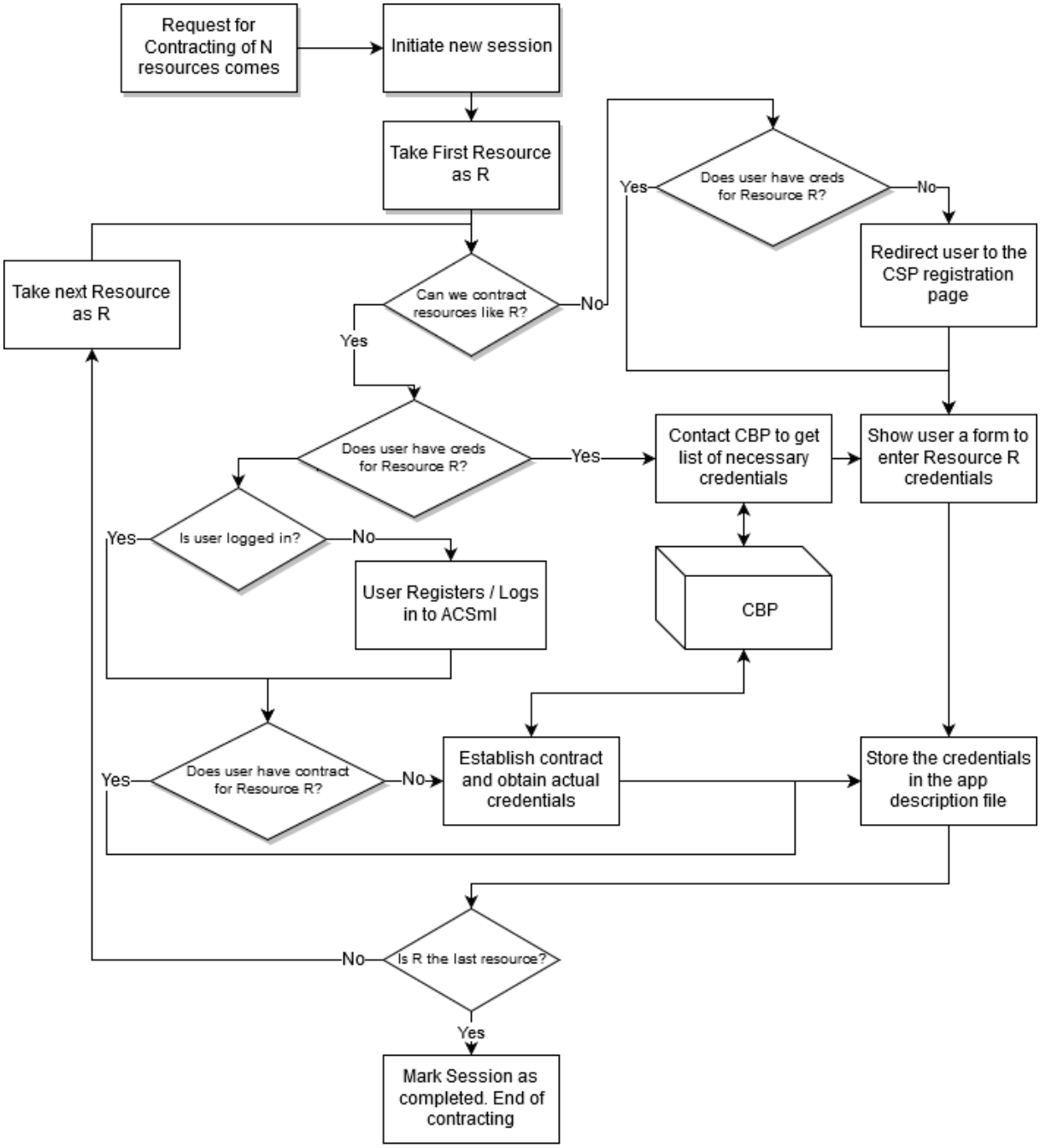


Figure 32. Multi-contracting Flow

4.2 Technical description

4.2.1 Prototype architecture

The architecture of the ACSmI Contracting prototype is presented in the Figure 33.

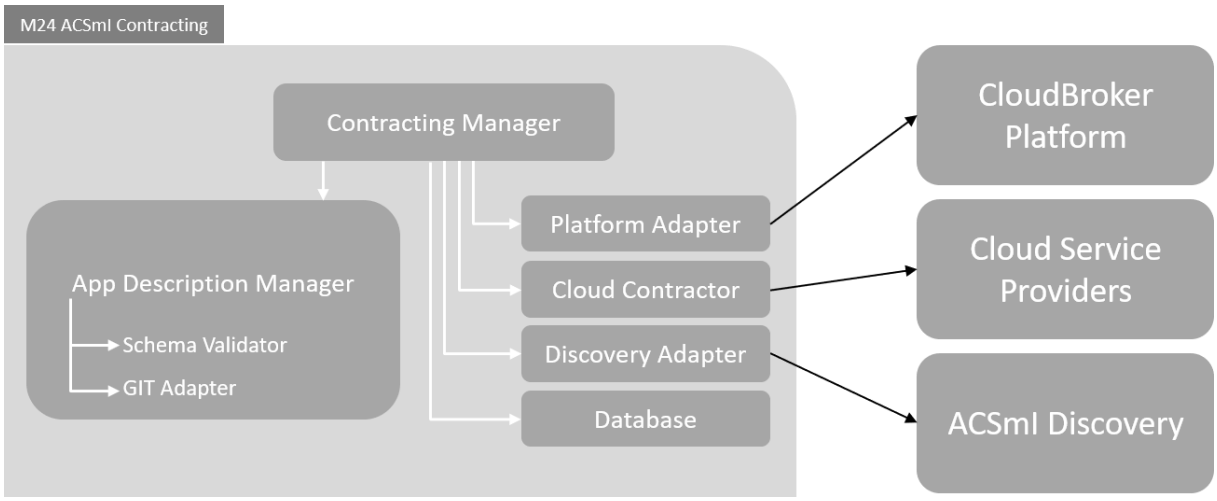


Figure 33. ACSmI Contracting architecture

The purpose of the different components of the figure above is:

- *Contracting Manager* is responsible for the contracting depending on the presence or absence of user's cloud resource credentials.
- *App Description Manager* interacts with the Application Description file - a file in json format containing all the information about the DECIDE Application.
- *GIT Adapter* is responsible for receiving and initiating API calls related to the Application Description file.
- *Schema validator* is responsible for the validation of the Application Description file.
- *Platform Adapter* serves for the communication with the CloudBroker Platform to define the configurations of VMs.
- *Cloud Contractor* allows developers to contract their services directly with the CSPs. The subcomponent is in development by M24.
- *Discovery Adapter* serves for the communication with the ACSmI Discovery component to define the service requirements.

The prototype is structured in one container with a monolith Ruby on Rails application (*Frontend* and *Backend)* and SQLite3 database inside.

4.2.2 Technical specifications

The prototype design expects the App Description Manager to receive the API call to determine the path and credentials to git repository where Application Description file is located.

The git repository data is expected to be passed through parameters *git_url, git_username, git_password*. All three parameters are mandatory. There should be no other parameters expected. The names of the parameters are not yet finalized, thus may get changed, however the principle should remain as described. The ACSmI Contracting Component is requested to work with the application description *json* file.

It is expected, that the Application Description file is located in the root folder of the repository and is called DECIDE.json. Name and location of the file can get changed.

After the pull, the Application Description file is validated against schema file by Schema Validator. Fields "csId" (ID of the service on ASCmI Discovery) and "index" (int index for mapping to the VM) are mandatory for each service present in description file.

Platform Adapter sends a request to CloudBroker Platform to get the VMs' configurations to fit the service requirements (if any) requested by Discovery Adapter from ACSmI Discovery.

If successful, the URL of the contracting session is returned to be opened within the Framework. Following the design, the UI of the ACSmI Contracting component is not shown as a separate page but is embedded to the DevOps Framework UI using iframe. It is requested to write all the results of the contracting to the Application Description file. This should include the URL to the platform, user's credentials and the following parameters for each contracted service: csid, software_id, resource_id, region_id, instance_type_id, key_pair_id, and other if necessary. As soon as the parameters are written to Application Description file, the file should be automatically pushed back to the git repository.

4.3 Delivery and usage

4.3.1 Package information

The structure of the package for the ACSmI Contracting component is presented in **¡Error! No se encuentra el origen de la referencia.**.

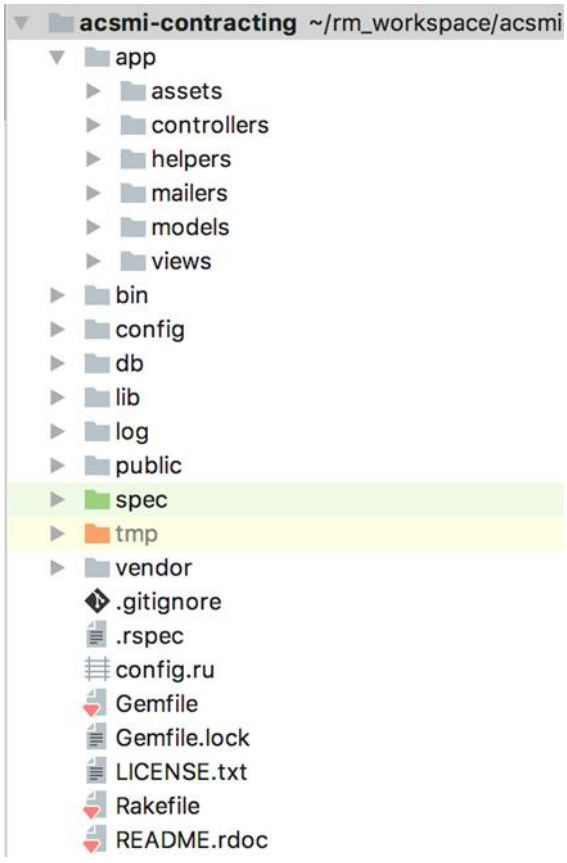


Figure 34. ACSmI Contracting Package Structure

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

The below steps should be followed to install this component:

1. Install ruby version 2.3.X (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 Resource Contracting

In order to communicate with the ACSmI Contracting component an API call to /decide/acsmi/contracting/api/v1/sessions with application description file storage repository data should be made.

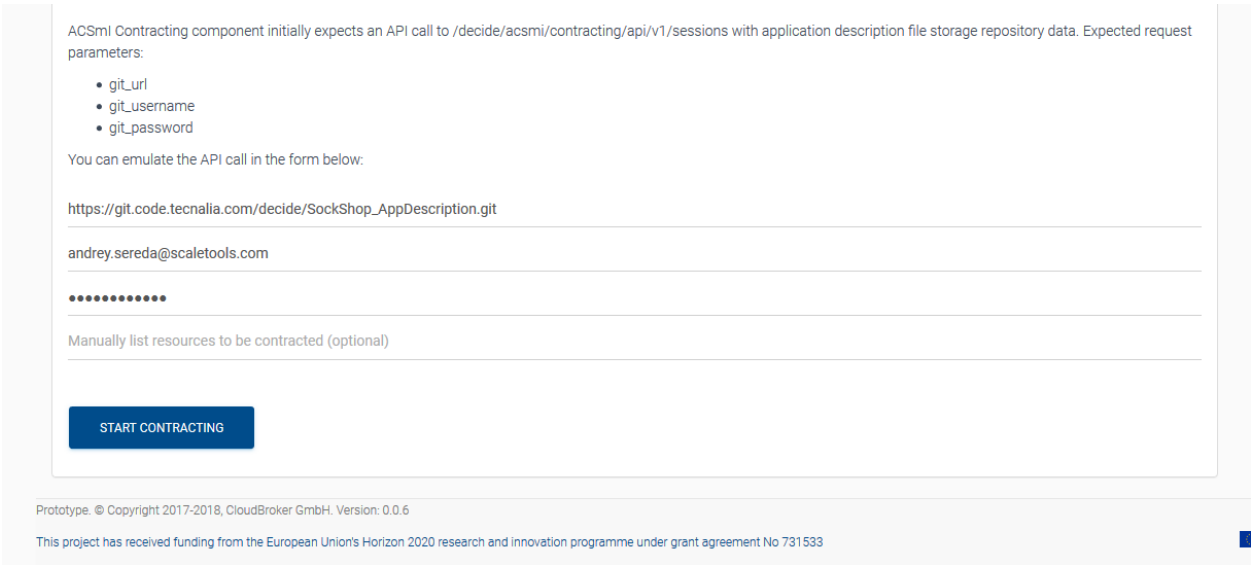


Figure 35. Expected request parameters

By click on "Start Contracting" button, 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.

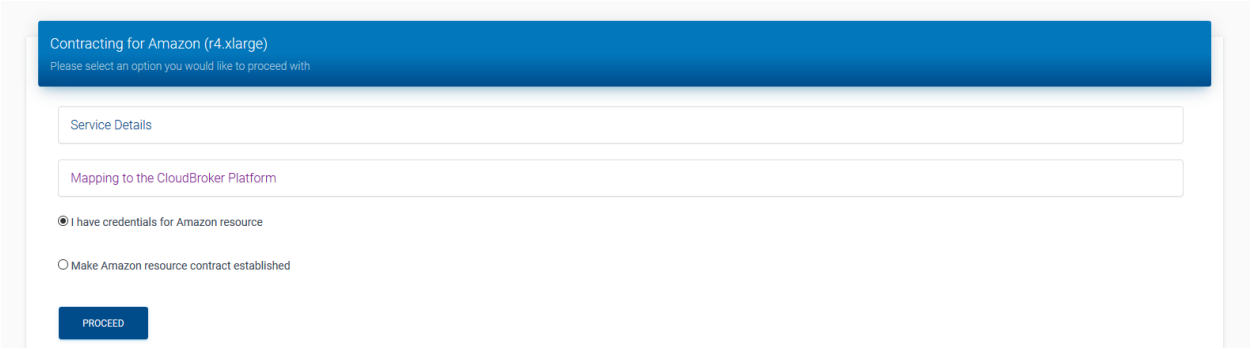


Figure 36. Entry point for contracting (Scenario 1)

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

Scenario 1: User selects the “I have credentials …”.

Step 1.1: In this scenario, a form is displayed to the user where he/she has to enter already existing resource credentials. User can also go back to the previous page by clicking on the “Back” button.

Figure 37. Form to enter existing resource credentials

Step 1.2: The credentials are then stored within the ACSmI. The contracting has been successfully performed, the following page shall be displayed. The first scenario is then successfully completed.

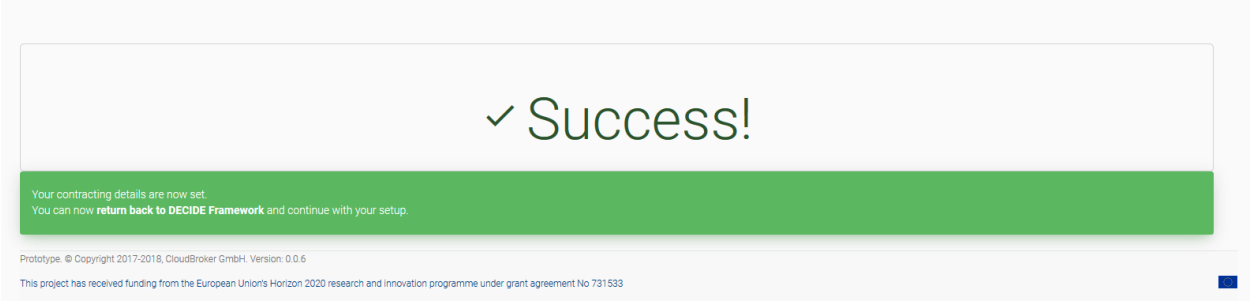


Figure 38. Successful contracting

However, the user can initiate the second scenario if the option “Make … resource contract established”. is selected on the entry point screen below:

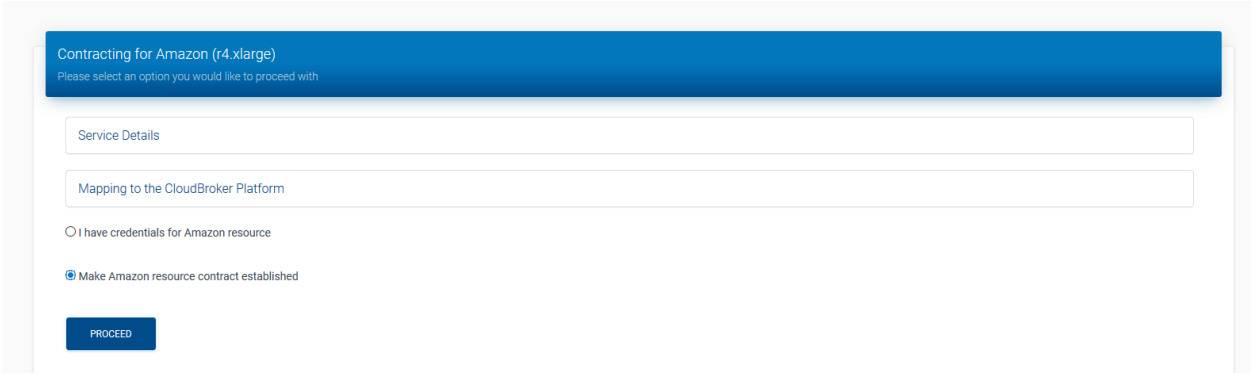


Figure 39. Entry point for contracting (Scenario 2)

Scenario 2: User selects the "Make … resource contract established".

Following the scenario 2, the first stage of this scenario is ACSmI to User contracting. To do this, user has to be signed into ACSmI. In case the 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.

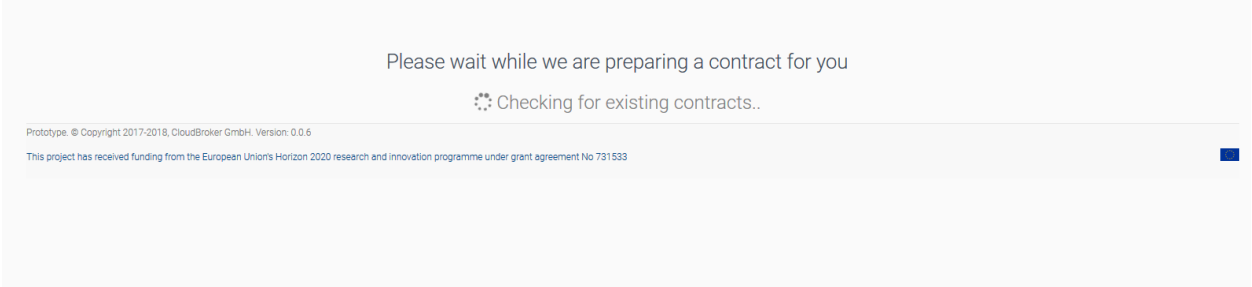


Figure 40. Contracting processing screen

Step 2.1.1: In case the 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".

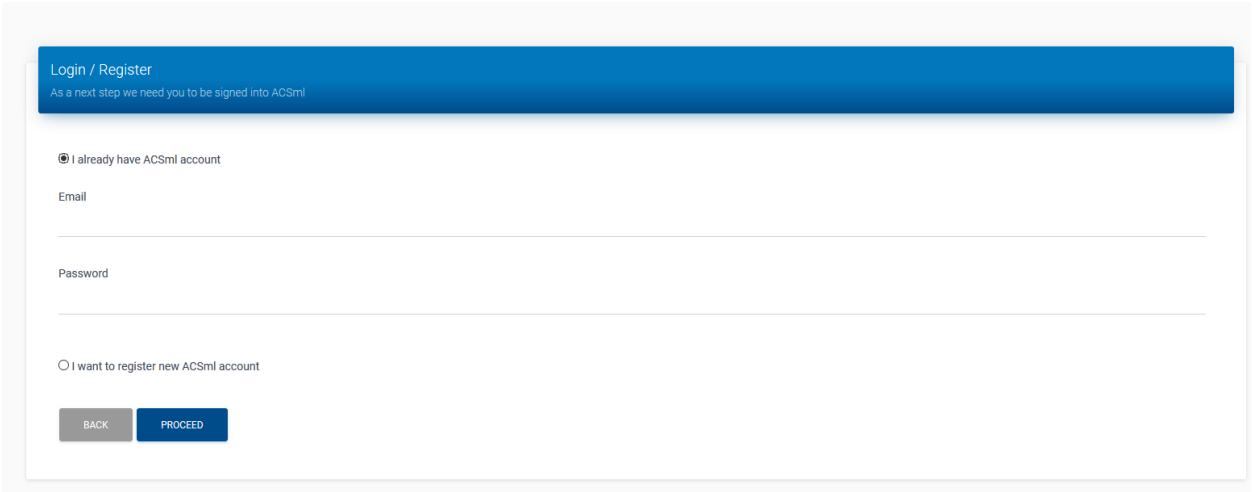


Figure 41. Form to provide existing ACSmI credentials

Step 2.1.2: In case the 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.

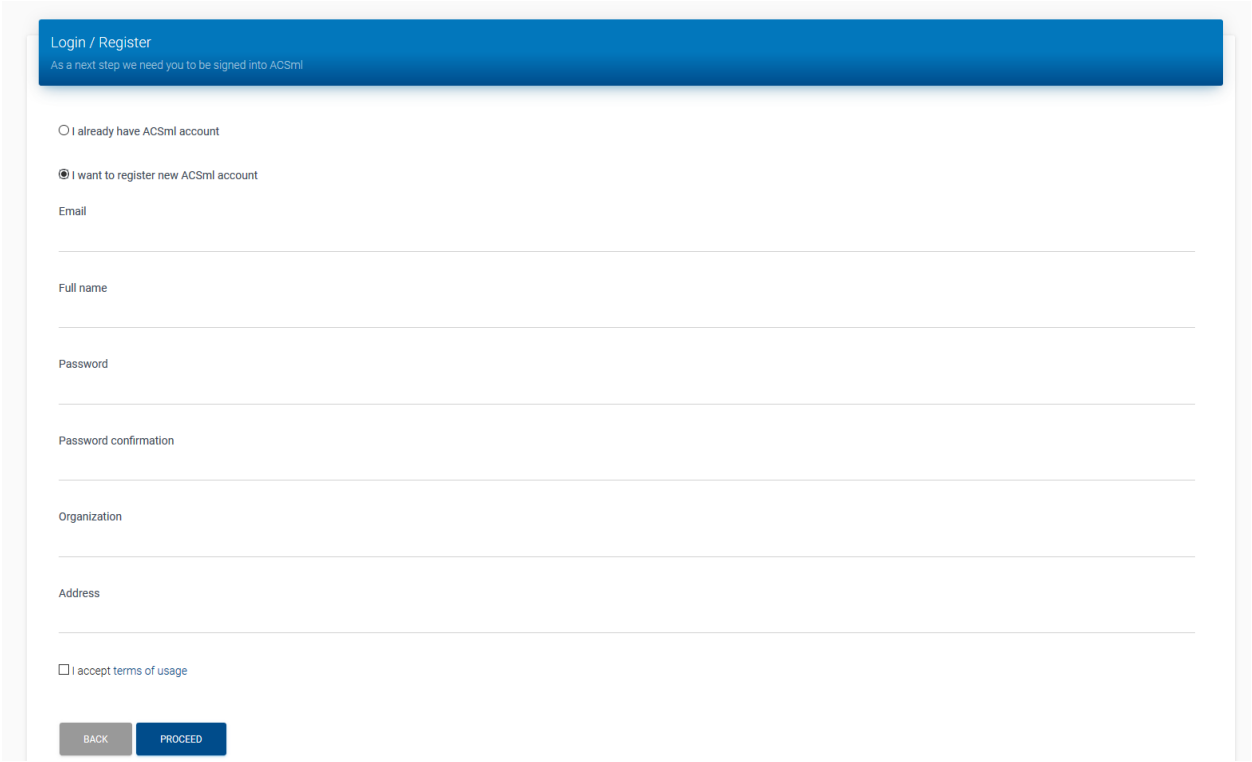


Figure 42. Form to register a new ACSmI account

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:

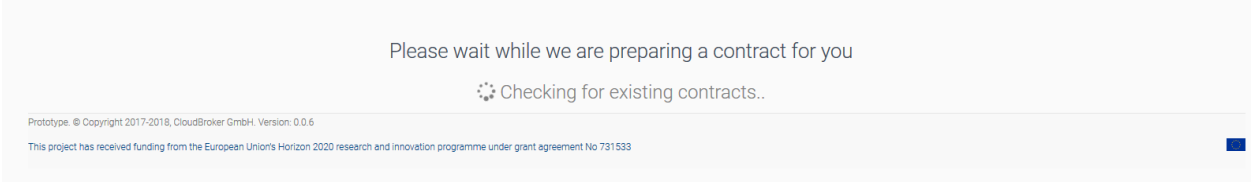


Figure 43. Contracting processing screen

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

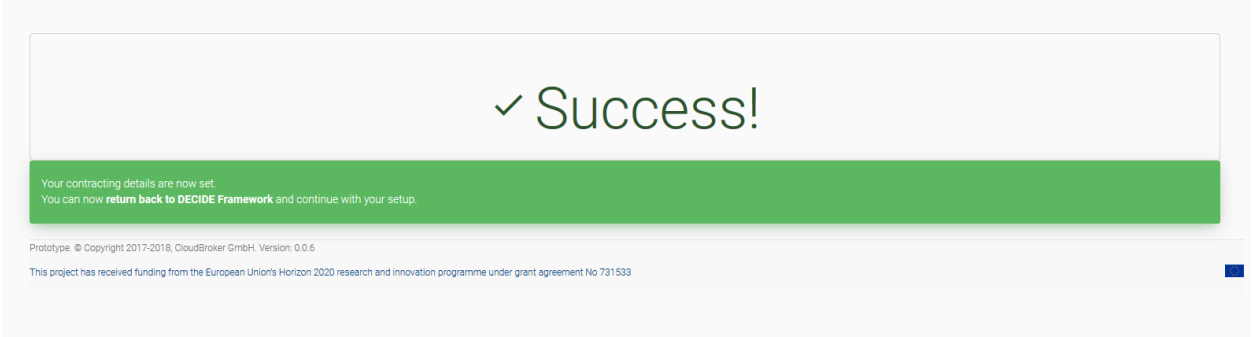


Figure 44. Successful contracting

4.3.3.2 Contracts Management

Once logged in, user is able to view and manage contracts attached to his/her account.

Figure 45. View of the contracts for an account

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. By clicking on "Delete" button user can delete the given resource contract. The "Back" button shall lead back to "Contracts" page.

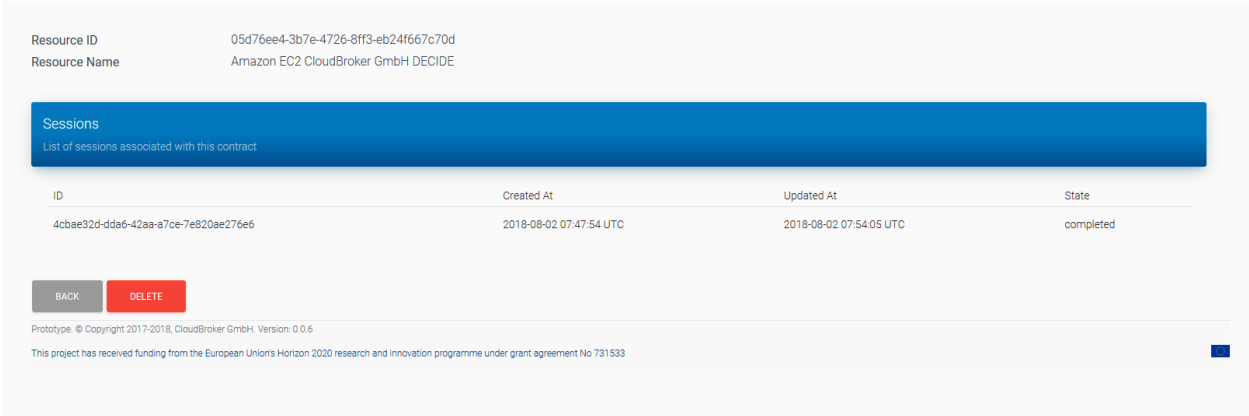


Figure 46. Screen for deleting contracts.

On the "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".

4.3.4 Licensing information

The component is offered under "Apache 2.0" license.

4.3.5 Download

The source code is available in the EC portal for deliverables, included in the zip file for D5.3.

This release is available in the DECIDE open git repository (https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git), more precisely at the following address:

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

5 ACSmI Monitoring

5.1 Functional description

ACSmI Monitoring component described in this section monitors the fulfilment of the SLAs for each Cloud services contracted.

This component implements the following functionalities explained in the section above:

F3. **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.

The following table details the requirements covered in this prototype based on the ACSmI requirements listed in the D2.2 [1].

Requirements covered by the prototype:

Table 5. Requirements covered by the M24 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 M12, the NFRs taken into account to derive the associated parameters was the Availability of the Cloud service. For this prototype (M24), the associated parameters that have been defined are performance and location
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 (MON03)	For M24, it has been agreed that ACSmI alerts ADAPT VH about the violations. ADAPT VH is responsible to take care of all the activities that this violation requires. The parameters associated are: availability, performance and location.
WP5-MON06	If a SLA parameter is violated, means to alert the operator (ADAPT VH) about which parameters have been violated in order to create a new deployment configuration shall be put in place. This shall ensure a more reliable service.	This prototype uses the interface provided by ADAPT VH to send it the information regarding the cloud services that have violated its SLAs.
WP5-MON08	A SLA Assessment has to take place as it provides insight on whether the CSPs will fulfil the SLA in its entirety or whether it needs to be re-evaluated, amended or undergo through changes.	This subcomponent takes the information from the InfluxDB data base and assesses it to check if the metrics obtained fulfil or not the SLA of the services. This subcomponent uses the MCSLA core library to support this assessment.
WP5-MON09	All violations shall be logged and the log shall be obtainable by the users. The log	Once a service cloud SLA violation is detected, ACSmI monitoring uses the interface provided by ACSmI discovery to

Req. ID	Req. Description	Requirement coverage by the prototype
	shall hold the following parameters and values: • CSP Id/info • Violated parameters • Value of violated parameters • Time and date of parameters The log should be read only, hashed and signed by ACSmI	record in the ACSmI registry the information regarding this violation.
WP5-MON10	"Push Monitoring" means that no facilities or agents required. As it does not need additional software installation, the monitoring activities will not impact the performance.	This prototype implements this type of push monitoring using the Telegraf plugin. For VM the plugin to be used is "ping".

As mentioned in the table above in this intermediate prototype, the NFRs to be assessed are performance, availability and location for virtual machines because VMs are the only cloud service that are used on the use cases at this moment.

In order to assess the fulfilment of the these two NFRs, ACSmI monitoring will take the parameters associated (WP5-MON03).

- Availability. This NFR has defined as A = MTBF/(MTBF+MTTR). For this metric, two parameters have been defined:
  - MTBF: Mean time before failures
  - MTTR: Mean time to recover.
- Performance. This NFR has been defined as response time. This parameter is defined as the time required to respond to a request.
- Location. This NFR determines where a virtual machine is located, reading its IP address from Service registry. The information that relates the IP addresses with their locations is taken from MaxMind GeoIP2 Java API [22] with the free GeoLite2 database.

This M24 prototype interacts with different components of the DECIDE tool suite (see Figure 47), namely:

- **ADAPT.** ACSmI monitoring interacts with two components of ADAPT: 1) ADAPT monitoring, that will indicate ACSmI monitoring that a new application is deployed, and new cloud services should be monitored. 2) ADAPT Violation Handler that will be informed by ACSmI Monitoring when a SLA violation is detected in order to take care of the required actions.
- **MCSLA Core library.** ACSmI monitoring uses the library provided by MCSLA to calculate if a NFR (Availability and performance) does not accomplish the SLO.
- **ACSmI registry** to inform that a cloud service has violated the SLO and to record this violation in the ACSmI registry.

5.2 Technical description

5.2.1 Prototype architecture

For the implementation of this component of ACSmI, a similar approach to the one followed in ADAPT monitoring [16] will be used.

Figure 47 represents the architecture for this component.

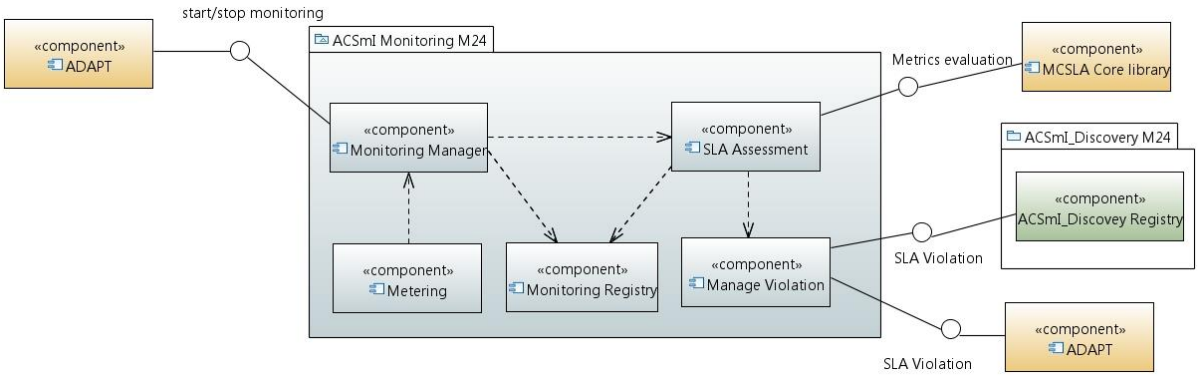


Figure 47. General architecture of ACSmI monitoring.

In the subsequent section these components are explained.

5.2.2 Components description

The current prototype includes the 5 components envisioned for ACSmI monitoring.

- **Metering:** This sub-component collects the data from the different cloud services where the application is deployed. In the current prototype metering sub-component has been configured in an automatic way based on the information registered in the application description. This subcomponent gets the raw metrics (explained in section 5.1) and stores them in the monitoring registry.
- **Monitoring registry:** This sub-component is in charge of storing the data collected from the metering sub-component
- **Monitoring manager:** This component manages all the activities of ACSmI monitoring. It receives the request to start the monitoring of the cloud services from. Then this sub-component configures the different agents in the metering sub-component and sets up the monitoring registry.
- **SLA Assessment**, this sub-component is in charge of the aggregation of the different raw metrics in order to assess the values of the different NFRs with respect to the SLOs.
- **Manage Violations**: Once the SLA Assessment detects a violation of the SLA CSP, this subcomponent is in charge of carrying out the corresponding activities: 1) inform the violation handler of ADAPT, 2) registry the violation in the ACSmI registry and 3) inform the CSP of the occurred violation.

5.2.3 Technical specifications

ACSmI monitoring prototype for M24 has been implemented as a monitoring stack, covering the data collection, the data storage, data aggregation and Cloud service SLA assessment.

- **Data collection** (Metering subcomponent): For the data metering, Telegraf [17] open source technology is used. Telegraf is a plugin-driven server agent written in Go for collecting, processing, aggregating, and reporting metrics. It is a compiled and standalone binary that can be executed on any system with no need for external dependencies. For the purpose of this M24 prototype of ACSmI monitoring the two Telegraf plugin (one output and one input) have been configured to acquire the parameters required and indicate where to record them:
    - *Ping* plug-in is used to get the raw metrics that allow to calculate the parameters defined for calculate availability and performance. The raw metric to calculate availability is taken from the field “result_code (int, success = 0, no such host = 1, ping

error = 2)". With this metric and the time stamp recorded in the influx DB, ACSmI monitoring can calculate the availability and assess if the SLA is fulfilled or not. The raw metric to assess performance is "maximum_response_ms (integer)". This metric provides the information to assess the compliance or not of the performance NFR.

- Output plugins: *Influx DB plugin*. This plugin sends the metrics gathered by the agent to a specific Influx DB instance (with a given url). The name of the database, login parameters, etc. need to be specified.

Beyond telegraf, ACSmI monitoring also uses "*MaxMind GeoIP2*" Java API with the free GeoLite2 database to gather information regarding the location of the VMs.

- **Data storage** (Monitoring registry sub-componet): For the storage of the metrics, InfluxDB storage technology has been implemented. InfluxDB is used as a data store for cases involving large amounts of timestamped. Influx DB supports plugins for data collection protocols like Telegraf. The measurements are injected by the Telegraf plugins in the Influx DB database.
- **Monitoring manager**: It is a Java program that manages the different activities and workflow to be able to start and configure all the elements for the monitoring. It is deployed using a Jetty server.
- **Cloud Service SLA Assessment**. It is a java function that compares the SLO introduced by the CSP when endorsing a service (ACSmI Discovery). The current prototype uses the MCSLA library to support this assessment. This function is included as part of the monitoring manager.
- **Manage violation**: this sub-component has been developed as a function of the Monitoring manager. This function alerts to the VH and the ACSmI registry that a violation is occurred using the interfaces provided for these two different components.

5.3 Delivery and usage

5.3.1 Package information

Next figure shows the structure of the package for the ACSmI monitoring component

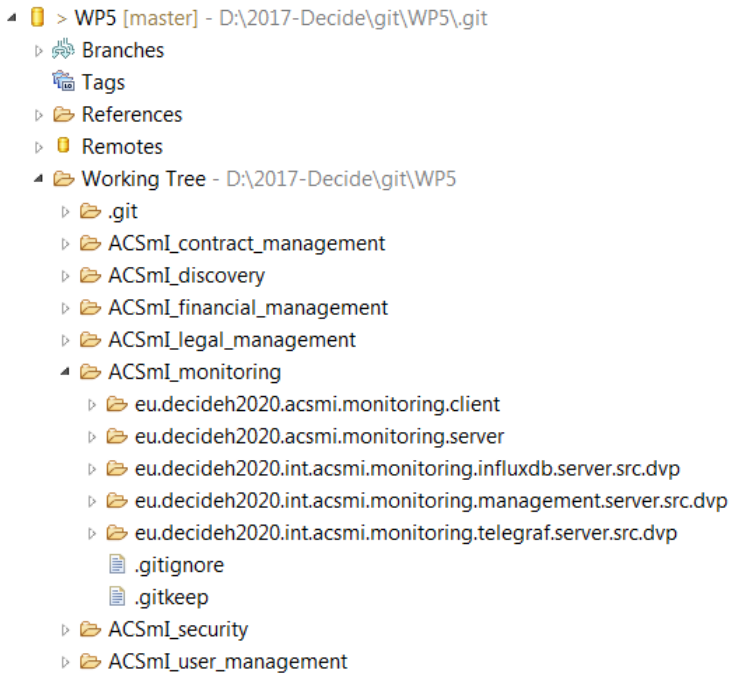


Figure 48. ACSmI Monitoring Packages information.

ACSmI monitoring is composed of five main packages, described below:

- *eu.decideh2020.acsmi.monitoring.client*. It contains the source code of the generated client of the acsmi.monitoring server that it will be used by other tools to communicate to ACSmI monitoring.
- *eu.decideh2020.acsmi.monitoring.server*. It contains the source code of the component "monitoring manager. It contains the logic of the ACSmi Monitoring. The figure below shows the most relevant structure of this package.

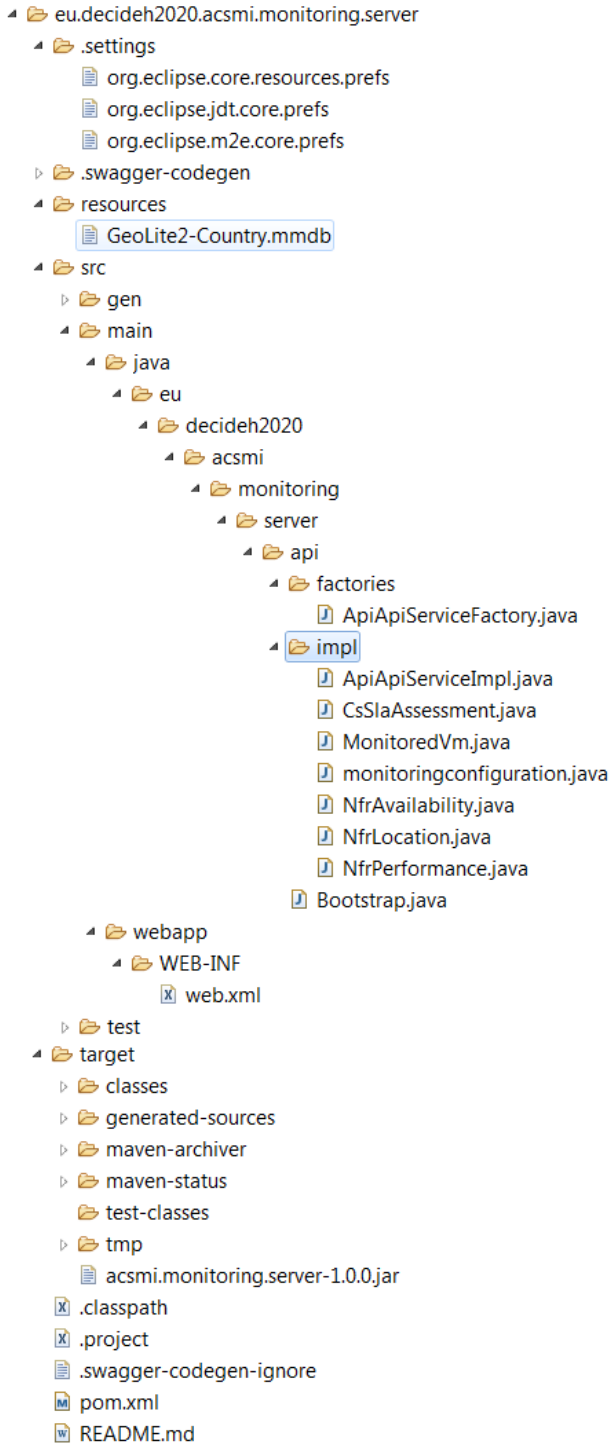


Figure 49. eu.decideh2020.acsmi.monitoring.server package.

The most relevant files are:

- ApiApiServiceImpl.java. This file records the call from ADAPT monitoring and launches all the activities to be done
- Monitoringconfiguration.java is the file responsible to configure Telegraf and to launch the assessment process
- CSSLAAssessment.java. The objective of this file is received the notifications of possible violations and alert to ACSmI Registry.
- NFRAVailability, NFRPerformance and NFRLocation are the responsible to detect if a violation has occurred in each NFR.

- *eu.decideh2020.int.acsmi.monitoring.influxdb.server.src.dvp*. It contains the source code required to record the metrics in InfluxDB. Also, it contains the Docker file to generate the container.

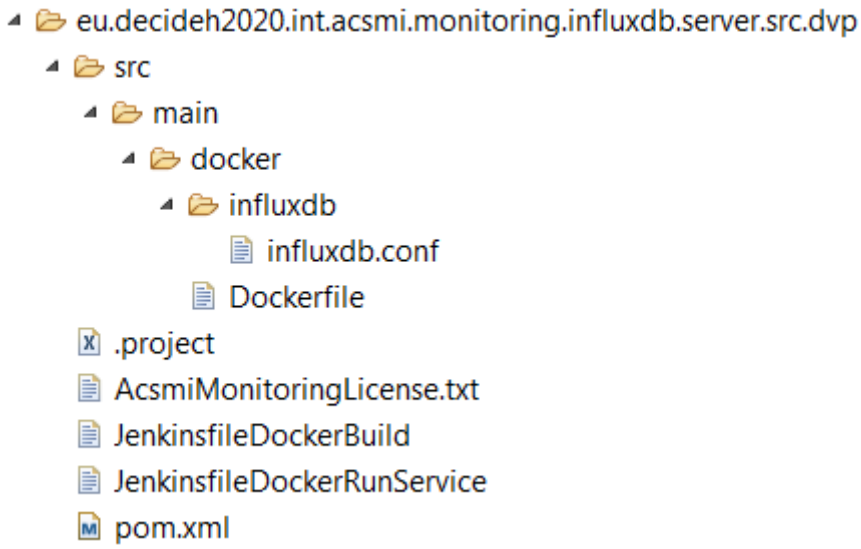


Figure 50. eu.decideh2020.acsmi.monitoring.influxdb.server.src.dvp package.

- *eu.decideh2020.int.acsmi.monitoring.management.server.src.dvp*. It contains the corresponding docker file to generate the container for deploying ACSmI monitoring component.

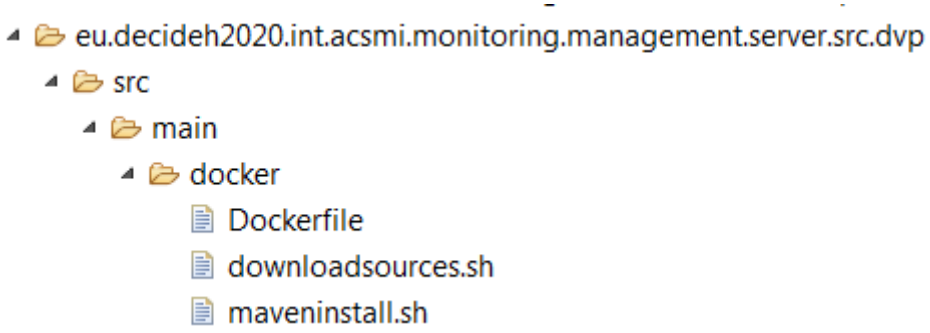


Figure 51. eu.decideh2020.acsmi.monitoring.management.server.src.dvp package.

- *eu.decideh2020.int.adapt.monitoring.telegraf.server.src.dvp:* It contains the source code required to allow telegraf getting the metrics. *Telegraf.config* file contains the inputs and outputs plug-ins required to measure the cloud services. Also, it contains the docker file to generate the container.

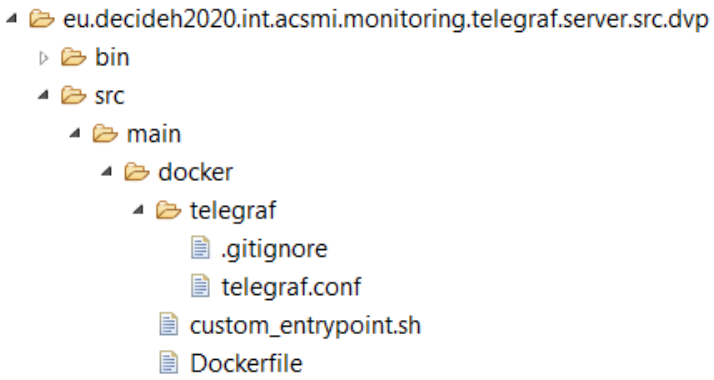


Figure 52. eu.decideh2020.acsmi.monitoring.telegraf.server.src.dvp package.

Installation instructions

The prototype has been installed and tested using the following software (see Pre-requirements section):

- Docker to create the containers of the different sub-components.

The following steps need to be executed to get the prototype up and running:

1. Download the src from the DECIDE repository.

```
Git clone
https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git
```

2. Insert the credentials and the git path for the git repository to be used to store the Application description[1] in the *ApiApiServiceImpl.java* file.

```
String localAppDesc = "";
localAppDesc = System.getProperty("java.io.tmpdir");
localAppDesc = localAppDesc+"\\"+System.currentTimeMillis();
Path localappdescpath=Paths.get(localAppDesc);
// Repository where the app description is... When integrated it will be resourcemonitoring.appdescuri
//Local DECIDE.json
String gitRef="https://git.code.tecnalia.com/j2018b/106779.git";
// Introduce the user and password of the repository where the DECIDE.json is
//String gitusername="";
//String gitpassword="";
//open appManager
```

Figure 53. Excerpt of the ApiApiServiceImpl.java file where the credentials for the git where the application description is stored are set up.

3. Each of the containers of the different sub-components needs to be built

[1] The application description JSON file (DECIDE.json) used as input for this intermediate ACSmI monitoring is included in the source code inside the eu.decideh2020.acsmi.monitoring.server folder

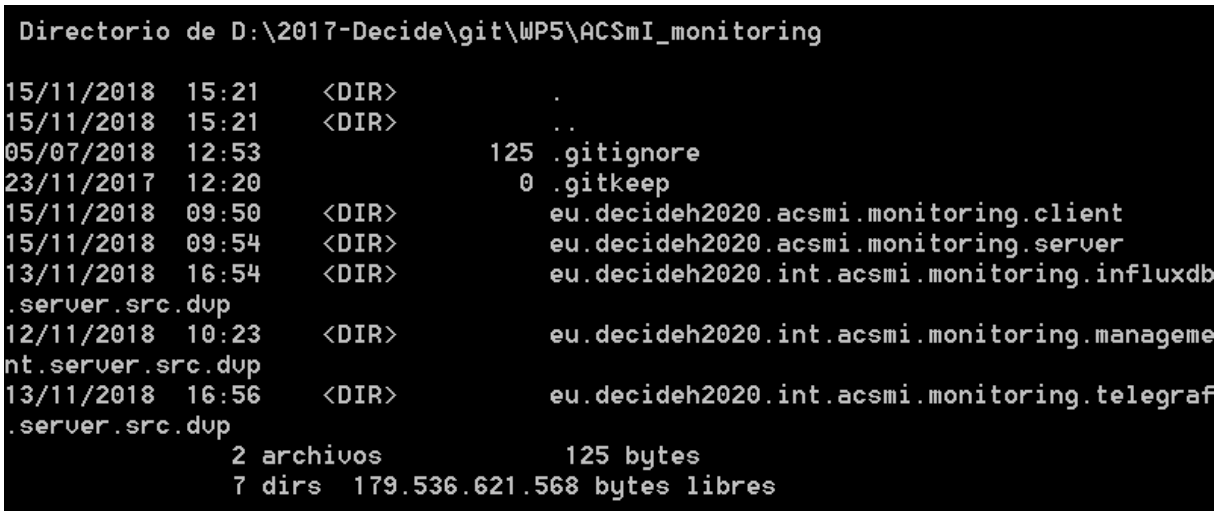


Figure 54. ACSmI monitoring folder structure

a. Telegraf container:

i. Go to the folder where the docker file for telegraf is located.

```
Cd ..\
ACSmI_monitoring\eu.decideh2020.int.acsmi.monitoring.telegraf.se
rver.src.dvp
```

ii. Build the docker image for telegraf with the following arguments

```
docker build – f docker.telegraf.server –t
tecnalia/eu.decideh2020.acsmi.monitoring.telegraf.server
```

iii. Run the docker image with the following arguments (Telegraf container):

```
docker run
tecnalia/eu.decideh2020.adapt.monitoring.telegraf.server -d --
restart=always --name
eu.decideh2020.adapt.monitoring.telegraf.server --add-host
influxdb:172.18.0.1 --add-host sockshop:172.26.252.81
```

In case of the telegraf container, the influxdb Database where the agents are storing the metrics needs to be specified (IP of its docker container) as well as the IP address of the cloud service to be monitored (its docker container IP address). In this case the current release incorporates a testing application (Sock Shop) for testing purposes[2].

b. InfluxDB Container.

i. Go to the folder where the docker file for InfluxDB is located.

```
Cd ..\
ACSmI_monitoring\eu.decideh2020.int.acsmi.monitoring.influxdb.se
rver.src.dvp\src\main\docker
```

ii. Build the docker image for influxDB with the following arguments

[2] To change the application to be monitored this IP address should be accordingly changed.

```
docker build -f /docker.influxdb.server -t
tecnalia/eu.decideh2020.acsmi.monitoring.influxdb.server
```

iii. Run the docker image with the following arguments:

```
docker run
tecnalia/eu.decideh2020.acsmi.monitoring.influxdb.server -d -p
14086:8086 --restart=always -e INFLUXDB_DB='decideh2020acsmi' -
-name eu.decideh2020.acsmi.monitoring.influxdb.server
```

In the case of the influxdb container the mapping of the ports needs to be configured as well as the name of the influx DB. In this case, ACSmI monitoring uses the port 14086 to avoid conflicts with other tools of DECIDE such us ADAPT monitoring.

c. ACSmI Monitoring server Container

i. Go to the folder where the docker file for ACSmI monitoring server is located.

```
Cd                                                                    ..\
ACSmI_monitoring\eu.decideh2020.int.acsmi.monitoring.management.
server.src.dvp\src\main\docker
```

ii. Build the docker image for ACSmI Monitoring server with the following arguments

```
docker build -f docker.acsmi.monitoring.server -t
tecnalia/eu.decideh2020.acsmi.monitoring.management.server --
build-arg GIT_CREDENTIALS=$CREDENTIALS_GIT
```

For ACSmI monitoring server container, the last version of the implemented code needs to be downloaded from git, and for this the GIT credentials are needed.

iii. Run the docker image with the following arguments:

```
docker run
tecnalia/eu.decideh2020.acsmi.monitoring.management.server -d -
p 14080:8080 --restart=always --name
eu.decideh2020.acsmi.monitoring.management.server
```

As with influxdb container, in ACSmI monitoring server the mapping of the ports needs to be configured. In this case, ACSmI monitoring uses the port 14080 to avoid conflicts with other tools of DECIDE such us ADAPT monitoring.

5.3.1.1 Pre-Requirements

The following lists the minimum requirements to get the component working.

- Machine or virtual machine with the O.S (Ubuntu Xenial 16.04) and the Docker server.
- JRE

5.3.2 User Manual

ACSmI monitoring is launched by ADAPT monitoring once the application to be monitored is deployed. ADAPT monitoring uses the interface provided by ACSmI monitoring. In the figure below it can be seen that this interface defines four main methods, in this prototype only the POST method is implemented.

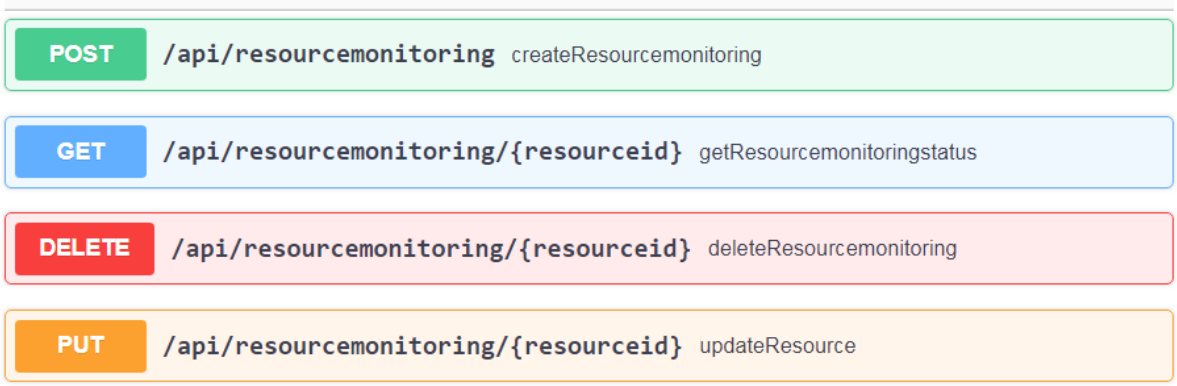


Figure 55. Methods of the ACSmI Monitoring REST API.

In order to test it independently, the starting of the resource monitoring process needs to be launched externally by a call to the ACSmI monitoring REST interface. This can be done using the Swagger editor, loading the corresponding interface json (included in annex 1) and trying the POST method:

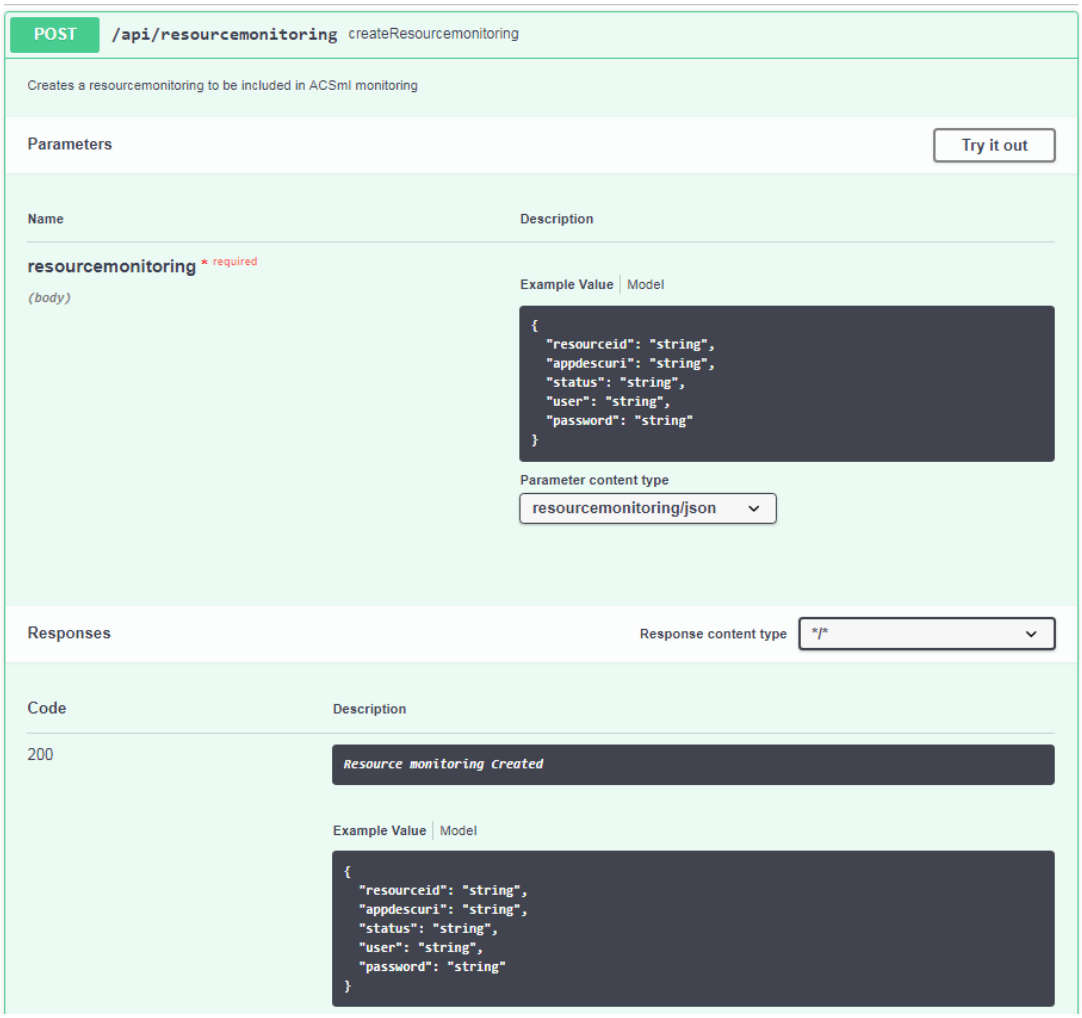


Figure 56. POST method of ACSmI monitoring interface

If the Jetty server is started, this interface can be tested with the "*try out & Execute*" buttons, and an example of the response could be seen in the figure below. This response will be received by ADAPT monitoring to be sure that the monitoring of the resources has been started.

The following steps carried out by ACSmI monitoring are transparent to the user. ACSmI manager configures the configuration file of telegraf based on the information of the application description (Uri of this application description is the parameter passed by ADAPT monitoring in the call). Once telegraf is configured, the monitoring of the different cloud services starts. In this prototype, the NFRs availability, performance and cost are monitored. The application description used in this prototype for testing purposes can be found in the following repository https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components.git.

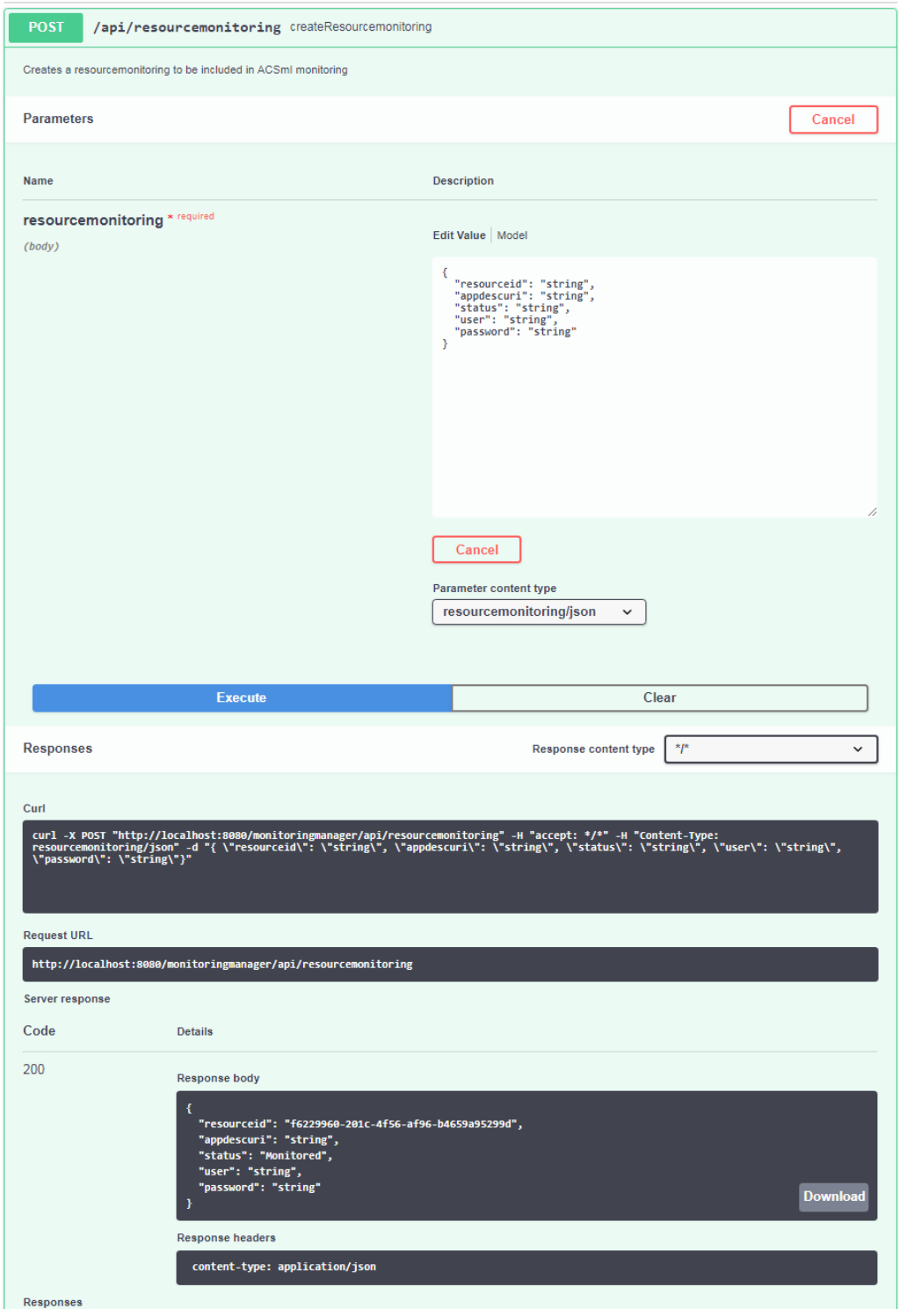


Figure 57. Result of the test call to the ACSmI monitoring interface

5.3.3 Licensing information

This component is offered under MIT license.

5.3.4 Download

The source code is available in the EC portal for deliverables, included in the zip file for D5.3.

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/master/ACSmI/Monitoring

6 ACSmI Billing

6.1 Functional description

ACSmI Billing component, described in this section, is responsible for:

- setting specifi rules for contract(s) billing;
- tracking the usage information;
- charging users in accordance with the defined rules;
- providing billing and usage-related reports.

Requirements covered by the prototype:

Information on the requirements for the component's prototype is presented in table 6:

Table 6. Requirements covered by the M24 ACSmI Billing prototype.

Req. ID	Req. Description	Requirement coverage by the prototype
WP5-BUS03	Each user shall be charged for service usage if there are specific prices for this service. To ensure this, a reasonable billing mechanism shall be available.	This requirement has been postponed to the M30 release.
WP5-BUS04	Since the user is charged on actual service consumption basis, detailed reports related to the resources consumed shall be provided to the user	This requirement has been postponed to the M30 release.
WP5-BUS05	The objective is to enable a regular invoicing.	This requirement has been postponed to the M30 release.
WP5-BUS06	To provide a user with detailed reports related to the resources consumed and costs related to this consumption.	This requirement has been postponed to the M30 release.

The release of the prototype was postponed. By M24 the following functions were designed to be implemented:

- Keeping track of project-related costs (WP5-BUS03)
- Generating billing and usage reports (WP5-BUS04, WP5-BUS06)
- Invoicing user for the contracted resources usage as well as for contract obligations (WP5-BUS05)
- Store charging rules for specific contract

The prototype is expected to interact with the following components of the DECIDE tool suite:

- ACSmI Contracting Component
- ACSmI Billing Component
- ADAPT DO

Expected flow is presented in the Figure 5857 and described below.

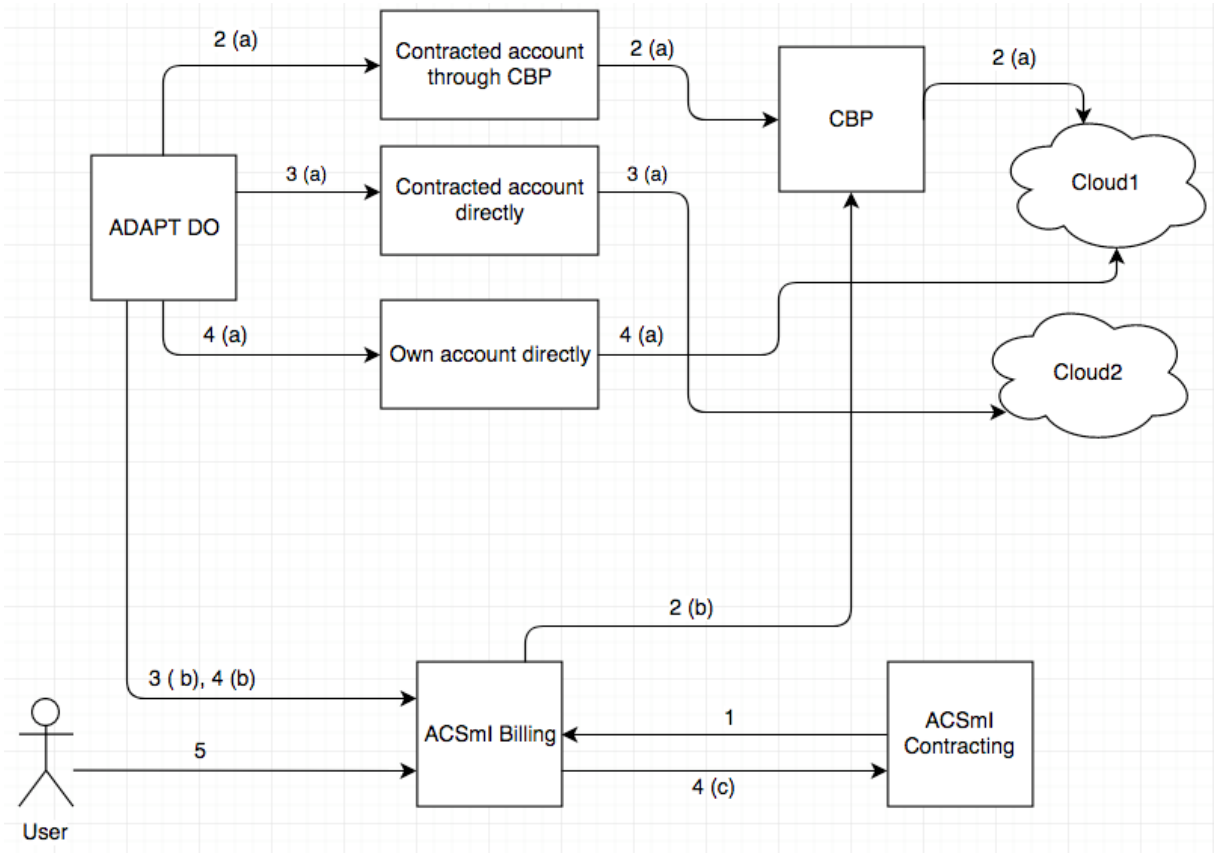


Figure 58. ACSmI Billing Flow

1. ACSmI Contracting Component (ACC) communicates to ACSmI Billing Component (ABC) in the following cases:
    1. User has just established a new contract. Contract details (id + contract price conditions) together with the user data is sent. ACSmI Billing Component initiates the contract's 'entry' charge plus establish periodical charges if specified by the contract;
    2. User has just cancelled the contract. User data and contract id is sent. ACSmI Billing Component finalizes charges for the specified contract, which may include final charge, penalty charges for contract termination and other.
2. For the case when the user has established a new contract and ADAPT DO launches the infrastructure through CloudBroker Platform:
    1. ADAPT DO performs infrastructure launch using CloudBroker Platform and uses the credentials obtained as a result of contracting;
    2. ACSmI Billing Component contacts CloudBroker Platform and requests information on the activity of contracted user and related costs information. The billing is performed accordingly.
3. For the case when the user has established a new contract and ADAPT DO launches the infrastructure directly:
    1. ADAPT DO performs the infrastructure launching directly and uses the credentials obtained as a result of contracting;
    2. ADAPT DO contacts ACSmI Billing Component (ABC) and reports when and how many instances were started and when the instances got stopped
    3. ABC contacts ACC to get the contracting conditions for the specific resource. ACSmI Billing Component performs billing correspondingly.
4. For the case when the user is using own credentials and ADAPT DO launches the infrastructure directly:

1. ADAPT DO performs infrastructure launch directly and uses the credentials provided by user. As the credentials are user-owned, the billing is not to be performed (as cloud provider will bill user directly), however the usage reports are still to be collected;
2. ADAPT DO contacts ACSmI Billing Component and reports when and how many instances were started and when the instances got stopped. This is only used to generate usage reports.

5. User can access the UI of ACSmI Billing Component to get:
    1. List of usage reports
    2. List of billing reports
    3. Corresponding invoices

The prototype architecture is expected to be similar to ACSmI Contracting component: the prototype will be structured in one container with a monolith Ruby on Rails application and SQLite3 database inside.

GUI will be available for the users to obtain billing and usage-based information. Other prototype functions will be performed via API calls.

7 ACSmI Legal

7.1.1 Concept of legally compliant cloud services in ACSmI and relationship to DoA

In the DoA [23], task 5.4 relates to legislation compliance and monitoring of the cloud services that are used in ACSmI. It relates to both ensuring legal compliance when a service is endorsed into the ACSmI registry of services potentially to be used for the deployment of a multi-cloud application, as well as to the monitoring that legal evolutions and changes and taken into account, both when endorsing new services and with regards to existing already endorsed services.

In order to enable such a functionality, it is presupposed that a legal expert will be available to the entity exploiting ACSmI who can take into account these changes and implement them with regards to ACSmI so that the legal compliancy of the endorsed Cloud Services is always correctly assessed and up-to-date.

This is done through assigning to each cloud service a legal level (tier 1, 2 or 3, 1 being the best), based on an extensive questionnaire and guidelines set guiding the legal interpretation of the existing legal compliance situation of a cloud service. The reasoning and logic behind assigning a legal level will be explained in more detail in section 7.1.2. below (the questionnaire and the reason to assign the legal level will be implemented in the next prototype M30). This reasoning will be part of DECIDE's contractual framework, which will consist of:

- A contract with the users, referring to the legal assurance policy detailing the aspects of what the service provides and what it does not.
- An assurance policy, detailing toward the DECIDE users (the application developers) what things are being assessed and monitored, how the objectivity and independence of the legal expert is guaranteed, what the logical behind assigning a legal level is, and what responsibilities rest with the client and/or the CSP. It will detail the onboarding process and how new legislations and developments will be accounted for. It will explain that DECIDE takes no responsibility whatsoever for the information being provided by the CSP to be correct.
- A contract with the CSPs, including a terms and conditions of the use of ACSmI, detailing what the CSPs rights and obligations are, e.g. providing truthful information, undertaking the obligation to notify any change of their contracts, their right to protest the assessment of their service etc. This describes the details of the interaction between the CSP and ACSmI and is aimed at limiting responsibility while creating an amicable relationship with the CSPs.

The assessment of the legal compliance is based on information to be provided by the CSP, namely:

- The service contract applicable to the service
- The SLA applicable to the service
- The Data processing agreement governing the service
- Any other contract also governing the service, if any

These contracts could be uploaded to the ACSmI registry when the CSP endorses its services.

These contracts and any of their updates will be assessed by the legal expert to assign a legal level, based on the logic explained further in section 7.1.2. It suffices to say here that the assessment is done based of a set of questions which are assigned a value and together determine a certain total score for the cloud service, leading it to be assigned a certain legal level. The legal level is assigned automatically by ACSmI after the questionnaire has been filled out and will enable application developers to easily determine at what legal compliance level the cloud service is. Based on their compliance needs they can then make appropriate choices and determine the expected legal compliance level as an NFR.

The assessment for the legal level being done through a questionnaire and then giving an automated result is done to ensure that all cloud services are assessed in the same manner. The reason this is done by ACSmI/DECIDE in-house and not by the CSP is twofold:

- Firstly, the questionnaire, in order to come to a meaningful legal level assessment is quite lengthy and therefore might substantially affect a CSPs willingness to endorse its services into ACSmI. Certainly, when ACSmI will be a new product on the market, which will not yet have a substantial user base and therefore not move much volume of the cloud services, this is a real concern: CSPs might not be willing to engage if the threshold is too high.
- Secondly, there would be real bias in the CSP assessing its own contracts. All CSPs would likely end up with legal level tier 1.

Through basing the assessment on a set of objective questions and by making the reasoning behind the assessment available to CSPs, DECIDE aims at making things transparent and fair for the CSP, so no threshold should exist for a CSP to endorse their services into ACSmI. Moreover, the contract with the CSPs described above will foresee an option for CSPs to submit complaints or questions about the legal assessment. DECIDE will strive through specific safeguards and guarantees as to the objectivity of the legal expert carrying out the assessment through filling out the questionnaire. A fully automated assessment of contracts is far out of scope of the DECIDE project and therefore is not considered here (nor does the consortium believe technologies exist that have been proven to be superior to assessment by a human legal expert).

ACSmI will provide a legislation awareness component which will contain the legislation and relevant legal texts, codes of conduct, policies, model contracts and legal definitions in as far as they are relevant which may be taken into account by the legal expert in its assessment through filling out the questionnaire. CSPs and ACSmI users alike may ask to consult the full list of documents.

After a service has been endorsed in ACSmI, there will be two events which will trigger a reassessment of the legal level:

- A CSP changes its contracts, which it will have to report to the entity exploiting ACSmI under the contract with the CSP. In this case only the changed contracts need to be reassessed.
- There is an important change in legislation, case law or interpretation, which requires to reassess all or certain contracts. The assurance policy will detail which changes trigger this process and what needs to happen. Only the affected questions will be re-answered.

Next to the legal assessment, which will be carried out for every service as a basic functionality of ACSmI, there are two more legally relevant aspects:

- There are elements which are legally relevant, but which are not to be assessed through the legal assessment described above. An example of this is certification of the cloud service. While this is relevant for e.g. the GDPR, this is not caught in the assessment above. Other aspects may relate to adherence to standards or codes of conduct (e.g. the Code of Conduct for Data Portability and Cloud Service Switching for Infrastructure as a Service (IaaS) Cloud services by the SWIPO IAAS WG in implementation of the proposed article 6 of the Regulation on the free flow of non-personal data [18] and the data protection code for cloud service providers created by the C-SIG [19]). Work done by Tecnalia in this field under the SMART 2016/0029 tender [27] may be leveraged for DECIDE. ACSmI discovery addresses this gathering information on this from the CSP by asking them to fill out some limited multiple choices, in order to have this information in the ACSmI registry. This may be used for added value services which the customer will have to pay for, enabling setting these types of requirements as extra NFRs, potentially combined with expert advice by the entity exploiting ACSmI on what certifications, standards or adherence to Code of Conducts to look for with CSPs.

- Depending on the data (e.g. sensitive data), the branch of activity (e.g. healthcare, banking) of the application user and the country they are active, different legal requirements and sensitivities may apply. While this is very important, in principle DECIDE and ACSmI do not take responsibility away from either the application developer and/or user or the CSP. It just serves to make the interaction easier. So, in principle, it remains the responsibility of every application developer/user to determine what cloud services are suitable for their business, based on the nature of the data, the branch of the activity and the country in which they are active, where different legal rules and sensitivities will apply. The assurance policy will set out to what extent a customer of DECIDE/ACSmI may rely on the legal assessment, what is assessed and under which legal rules. It will also contain some indicative guidance on what legal level might be better suited for what organization, based on considerations such as the type of data concerned, the branch of activity and the country in which the application developer/user is active, but, as will be specified in the assurance policy, this is merely indicative. Given that such assessments of legal compliance are ad hoc and based on the specific situation of the application developer/user, they cannot be automated in any way and are more akin to standard legal advice. Thus, it is not a part of the basic ACSmI service. However, again it is envisionable that legal and other DECIDE experts may provide added value services, helping application developers/users in determining to what extent they are fulfilling their legal and business requirements through using DECIDE.

To recap, the CSP will, for every service, be required to provide the following information for the service to be endorsed:

- The contractual documents mentioned above
- The answers to open or multiple-choice questions relating to legally relevant matters not covered in the legal level, such as accreditations present for the CSP or service.
- The contract with the CSPs will detail the conditions of this interaction.

In this prototype M24, the focus lies on showing the concept of the legal level and its basic implementation into the tool. Specifically, the M24 release will enable the functionality of uploading the necessary contracts and the functionality to assign a legal level to every cloud service. The reasoning on the legal level and some more information is set out in section 7.1.2 below.

In the next and final iteration of the ACSmI (D5.4, due M30), there will be a full version of the questions that will be used to determine the legal level, as well as additional information on the points mentioned above. Moreover, at that time, a proof of concept will be made available of the full questionnaire for the legal level having been applied to a certain number of selected cloud services which are expected to be important for DECIDE/ACSmI and which should likely be included into ACSmI. At that time, the full functionality of automatically determining the legal level based on the questionnaire and reasoning presented below, and the possibility to set the legal level as an NFR will be implemented in the tool, as well as implementing the role of legal expert.

7.1.2 Determining the legal level of a Cloud service in ACSmI

As explained above, the legal level will be determined on the basis of a questionnaire addressing all legally relevant aspects of the cloud service which can be ascertained from the contracts offered by the CSP which are to be analysed by the legal expert in order to answer the questions.

The legal expert will take into account EU law and national implementation by the Member States to answer the questions. Laws of third countries are not considered and fall under the sole responsibility of the application developer/user. Moreover, for EU law and national implementation, ACSmI (i.e. the entity exploiting ACSmI) and DECIDE merely provide a service which aims to facilitate the choice of cloud services by the application developer/user. DECIDE/ACSmI provides the service ‘as is’ and takes no responsibility for the information provided by the CSP being correct. Nor does ACSmI (i.e. the entity

exploiting ACSmI) or DECIDE make any promises that the legal level represents a situation of compliance with applicable law of the application developer/user or aids in the determination of such.

The legal level is merely a tool to be used by the application developer/user in their legal compliance exercise (NFR) when determining which cloud services to use. As the assurance policy will explain, the use of the legal level as a tool is entirely on their own responsibility and at their own risk. It does not relieve the application developer/user of their obligations under applicable law, nor does it transfer any of this responsibility to ACSmI (i.e. the entity exploiting ACSmI) or DECIDE.

Each question in the legal assessment is written in a multiple-choice format, where the answer closest to the reality of the contract must be chosen by the legal expert. Each of the answers is given a value and each of the questions is given a weight (not all matters are equally important after all). Based on the aggregated score of the cloud service in question, it will be awarded a legal level, ranging from tier 1 to tier 3.

The legal level will measure at least the following:

- GDPR safeguards for data transfers if relevant
- GDPR compliance of data processing agreement with article 28 GDPR
- Presence of a representative in the EU and/or DPO, if relevant
- Applicable law and conflict resolution clauses
- Liability level (caps) and liability clauses
- Exit clauses and penalties (mostly to exclude services that prevent dynamic contracting by imposing penalties)
- Data portability and Cloud switching clauses

The exact values for the levels as well as the threshold value will be determined in D5.4 when the full list of questions has been compiled and tested. The same is true for the exact determination of which level will represent what characteristics. This will depend on the weight being given to the different monitored aspects. GDPR compliance will be rated high because of the potential enforcement risks as well as reputational damage, whereas e.g. applicable law and conflict resolution obligations might be something that carries less weight for most application developers/users and is therefore not as relevant a factor in determining the legal level. Consequently, the weight of the different questions will vary. Weight can be given to different questions in a certain topic or on topic level.

Thus, an agreement that contains no reference to the obligatory obligations imposed by article 28 GDPR will lead to exclusion of the service. The same is true for a description that is so inadequate that it cannot be enforced.

- Obligations that are faulty, provide a low level of protection, or are otherwise clearly less than ideal, but seem to be sufficiently conforming to the applicable law to not be a compliance issue are labelled “low protection”. They add 1 point to the service’s score.
- Obligations that are adequately described and provide a more or less balanced level of protection and obligations and rights are labelled “medium protection”. They add 2 points to the service’s score.
- Obligations that are adequately described, with strong protection & rights for the application controller/user, are labelled “high protection”. They add 3 points to the score.

Some of the obligations may bring specific costs with them, especially some of the GDPR related obligations (e.g. article 28 GDPR audit rights). When the law does not define who should bear these costs, this will be a relevant factor in determining the score. This does not relate to the cost of the service but to the cost of exercising certain rights.

In order to illustrate this, in the annex 2, an example of a cloud service X is presented, where the legal level would be determined by only two of the above aspects, rather than all of them combined, namely:

- GDPR safeguards for data transfers if relevant
- GDPR compliance of data processing agreement with article 28 GDPR

8 Conclusions

This deliverable presents the intermediate prototype of ACSmI. The M24 prototype is an update of the one delivered in the M12 [1] but includes new functionalities and improvements in the three main components that have been developed. Hence, this M24 prototype is composed of three main components: 1) *ACSmI discovery* whose objective is to discover and benchmark services and also to allow the endorsement of cloud services to the service registry by the CSPs, 2) *ACSmI contracting* whose objective is the execution and management of the core functions with respect to the service contracts and 3) *ACSmI monitoring* whose objective is to monitor the fulfilment of the SLAs for the availability, performance and location of each contracted service cloud. In addition to these three main components, the first aspects (Functional and Technical) of the ACSmI billing component are detailed and the approach to be followed to assess the legal compliance is defined. This prototype also implements the integration with OPTIMUS, ADAPT and MCSLA.

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.

8.1 Future work

There is an important part of implementation work that will be included in the next and last iteration of the prototype.

During the next months of the project, the next and last interaction of this prototype due for M30 will be produced. This prototype will include the following open issues:

- Development of the ACSmI Billing component
- Development of the legal compliance approach. In the context of the ACSmI discovery component, the questionnaire to be filled by the legal expert and the mechanism to derive the legal level will be developed.
- To include in ACSmI Monitoring the assessment and alert of the pending NFRs (cost and scalability) and to increase the type of the cloud service (i.e. Databases or storage).
- To increase the CSP connectors supported by ACSmI contracting and to improve the contracting process.

The next version of this deliverable, D5.4, will include all the issues explained in the point above.

9 References

[1] DECIDE Consortium, "D5.2 Initial Advanced Cloud Service meta-Intermediator (ACSmI)," 2017.

[2] DECIDE Consortium, «D3.8 Intermediate DECIDE OPTIMUS,» 2018.

[3] DECIDE Consortium, «D3.14 Intermediate multi-cloud native application composite CSLA definition,» 2018.

[4] DECIDE Consortium, «D3.11 Intermediate multi-cloud native application controller,» 2018.

[5] DECIDE Consortium, «D4.5 Intermediate multi-cloud application deployment and adaptation».

[6] DECIDE Consortium, "D4.8 Intermediate multi-cloud application monitoring," 2018.

[7] DECIDE Consortium, «D2.2 Detailed requirements specification v2,» 2018.

[8] DECIDE Consortium, «D3.5 Intermediate profiling and classification techniques,» 2018.

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

[10] JHipster, "JHipster," [Online]. Available: http://www.jhipster.tech/. [Accessed November 2017].

[11] Spring, "Spring Boot," 2017, [Online]. Available: https://projects.spring.io/spring-boot/. [Accessed November 2017].

[12] Angular, "Angular Docs," 2017. [Online]. Available: https://angular.io/. [Accessed November 2017].

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

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

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

[16] Docker, "Docker," [Online]. Available: https://www.docker.com/. [Accessed November 2017].

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

[18] Webpack, "Webpack Module Bundler," [Online]. [Accessed November 2017].

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

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

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

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

[23] DECIDE Consortium, «DECIDE Description of Action,» 2016.

[24] S. I. WG, «Code of Conduct for Data Portability and Cloud Service Switching for Infrastructure as a Service (IaaS) Cloud services,» [online], available at http://cloudswitching.eu/wp-content/uploads/2018/06/20180611-Cloud-Portability_CoC_Draft_V016-PublicConsultationR1.docx, 2018.

[25] «EU Data Protection Code of Conduct for Cloud Service Providers,» [online], available at https://eucoc.cloud/fileadmin/cloud-coc/files/European_Cloud_Code_of_Conduct.pdf, 2017.

Annex 1 - ACSmI monitoring API REST


```
{
  "swagger": "2.0",
  "info": {
    "description": "ACSmI monitoring API documentation.This API describes the methods to
access the ACSmI monitoring functionalities",
    "version": "0.0.1",
    "title": "acsmi monitoring API"
  },
  "host": "localhost:8080",
  "basePath": "/monitoringmanager",
  "schemes": [
    "http"
  ],
  "tags": [
    {
      "name": "resourcemonitoring",
      "description": "acsmi resourcemonitoring tag"
    }
  ],
  "paths": {
    "/api/resourcemonitoring": {
      "post": {
        "description": "Creates a resourcemonitoring to be included in ACSmI monitoring",
        "tags": [
          "resourcemonitoring"
        ],
        "summary": "createResourcemonitoring",
        "operationId": "createResourcemonitoringUsingPOST",
        "consumes": [
          "resourcemonitoring/json"
        ],
        "produces": [
          "*/*"
        ],
        "parameters": [
          {
            "in": "body",
            "name": "resourcemonitoring",
            "required": true,
            "schema": {
              "$ref": "#/definitions/Resourcemonitoring"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "Resource monitoring Created",
            "schema": {
              "$ref": "#/definitions/Resourcemonitoring"
            }
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden"
          },
          "404": {
            "description": "Not Found"
          }
        }
      }
    },
    "/api/resourcemonitoring/{resourceid}": {
      "get": {
        "description": "Gets the status wrt monitoring of a given resource",
        "tags": [
          "resourcemonitoring"
        ],
        "summary": "getResourcemonitoringstatus",
        "operationId": "getResourcemonitoringUsingGET",
        "consumes": [
          "resourcemonitoring/json"
        ],
```

```
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "resourceid",
        "in": "path",
        "description": "id",
        "required": true,
        "type": "integer",
        "format": "integer"
      }
    ],
    "responses": {
      "200": {
        "description": "Specific resource monitoring is returned",
        "schema": {
          "$ref": "#/definitions/Resourcemonitoring"
        }
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      },
      "404": {
        "description": "Not Found"
      }
    }
  },
  "delete": {
    "tags": [
      "resourcemonitoring"
    ],
    "summary": "deleteResourcemonitoring",
    "operationId": "deleteResourcemonitoringUsingDELETE",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "*/*"
    ],
    "parameters": [
      {
        "name": "resourceid",
        "in": "path",
        "description": "id",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "Resorcemonitoring deleted form the monitoring list"
      },
      "204": {
        "description": "No Content"
      },
      "401": {
        "description": "Unauthorized"
      },
      "403": {
        "description": "Forbidden"
      }
    }
  },
  "put": {
    "description": "Updates the monitoring status of a given resource",
    "tags": [
      "resourcemonitoring"
    ],
    "summary": "updateResource",
    "operationId": "updateResourceUsingPUT",
    "consumes": [
      "resourcemonitoring/json"
    ],
```

```
        "produces": [
          "*/*"
        ],
        "parameters": [
          {
            "in": "path",
            "name": "resourceid",
            "description": "id",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "Resource Status updated",
            "schema": {
              "$ref": "#/definitions/Resourcemonitoring"
            }
          },
          "401": {
            "description": "Unauthorized"
          },
          "403": {
            "description": "Forbidden"
          },
          "404": {
            "description": "Not Found"
          }
        }
      }
    }
  },
  "definitions": {
    "Resourcemonitoring": {
      "type": "object",
      "required": [
        "appdescuri"
      ],
      "properties": {
        "resourceid": {
          "type": "string"
        },
        "appdescuri": {
          "type": "string"
        },
        "status": {
          "type": "string"
        },
        "user": {
          "type": "string"
        },
        "password": {
          "type": "string"
        }
      }
    }
  }

}
```

Annex 2 - Example of a Legal Assessment for a cloud service

This section presents a simple example of a cloud service, where the legal level would be determined by only two of the above aspects, rather than all of them combined, namely:

- GDPR safeguards for data transfers if relevant
- GDPR compliance of data processing agreement with article 28 GDPR

Assuming both aspects have equal weight, the assessment would then look as follows:

Cloud service X legal assessment	
Topic 1 - GDPR safeguards for data transfers if relevant	
Question 1.1: Does the contract contain a transfer of data outside the EEA (i.e. is it mentioned the data will be sent outside the EEA for any reason or duration)? (no points)	
A	No (end of topic, topic is not scored)
B	Yes (go to next question, topic is scored)
C	Not specified (go to next question, topic is scored)
Question 1.2: Does the contract specify transfer safeguards compliant with GDPR?	
A	No => service should not be included
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Topic 2 – GDPR compliance of data processing agreement with article 28 GDPR	
Question 2.1:Is there a data processing agreement? (no points)	
A	No => service should not be included
B	Yes (go to next question, topic is scored)
Question 2.2: Assessment of the description of subject-matter and duration of the processing, the nature and purpose of the processing, the type of personal data and categories of data subjects.	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Question 2.3: Assessment of the obligation to only process data on documented instructions of the controller	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Question 2.4: Assessment of the obligation to have all CSP personnel have committed themselves to confidentiality or for there to be an equal statutory obligation present	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Question 2.5: Assessment of the obligation to take all security measures pursuant to article 32 GDPR (include costs as a relevant aspect)	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)

Question 2.6 Assessment of the obligation to only engage sub-processors based on specific or general written authorization of the controller (with respect for the corresponding conditions) and to impose the same data processing terms on those sub-processors	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Question 2.7: Assessment of obligation to assist the controller with the handling of data subject rights requests (include costs as a relevant aspect)	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Question 2.8: Assessment of obligation to assist the controller with the obligations in article 32-36 GDPR, i.e.: taking security measures, notification of breach to supervisory authority, communication of breach to data subject, data protection impact assessment execution, prior consultation with supervisory authority (include costs as a relevant aspect)	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Question 2.9 Assessment of the obligation to, at the choice of the controller, delete or return the personal data to the controller at the end of the service and delete all existing copies (include costs as a relevant aspect)	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation, e.g. no choice or unreasonable retention beyond allowed exceptions for legal obligations (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Question 2.10 Assessment of the obligation to furnish the controller with all necessary information to demonstrate GDPR compliance, including submitting to inspections and audits (include costs as a relevant aspect)	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
4.11 Assessment of the obligation to inform the controller immediately if in the processor's opinion any instruction infringes the GDPR or data protection law.	
A	Not present (or really inadequate) => Service should not be included.
B	Present, but faulty description/low level of obligation (low)
C	Adequate description & medium level (medium)
D	Adequate description & high level (high)
Maximum points to be obtained:	
	Topic 1: 3
	Topic 2: 30

Scoring and determination of the legal level

If equal weight, then topic 1 points have to be multiplied by 10. Maximum points to be earned would be 60.

The legal level would then be determined as follows:

Less than 11 points or any times "A – service should not be included" = service should not be included

Scoring 11- 27 points = legal level tier 3

Scoring 28-44 points = legal level tier 2

Scoring 45-60 points = legal level tier 1

Of course, in this case, as one can tell, you could have a reasonably bad data processing agreement which however has a clear description of safeguard measures and still end up in a high legal level.

This is where the determination of the weight of certain aspects becomes very relevant. After all, if there is no description of transfer safeguards, the service will in any case be excluded. So, the determination rests on what amount of risk can be taken. A Cloud service with a data processing agreement scoring very low but just acceptable (1 point on all questions in topic 2), but with a high score on the transfer safeguards (3 points) should likely be categorized as level 3.

Thus, a more appropriate weight would be to accord topic 2 five times the weight of topic 1 leading to a multiplication of the points in topic one by 2, rather than 10. Maximum points to be earned would then be 36.

The legal level would be determined as follows:

Less than 11 points or any times "A – service should not be included" = service should not be included

Scoring 11-19 points = legal level tier 3

Scoring 20-28 points = legal level tier 2

Scoring 29-36 points = legal level tier 1

Of course, this is merely a simplified example. The final legal level will take into account all of the other legal aspects and lead to a more nuanced assessment of all legal aspects of a Cloud service.

To determine the weight of the questions and in order to ensure that each legal level corresponds to an easy-to-understand reality, the assurance policy will contain a summary of the specific characteristics to be expected from each legal level, with some indication as to which organizations the level in question may be suited for, based on the type of data processed (sensitive or non-sensitive), the branch of activity (e.g. healthcare or banking vs. manufacturing) and other relevant factors. The assurance policy will define the terms it will use, such as what is considered as sensitive data etc.

The summary will specify per legal topic what can be expected from the legal level.

This will also provide extra guidance to the legal expert to check that the automatically generated legal level follows the minimum guarantees of the legal level obtained.

Taking the foregoing simplified example, it would look somewhat like this:

Legal level	GDPR safeguards for data transfers if relevant	GDPR safeguards for data transfers if relevant	Suited for which organizations or projects
Legal level tier 3	Low protection	Low protection	Suited for non-data driven organizations or projects with little or no sensitive data, low compliance risk or higher risk appetite and limited business complexity. Examples may include non-technical, non-data driven start-ups and SMEs.
Legal level tier 3	Low protection	Medium protection	
Legal level tier 3	Medium protection	Low protection	
Legal level tier 2	Medium protection	Medium protection	Suited for organizations or projects with average data processing activities and average risk appetite, which may process large amounts of data but not large amounts of sensitive data or special categories of data. Examples may include smaller data-driven companies or larger non-data driven companies. Non-sensitive governmental entities may also choose this level.
Legal level tier 2	High protection	Medium protection	
Legal level tier 2	Medium protection	High protection	
Legal level tier 1	High protection	High protection	Suited for organizations or projects which have a low risk appetite and higher compliance risks/burden because of the type of data processed (e.g. health data, financial data) or because of the sector in which they are active, adding regulatory requirements to the mix. Examples include health professionals and hospitals, banks and governmental organizations which also treat sensitive data.

Of course, this is a simplified example and the final prototype delivered in M30 will contain all of the aforementioned legal aspects, leading to a more refined legal assessment of the services. It will also already contain an implementation of the full questionnaire with relation to a few selected Cloud services (at least one service from Google, Amazon, Azure, AIMES and Arsys), as a proof of concept of the assessment.

The assurance policy will contain more detailed information on how to determine the legal level a certain organization needs, referring to existing standards and guidelines. Deliverable D5.4 will contain a proof of concept on this part as well.