Secure Provisioning of Cloud Services
based on SLA Management

SPECS Project - Deliverable 1.1.3

Architecture Design

Version 1.1
15 February 2016

The activities reported in this deliverable are partially supported
by the European Community's Seventh Framework Programme under grant agreement no. 610795.

Deliverable information

Deliverable no.:	D 1.1.3
Deliverable title:	Architecture Design – Finalized Design
Deliverable nature:	Report
Dissemination level:	Public
Contractual delivery:	February 15, 2016
Actual delivery date:	February 15, 2016
Author(s):	Massimiliano Rak (CeRICT), Valentina Casola (CeRICT)
Contributors:	Jolanda Modic (XLAB), Ruben Trapero (TUDA), Silviu Panica (IeAT), Alain Pannetrat (CSA), Andrew Byrne (EMC)
Reviewers:	Silvio La Porta (EMC), Jesus Luna (CSA), Umberto Villano (CeRICT)
Contributors version 1.1	Jolanda Modic (XLAB), Ruben Trapero (TUDA), Silviu Panica (IeAT), Alain Pannetrat (CSA), Andrew Byrne (EMC)
Reviewers version 1.1	Silvio La Porta (EMC), Jesus Luna (CSA), Umberto Villano (CeRICT)
Task contributing to the deliverable:	T1.1
Total number of pages:	43

Executive Summary

This deliverable is the last of three incremental deliverables (i.e., D1.1.1, D1.1.2 and D1.1.3) focused on the design of the full SPECS platform. In this deliverable, we present the final architecture, which is the result of a complex design activity spread over different tasks and coordinated by Task 1.1. For this reason, the SPECS main concepts and the design of all modules are summarized up in this document with latest available design results, while the details of each module are distributed among several deliverables, to which the reader is systematically referred.

In the first year of activities, the documents D1.1.1 and D1.1.2 presented:

- The SPECS glossary;
- The SPECS SLA-based approach;
- The SPECS User and Functional views;
- The main architectural concepts: the SPECS platform and the SPECS framework;
- The preliminary design of the SPECS Platform;
- The preliminary interaction protocols among modules;
- An observatory on available solutions at the state of the art, related to all SPECS main concepts, modules and technological choices;
- Preliminary considerations to develop and deploy real-world SPECS applications on top of the SPECS framework.

The final version of the present document (D1.1.3) replaces the previous versions (D1.1.1 and D1.1.2) and offers a complete overview over the final SPECS architecture. It presents:

- The SLA life cycle, according to results from all other WPs;
- A detailed description of SLA phases: we included details on remediation and renegotiation phases, according to results from WP2, WP3 and WP4;
- A refined description of the SPECS architecture, including the refined SLA platform and Enabling platform design that can now be hosted by different providers, new Security Mechanisms and cross cutting services, according to results from other tasks in WP1 and WP4 and the feedbacks from End-users gained from activities in WP5 and WP6;
- A description of SPECS usage with four close-to-market applications being developed by SPECS partners to exploit project results;
- A final discussion on security assessment and compliance to regulations of the SPECS platform.

Table of Contents

List of Figures

List of Tables

1. Introduction

This deliverable illustrates the final architecture of the SPECS platform, resulting from the design activities conducted in several tasks since the start of the project. The objective of the SPECS project is to design and develop an innovative *Security Platform-as-a-Service* (i.e., the *SPECS platform*) devoted to delivering cloud services characterized by specific security features, negotiated with End-users and guaranteed through SLAs. The SPECS platform is built on top of the *SPECS framework*, which provides all the software tools and components needed to develop applications that offer secure services based on SLAs.

According to the SPECS approach, a cloud service is delivered to customers by following the main phases of the SLA life cycle. According to WS-Agreement[1] [1] as well as the preliminary results of ongoing international standardization efforts towards the specification of cloud SLAs (ISO 19086 [2]), the SLA life cycle involves five main phases, namely *Negotiation*, *Implementation*, *Monitoring*, *Remediation* and *Renegotiation* (see Figure 1).

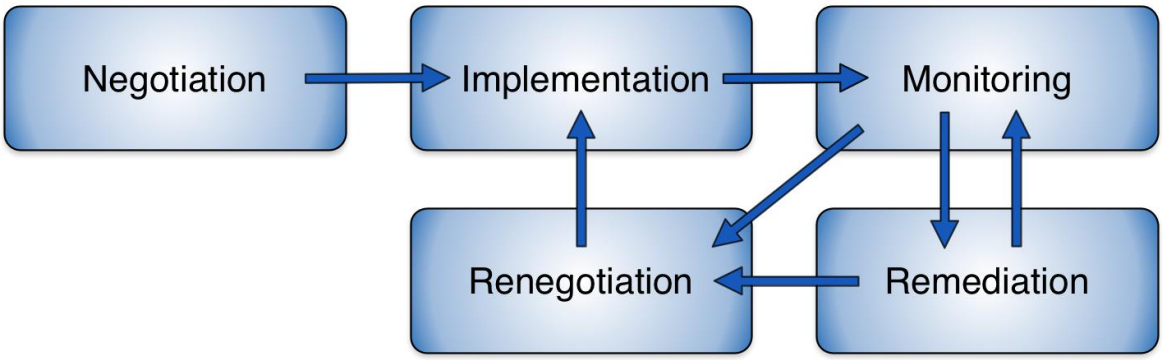

Figure 1: SLA life cycle

During *Negotiation*, the End-user specifies his/her requirements and the provider replies with one or more SLA Offers (cf. WS-Agreement specification), possibly characterized by different levels of guarantees. When an offer is formally accepted, the corresponding SLA is generated and signed by both parties. After the signature, the SLA is ready to be *Implemented*: the cloud service is started and configured based on what has been agreed in the SLA. Then, the SLA is *Monitored* to verify its fulfilment during its validity period. In case of a violation of the SLA, the provider applies proper *Remediation* actions, which typically includes paying penalties. Finally, in some cases, the SLA may be *Renegotiated* to change some of the requirements if the original settings can no longer be enforced on the provider.

The SPECS platform enables the automation of cloud service provisioning, according to the SLA life cycle discussed above; the goal of this document is to summarize how this task is accomplished by the SPECS platform, giving a clear view of the design of the SPECS solution and reporting the latest advances in the project. In particular, we will present the recent updates on the SLA life cycle and on the five modules that have been developed in other tasks. Indeed, thanks to the incremental release of software components and artefacts we gained significant feedback from development activities and from End-users, that enriched our analysis and led to an updated version of the proposed architecture.

[1] WS-Agreement is the only standard currently supporting a formal representation of SLAs and defining a protocol for their automation.

In the reminder of this document, after illustrating the relationship with other deliverables in Section 2, in Section 3 we summarize the SPECS main concepts by discussing the objectives of the project and the related challenges. We provide an overview of the SPECS platform and illustrate how it is meant to be used. Related to this, we also report the main SPECS usage trough three different interaction models and introduce a set of relevant applications that are currently available in the SPECS solutions portfolio[2].

In Section 4, we briefly analyse the updated behaviour of the different modules of the SPECS platform during the SLA life cycle phases that, with respect to previous versions, was enriched with a refined discussion on remediation and renegotiation phases. Furthermore, End-users, which were primary involved in the validation of requirements elicited in the first year and in the definition of close-to-market applications, along with prototype implementation activities, provided feedback to improve the design proposed at end of the first year. In this deliverable, we present the final architecture and we outline the updates provided in the second year of activities.
As already stated, this deliverable presents the current results derived from the design activities in all ongoing tasks and provides referencing to the deliverables where further details and the coverage of the requirements, are available.

To complete the overview of the SPECS framework, in Section 5 we present four applications being developed by the SPECS consortium to build a close-to-market solution portfolio, these examples also represent different usage scenarios (interaction models) where SPECS can be implemented. Of these, the Secure Web Container is already available as a prototype and it is widely described in D5.1.2 and D5.1.3 while the others are under development. Furthermore, in Section 6 we report some considerations on the security assessment of the services offered through SPECS and on the compliance to regulations issues arising when using SPECS due to existing geographical regulations and laws.

Finally, in Section 7, some conclusions are drawn and further references to external links and to the SPECS source code repository is reported.

2 http://www.specs-project.eu/

2. Relationship with other deliverables

This deliverable reports the final architecture of the SPECS platform and is the result of the activities of Task 1.1 on "Architecture Design". The Task 1.1 is strongly connected with all other tasks and deliverables on design activities:

- D1.3, focused on the design of interaction protocols among modules,
- D1.4.1, focused on the design of shared API and core services,
- D1.6.1, focused on the testbed design,
- D2.2. and D2.3.1, focused on the design and the implementation of the negotiation services,
- D3.3, focused on the design of monitoring services,
- D4.2.2, focused on the design of enforcement services and security mechanisms.

Hence, apart from deliverable D1.1.2, which represents the intermediate version of D1.1.3 released at month 12, this deliverable takes as input, deliverables D1.3, D1.4.1, D1.6.1, D2.2.2. D2.3.1, D3.3 and D4.2.2, as depicted in Figure 2.

Concurrently, the architecture design resulting from this deliverable is an input to all tasks focused on the definition of the usage scenarios and of the applications, both for demonstration purposes and industrial exploitation, i.e.:

- D5.1.2 and D5.1.3, which focus on usage patterns and example applications,
- D5.2.1 and D5.2.2, which focus on the description of how the SPECS platform is able to respond to security incidents through concrete application scenarios,
- D5.3, which focuses on negotiation of SLAs in a specific application for next generation data centres.
- D5.4, which focuses on the description of a specific application that offers an authentication-authorization-auditing service on demand.

The affected deliverables are reported in Figure 2 and in Section 4.2, Table 1, we detail the pointers to deliverable describing in detail each component/artifact developed.

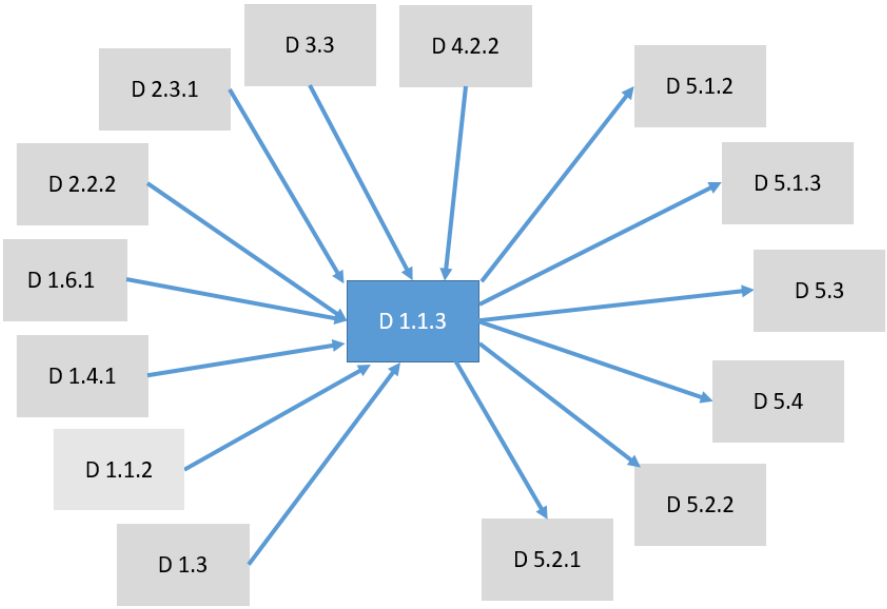

Figure 2: Relationship with other deliverables

3. SPECS general concepts and overview

This section provides an overview of the main concepts related to SPECS. In particular, we review the refined SLA life cycle considered in SPECS, introduce the SPECS Security SLA model, discuss the high-level architecture of the SPECS platform and analyse the main interaction models.

3.1. SPECS Security SLAs and the SLA life cycle

According to current standards and initiatives on SLAs (WS-Agreement, ISO 19086, etc.), the SLA life cycle can be characterized by different states. This state diagram is a fine-grained description of the SLA life cycle, as illustrated in the introduction (see Figure 1). In Figure 3, we report the state diagram for the SLA life cycle. It has been used a UML standard notation for state machine diagrams.

In the figure, rounded edge rectangles represent the system states, while arrows depict transitions. Each transition points out the movement between the states and the conditions in which it can occur. This rules are expressed by a label on the arrow, divided in three parts in the form "trigger-signature [guard] / activity".

The trigger-signature describes the event(s) that causes the transition. The guard is the boolean condition to satisfy, in order to have the transition. Finally, the activity is the sequence of operations that takes place during the transition.

The diagram is an enrichment of the WS-Agreement SLA state diagram [5] (even if the terminating states collapse in a single state). Moreover, it outlines aspects related to possible violation, proactive reactions and re-negotiation of the SLAs, that state of the art standards support.

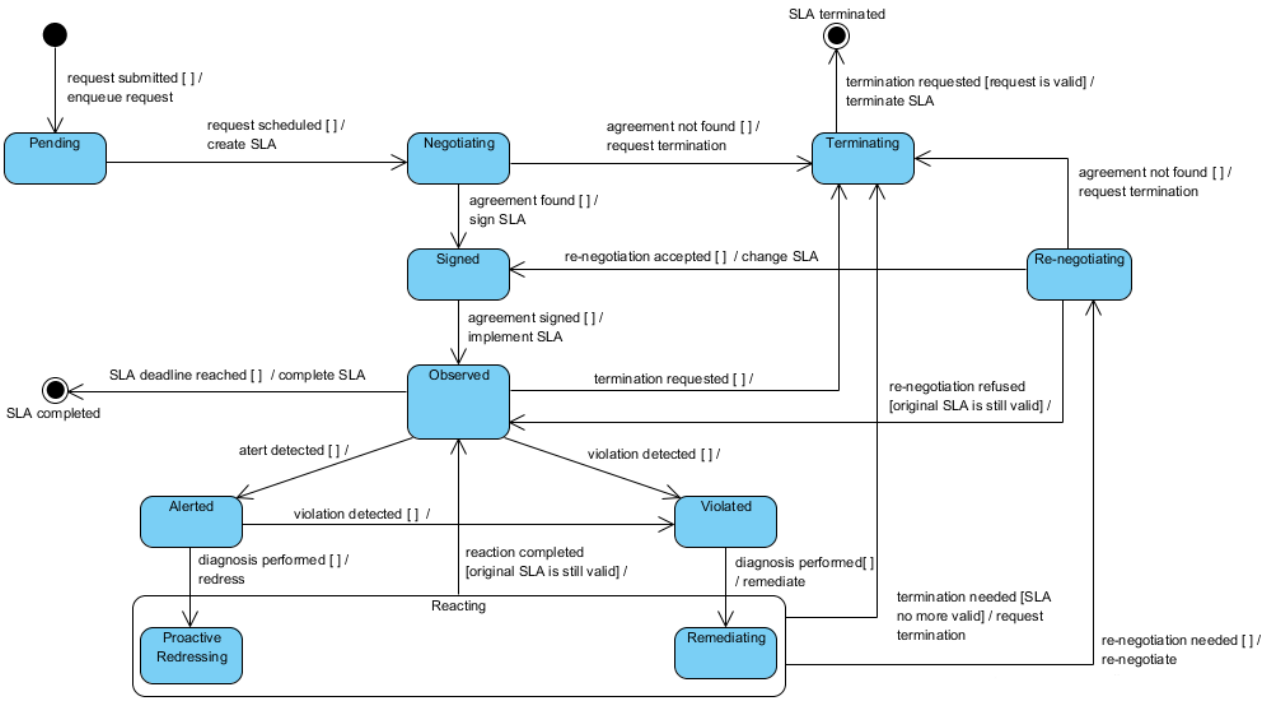

Figure 3: The SLA life cycle

The SPECS platform intends to deliver cloud services to End-users by following the phases of the SLA life cycle. To achieve this, the following functionalities are provided by SPECS:

- The service brokering and provisioning, according to SLA life cycle
- The management of SLAs and of their life cycle;

- The user-centric negotiation of cloud services and related security features (SLA negotiation);
- The comparison of different service offers from providers with respect to the associated level of security (SLA evaluation);
- The automatic provisioning of a negotiated service through the activation and configuration of the software components devoted to offer the cloud service itself (e.g., a cloud storage service) and the negotiated security features (e.g., an end-to-end encryption feature) (SLA implementation);
- The continuous monitoring of the delivered services, in order to verify that the conditions stated in the related SLAs are correctly fulfilled (SLA monitoring);
- The management and the reaction to security incidents, in order to prevent SLA violations and to recover from them when possible (SLA remediation);
- The renegotiation of an SLA in case of an explicit request by the End-user or in case of a violation (SLA renegotiation).

Providing these functionalities is challenging due to the lack of effective solutions for the management of security-oriented SLAs. In particular, we proposed a solution based on the following key ideas:

- ***Declaring*** the functional and non-functional (i.e., security-related) features provided by a service in an SLA (useful for negotiation), in order to enable a Security SLA to support common best practices based on standard security controls (e.g. NIST Control Framework SP800-53, CCMv3.0)
- ***Quantifying*** the level of security provided with a service (useful for evaluation and monitoring), adopting: (i) *security metrics* defined according to standards and (ii) associating the security metrics to the security controls granted in the declarative part of the Security SLA.
- ***Automatically provisioning*** a service based on the functional and non-functional requirements stated in the Security SLA (useful for enforcement and remediation),

The declaration of the features offered by a service (both functional and non-functional related to security) is enabled by the *SPECS SLA conceptual model,* illustrated in detail in deliverable D2.2.2 and reported in Figure 4.

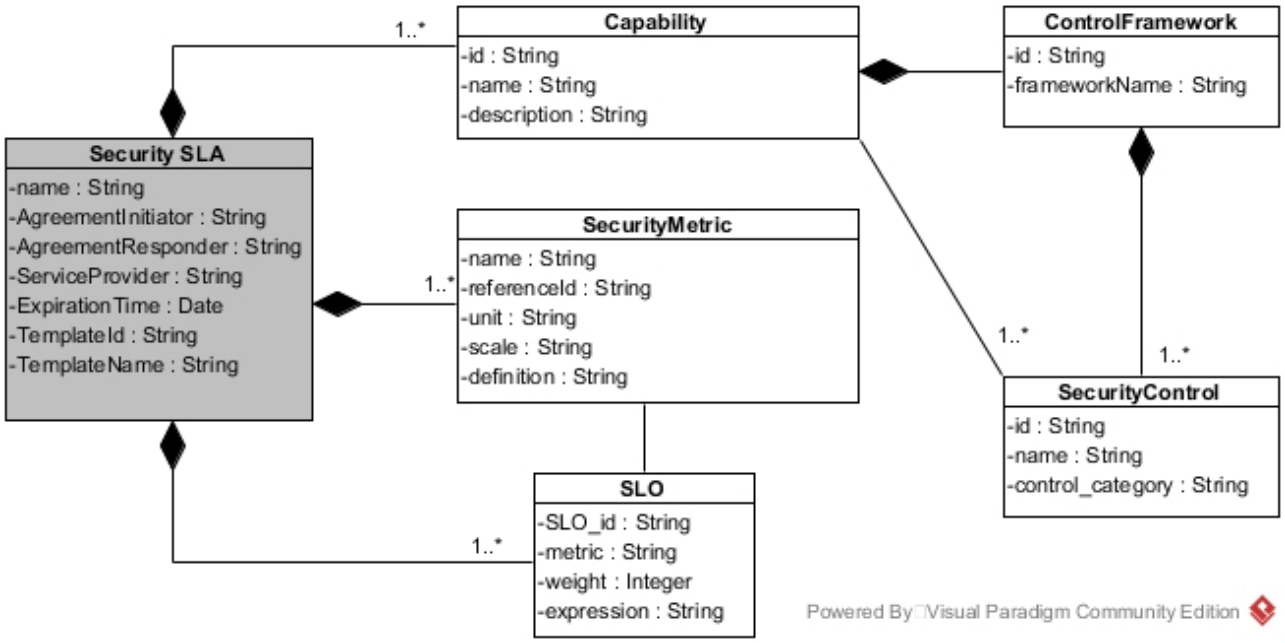

Figure 4: The SLA model

The model allows to specify security-related attributes of a service by means of *security capabilities*. They are expressed in terms of *sets of security controls,* in compliance with NIST definition [3]. Our model supports multiple security controls' definitions, according to the chosen *security control framework*.

The declaration of the security capabilities and of the security controls enforced by a service in an SLA enables *the automatic evaluation* of the associated security level and the comparison of services available from different providers.

This task, performed during the negotiation phase, is accomplished in SPECS by a Security Reasoner, which is able to evaluate different offers and assign them a score based on the level of security provided. The reader is referred to deliverable D2.2.2 for a complete discussion of the requirements of the Security Reasoner and of the techniques used for the evaluation.

The quantification of the security level associated with a service is useful not only for the evaluation task, but also for specifying the guarantees associated with it and for monitoring that the guarantees are met. With respect to this, the following concepts are modelled (see Figure 4):

- The *security metrics* are used to specify *measurable* aspects related to the declared security capabilities, in order to enable their monitoring (e.g., "availability" is a measurable attribute that can be used to monitor the correct provisioning of a "resilience to attacks" capability). Moreover, in SPECS, security metrics also include *configurable* parameters associated with the enforcement of some declared capabilities (e.g., "scanning frequency" is a configuration parameter related to a "vulnerability detection" capability);
- The *SLOs* are logic conditions defined over security metrics (e.g., "availability >99.9"). SLOs are negotiated by End-users and state the desired level of security; they are used to configure the services being delivered and monitored during service operation.

Finally, the automatic provisioning of services that fulfil specific security requirements (i.e., able to enforce specific security capabilities) is enabled by:

- The availability of suitable *security mechanisms* that offer the required security capabilities and can be activated in an as-a-service fashion
- The adoption of a cloud automation technology, which allows the deployment and configuration of software components in a cloud environment.

To satisfy the first point, a set of pre-defined security mechanisms was developed, capable not only of enforcing particular security capabilities specified in an SLA, but also monitoring related security metrics in order to ensure the fulfilment of the SLA itself. Such mechanisms can be activated on demand and configured based on End-users' requirements.

In relation to the second point, as discussed in deliverable D4.2.2, Chef [6] was chosen as the underlying technology used to deploy and configure not only the software components that implement the above mentioned security mechanisms, but also the core components of the SPECS platform itself. In order to manage the deployment and configuration of the whole SPECS infrastructure and the security-related components, we developed proper cookbooks, stored in the Chef server and executed on demand.

The concepts discussed in this section represent the basis for a clear understanding of the SPECS framework. These concepts will be presented in more detail in the following sections.

3.2. The SPECS framework and its usage: high-level architecture

In this section, we present a very high-level view of the framework to easily understand the concepts of SLA-based service provisioning trough SPECS applications. In Figure 5 we show the actors involved and the main building blocks of the architecture when SPECS is hosted by a public CSP and provides security services to protect cloud resources hosted by an external Cloud Service Provider. Deliverables D1.2 and D5.1.1/D5.1.2 describes the SPECS framework scenarios and actors in detail.. The SPECS framework has four main actors involved:

- The ***End-user***, who represents the customer of the cloud services covered by Security SLAs;
- The ***External CSP***, a(typically public) cloud service provider that offers cloud resources and infrastructure services (typically whitout security grants);
- The ***SPECS Owner***, which manages the SPECS platform through which security-enhanced services are delivered to End-users.
- The **SPECS Developer**, which develops new SPECS applications and new security mechanisms.

The End-user (i.e. the SPECS customer in Figure 5) negotiates his/her security requirements through a suitable interface provided by the **SPECS application**, which runs on top of the **SPECS platform** and orchestrates its services to fulfil the End-user's requests.

The SPECS platform is composed of five modules: *Negotiation, Enforcement, Monitoring, SLA Platform* and *Enabling Platform*. The first three modules (called the *core* modules) offer services for the management of the respective phases of the SLA life cycle, while the SLA Platform and the Enabling Platform are responsible for the management of the SLA life cycle itself and for the deployment and execution of all software components needed for the cloud services, respectively.

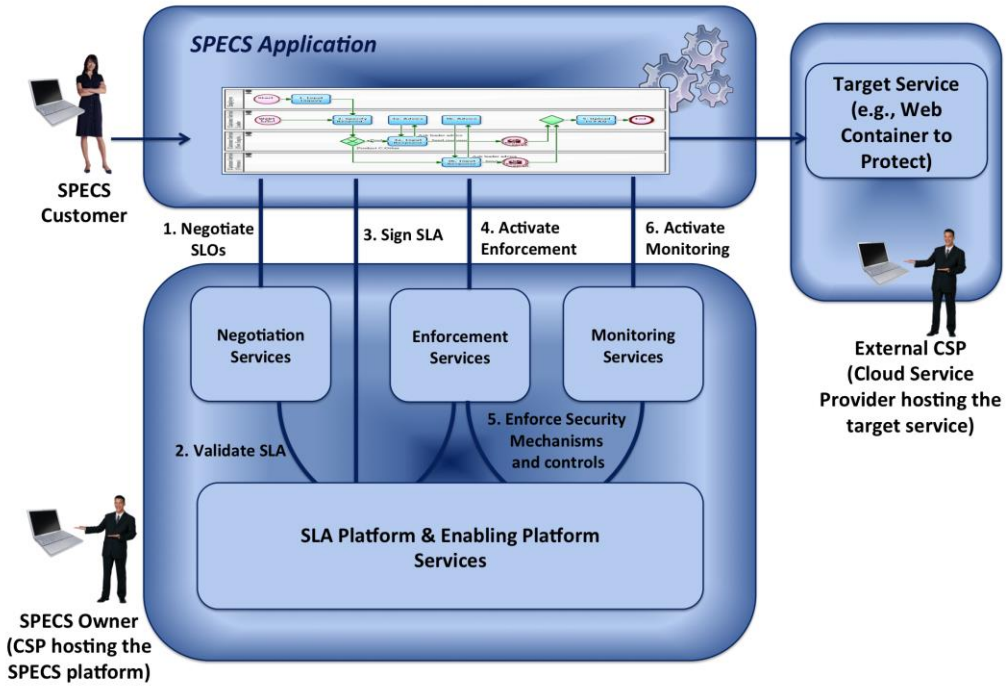

Figure 5: The SPECS high-level architecture

As shown in Figure 5, during negotiation, based on the requirements expressed by the End-user (step 1), a set of valid (i.e., feasible) offers are identified by the Negotiation module (step 2). These offers are represented by different *supply chains*, i.e., by different services compositions that are able to fulfil the End-user security requirements. The offers are ranked by the Negotiation module based on the associated level of security and returned to the End-user, who makes a selection. The selected offer is used to build an SLA that is signed by both parties (step 3) (i.e., the End-user and the SPECS Owner) and then implemented. The SLA implementation phase (step 4) is managed by the Enforcement module, which acquires resources from external CSPs and activates suitable components that provide, in an as-a-service fashion, the security features negotiated in the SLA (step 5). At the same time, suitable services are activated for monitoring the specific parameters included in the Security SLA (step 6). Monitoring data are collected by the SPECS Monitoring module and analysed against a monitoring policy: if an incoming violation of the signed SLA is suspected, it is forwarded to the Enforcement module, which performs a diagnosis and, in the case of an actual violation, suitable countermeasures may be adopted, consisting in re-configuring the service being delivered, or in applying remediation actions.

The behaviour described above is realized by properly orchestrating the SPECS core services through the SPECS application. Applications are built by **developers** (not shown in the figure) exploiting the software tools and services provided by the **SPECS framework**. The developer is a *cloud service partner* that supports the SPECS Owner in the development of SPECS applications and in the delivery of security-enhanced cloud services. In order to enable this, the SPECS framework provides a repository of *security components* that implement both security mechanisms and monitoring systems deployed and activated during the enforcement and monitoring phase respectively. The current implementation of the SPECS framework provides a set of security mechanisms associated with security controls and metrics, as defined in the NIST control framework. These can be easily extended by any application developer in order to

provide new security services or even include new security capabilities or different security control frameworks to build new SLAs.

We identified three different ways to use the SPECS framework, referred to as **interaction models** (see Deliverable D1.2 for further details).

In **Interaction Model 1** (IM1), the SPECS platform is hosted on a public CSP and the SPECS Owner acts as a 3rd party to provide the End-users with cloud services protected by a Security SLA. In this case, the target services are offered by External CSPs and the SPECS Owner acts as a broker to help the End-users select the best offer based on his/her requirements, and he/she may enhance existing service security applying SPECS security mechanisms.

In **Interaction Model 2** (IM2), the SPECS platform is hosted on the resources of a cloud service provider (typically, a private one) directly managed by the SPECS Owner itself. The SPECS Owner can use the platform to enhance its own target services with security capabilities and SLA management features.

Finally, in **Interaction Model 3** (IM3), the End-user is also the SPECS Owner, and the SPECS platform is hosted on the End-user's resources and is used to support him/her in the selection of providers from which to acquire desired services, and in the management of the acquired services.

These interaction models, corresponding to different business models, were instantiated through a set of applications that constitute the SPECS solution portfolio. The portfolio consists of a set of close-to-market software products being developed by SPECS partners (CeRICT, CSA, EMC and XLAB) to provide specific services of interest for a set of relevant End-users. The portfolio solutions are briefly described in Section 5 to provide a concrete example of SPECS usage and they are reported in details in D5.1.2, D5.2, D5.3 and D5.4 and D6.2.2.

4. The SPECS platform: behaviour and architecture

The goal of this section is to present the final SPECS platform architecture and behaviour in managing the SLA phases. With respect to the previous version delivered at month 12, in D1.1.2, some updates have been provided. Both behaviour and design were refined in the second year of the project thanks to continuous feedbacks from end-users and inputs from ongoing activities on all Core modules, specifically from Negotiation, Enforcement, Monitoring and SLA platform modules.

In the following subsections, we first present the final SPECS platform behaviour that has been improved with details on the remediation and renegotiation phases, as discussed in deliverables D4.3.2 and D2.2.2; then we will present a refined design of the SPECS architecture that takes into account updates and feedbacks from development activities reported in D1.4.1, D1.6.1, D3.3, D4.2.2 and from the final definition of inter-modules interaction protocols, defined in D1.3. As illustrated, there are few updates in almost all modules and in Section 3.2 we present the whole SPECS platform final design.

4.1. SPECS behaviour

This section recalls the main operations performed by the SPECS modules during the negotiation, enforcement, monitoring, remediation and renegotiation phases, in order to highlight the responsibilities of the respective components of the architecture. The overall behaviour is illustrated in Figure 6 with high level sequence diagram that highlights the main interactions among modules during the different phases; these, in fact, are included in corresponding UML boxes. In particular, the figure depicts not only the main SPECS modules (Negotiation, Enforcement and Monitoring), but it also outlines the role of the SPECS application. The SPECS application acts as an interface for:

- End-user, orchestrating core services,
- External CSP(s), providing the resources for the execution of the target services requested by the End-user.

The definition and refinement of details related to all phases has been a core objective of many different deliverables. The following subsections present an overview of the main behaviour of modules within each SLA phase; furthermore, a list of all references will be reported in order to give the final pointers to deliverables where the complete set of details is available. In addition, a more in-depth discussion of all the inter-modules interactions is reported in D1.3, while the internal processes are described in the dedicated deliverables.

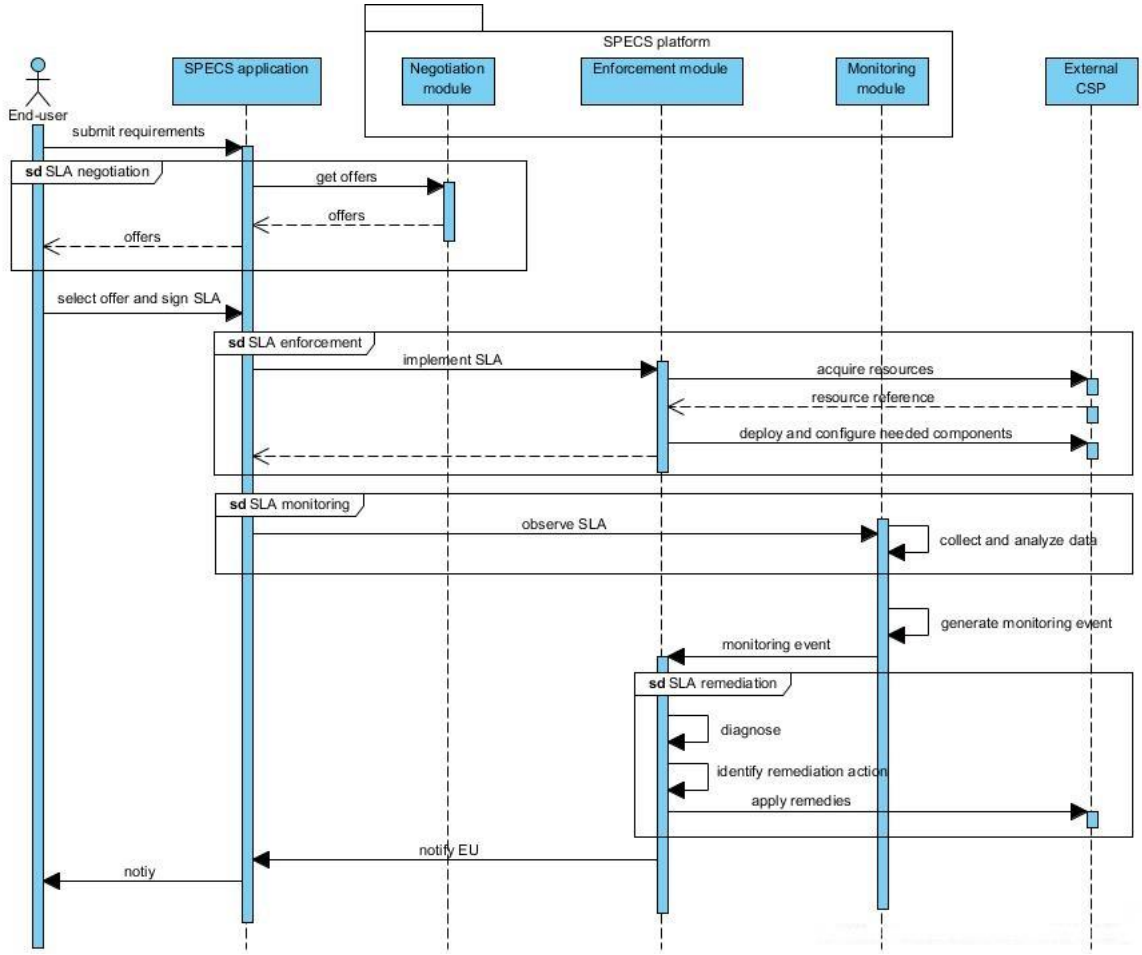

Figure 6: High-level SPECS behaviour

4.1.1. Negotiation phase

The negotiation process allows the End-users to negotiate the security features of the cloud services they wish to acquire. At the end of the process, a Security SLA including the negotiated security guarantees is built and signed.

The End-user starts the process by accessing the SPECS application interface that can be available via a web portal. The negotiation request is forwarded to the Negotiation module, which retrieves all the available SLA templates, representing the available security services. These services are displayed to the End-user through the SPECS web portal, who then selects the services that satisfy the relevant security requirements. Afterward, the End-user provides his/her preferences for security capabilities, security controls and security metrics (more detail can be found in D2.2.2). Once provided with the End-user's requirements, the Negotiation module invokes the Enforcement module for obtaining a set of valid supply chains, represented by different combinations of available CPSs and SPECS services that are able to fulfil the End-user's request.

Supply chains are built by the Enforcement module following an optimization process, described in deliverable D4.3.2, which consists of finding a list of possible deployments of all SPECS components involved in the supply chain. The Enforcement module also checks, the available cloud resources (i.e., available CSPs and offered VMs), the available security mechanisms devoted to offering the requested capabilities and to monitor the identified metrics. The optimization process returns a set of valid supply chains that can be actually implemented.

These supply chains are used by the Negotiation module to build a set of SLA offers that are ranked according to security assessment algorithms based on the quantification of metrics and on the quantitative/qualitative requirements provided by the End-user (see D2.2.2 for details). The process of negotiation ends when the End-user selects and sign an SLA Offer among a list produced by the Negotiation module. The SPECS Owner, then, signs the selected SLA and implements it, activating the Enforcement module.

4.1.2. Enforcement phase

After an SLA is signed, the Enforcement module prepares and implements all the components, which will be deployed and configured with the parameters specified in the SLA.

In other words, Implementation phase is in charge of provide and configure cloud services (eventually brokered from an External CSP) in order to respect the security properties agreed in the SLA. In order to provide such automatic enforcement we adopted one of the state-of-the-art solution for software deployment: Chef [6]

SPECS security mechanisms and components are provided as "Chef cookbooks", that enable the full automatization of SPECS components installation and configuration over a cloud provider. Each security mechanism will be enriched with a set of metadata (managed by our SPECS Platform) that summarize the information needed to correctly configure and install the components, according to SLA requests.

The enforcement process, consists of two phases, namely planning and implementation:

- During the planning phase, the Enforcement module builds an implementation plan for the SLA. That is, it translates high level End-user requirements (reported in the Security SLA) into an implementation plan, which includes the cloud services to be brokered and the security mechanisms to be configured (through Chef) on top of the acquired cloud resources.
- During the implementation phase, the implementation plan is executed, which entails acquiring and configuring external cloud resources (Target Services) and (re)configuring internal components. It also updates the Monitoring policy and activates proper monitoring system, associated to security metrics to guarantee security SLO.

Details on this phase are available in D4.2.2 and D4.3.2. After the SLA is successfully implemented, the SLA enters in the observed state and the monitoring phase begins.

4.1.3. Monitoring phase

The main objective of the monitoring module is to collect and filter the information gathered from the Target Services based on the configuration set up during the Enforcement phase. The monitoring process involves configuring suitable monitoring systems (based on existing monitoring tools and possibly deployed on the resources to be monitored) to filter the raw data coming from monitoring agents and/or probes, and to report back any event that may be suspicious.

Moreover, the SPECS Monitoring module is configured to route the events coming from these monitoring systems to specialized components that aggregate the events based on predefined aggregation rules or to archive the events for later analysis. The aggregated events are filtered based on a Monitoring Policy (MoniPoli) to detect possible SLA alerts or SLA violations. In case of such events the Enforcement triggers the remediation phase.

Details on this phase are available in D3.2 and D3.3, while the details related to the configuration of the MoniPoli are reported in D4.3.2.

4.1.4. Remediation phase

When the Monitoring module detects suspicious behaviour, the event is notified to the Enforcement module, which triggers a remediation activity, involving two phases:

- **Diagnosis phase:** the notified events are diagnosed (i.e., they are analysed and classified). After false positives are discarded and the event is either labelled as an alert or a violation, the remediation of the affected SLA is triggered.
- **Remediation phase:** A remediation plan is built for the alerted/violated SLA according to security guarantees specified in the SLA, and to the alert/violation details provided by the diagnosis phase.

After the diagnosis is completed and the remediation plan is built, remediation is performed by following a process similar to the one discussed for the implementation phase.
The details of the remediation process can be found in D4.3.2.

4.1.5. Renegotiation phase

As previously mentioned, SPECS considers two types of renegotiation, depending on the actor that triggers the process. A renegotiation occurs when an enforced and signed SLA becomes invalid, either as a result of a violation of any of the SLOs (CSP triggered renegotiation) or as a result of the explicit decision of the End-user (because he/she wants to add a new capability or change some specific SLO).

The renegotiation process is similar to the negotiation process, with the difference that the End-user has not to provide requirements for every security aspect but just those that he/she wants to modify. The details of the renegotiation process are dealt with in D2.2.2.

4.2. SPECS platform architecture

In this section we present an overview of the final architecture of the SPECS platform. In particular, with respect to the intermediate solution presented at M12, the SLA Platform the Enabling Platform and the available Security Mechanisms have been refined thanks to the feedback received from stakeholders, during the second year activities.
The definition of the Platform architecture was a complex activity, which was carried out over different tasks with architecture details distributed across different deliverables.
As illustrated in Figure 7. The final SPECS architecture is composed of the following main modules, orchestrated by a SPECS Application:

- Negotiation module,
- Enforcement module,
- Monitoring module,
- SLA Platform module,
- Vertical Layer,
- Enabling Platform module.

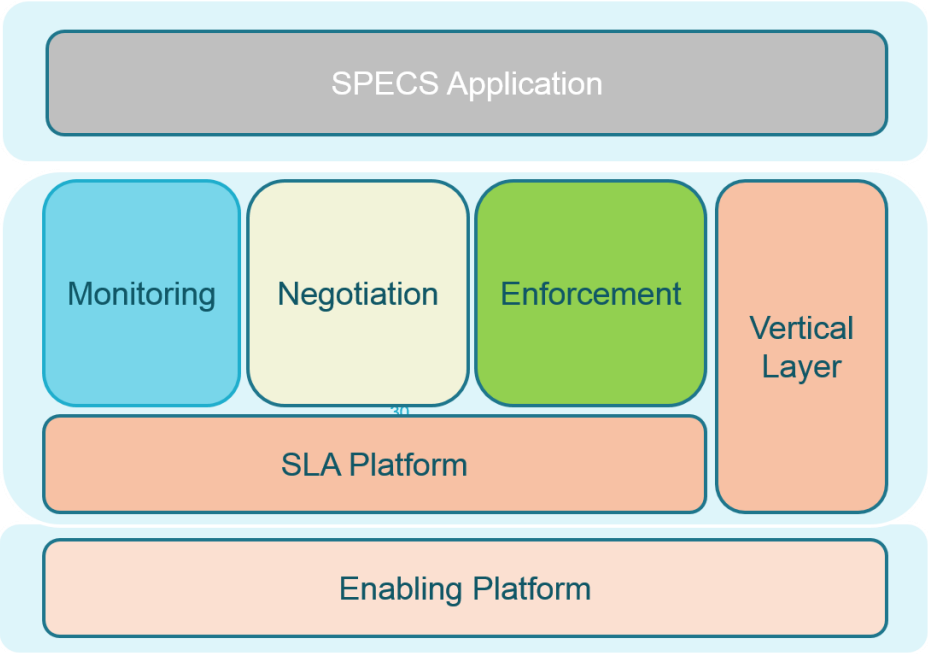

Figure 7: SPECS Platform Architecture

For each module, we recall the main functionalities offered and show the associated component diagram. In order to help the reader correctly locate the information related to each module, the following table summarizes the main components of the Platform architecture and reports a synthetic mapping between the modules and the deliverables in which all related design and implementation information are included. In the table, deliverables in parentheses will be released at M30.

Module	**Component**	**Architecture Design Details**	**Implementation Details**
SLA Platform	SLA Manager	D1.4.1	D1.4.2
	Service Manager	D1.4.1	D1.4.2
	Metric Catalogue	D1.4.1	D1.4.2
	Interoperability layer	D1.4.1	D1.4.2
Vertical Layer	User Manager	D1.4.1	D1.4.2

	Auditing	D1.4.1	D1.4.2
	Credential Service	D4.2.2	D4.4.2
	Security Tokens	D4.2.2	D4.4.2
Enabling Platform	Launcher	(D1.6.2)	D1.6.1/ (D1.6.2)
	Custom OS	D1.6.1 (D1.6.2)	D1.6.1/ (D1.6.2)
	Core Repository	D1.6.1 (D1.6.2)	D1.6.1/ (D1.6.2)
	Mechanisms Repository	D1.6.1 (D1.6.2)	D1.6.1/ (D1.6.2)
Negotiation Module	SLO Manager	D2.3.1 / (D2.3.2)	(D2.3.3)
	Supply Chain Manager	(D2.3.2)	(D2.3.3)
	Security Reasoner	D2.3.1 / (D2.3.2)	(D2.3.3)
Enforcement Module & Security Mechanisms	Planning	D1.1.2/D4.2.2	D4.3.2
	Implementation	D1.1.2/D4.2.2	D4.3.2
	Diagnosis	D1.1.2/D4.2.2	D4.3.2
	RDS	D1.1.2/D4.2.2	D4.3.2
	Broker	D4.2.2	D4.3.2
	WebPool	D4.2.2	D4.3.2
	TLS	D4.2.2	D4.3.2
	SVA	D4.2.2	D4.3.2
	DBB	D4.3.2	D4.3.2
	E2EE	D4.2.2	D4.3.2
	AAA	D4.2.2	(D4.3.3) / (D5.4)
	DoS	D4.2.2	(D4.3.3)
Monitoring Module	Event Hub	D3.3	D3.4.1
	Event Aggregator	D3.3	D3.4.1
	Event Archiver	D3.3	D3.4.1
	Monitoring Policy Filter	D3.3	D3.4.1
	SLO Metric Exporter	D3.3	D3.4.1
Interaction Protocols	(ALL) Module APIs	D1.3	D1.3

Table 1: Architecture design and implementation documents

4.2.1. SLA platform module

The SLA Platform module is mainly responsible for the management of the SLA life cycle and provides all the functionalities needed to guarantee the interoperability among the Enforcement, Negotiation and Monitoring modules.
The SLA Platform module provides the following functionalities:

- **Enables the management of SLAs**: It allows the creation of new SLAs (possibly based on an SLA Template) during negotiation, the retrieval of SLAs for processing in any phase of their life cycle, the updating of their content (only before signature) and their deletion (only if they have not been signed yet).
- **Enables read/write access to the SLA state:** It enables access to the state of an SLA and /or to update it according to the life cycle reported in Figure 3.
- **Manages security mechanisms:** Security mechanisms are software tools responsible for the enforcement of specific security capabilities or the monitoring of proper security metrics. The SLA Platform module provides the functionalities for managing security mechanisms and associated metadata, specifying information related to the software components that implement the mechanisms themselves and possible associated deployment constraints. Such information is used by the Enforcement module during the Planning phase.

- **Manages security capabilities and security metrics:** The SLA Platform module stores and maintains all the information related to security capabilities and security metrics, specified according to the conceptual model described in deliverable D2.2.2.
- **Allows interoperability among core modules:** It provides a virtual interface that allows the communication among the components.

The above features are offered by the components depicted in Figure 8:

- **SLA Manager:** Provides the basic functionalities to manage the SLAs and their life cycle;
- **Service Manager:** Responsible for the management of security mechanisms (with associated metadata) and security capabilities;
- **Metric Catalogue:** Stores and maintains information related to security metrics;
- **Interoperability Layer:** Enables the interoperation and decoupling among all components by offering a single access point for all APIs.

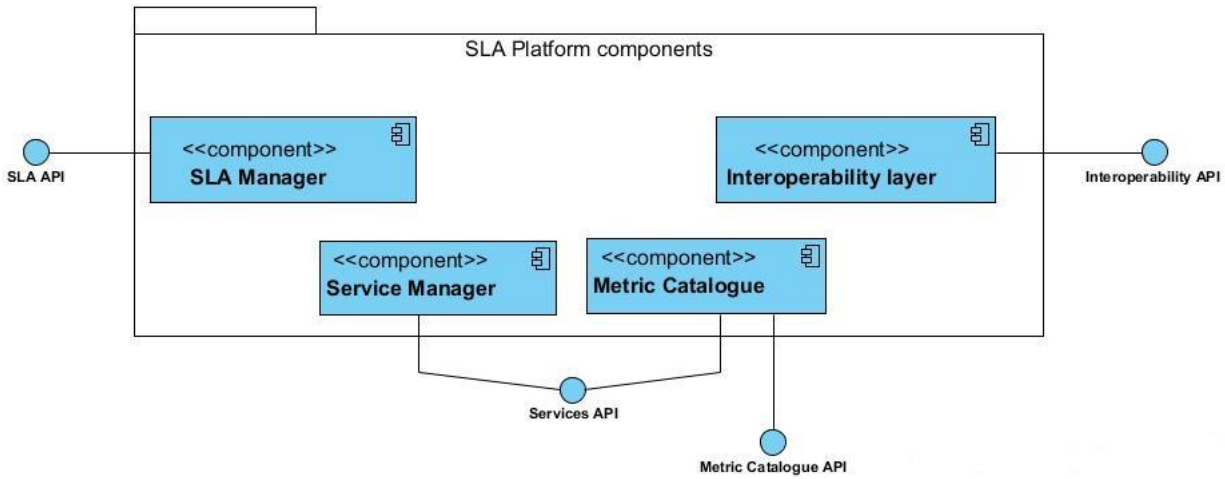

Figure 8: SLA Platform module component diagram

Note that this architecture is currently an updated version of the one presented in Deliverable D1.1.2, which considered only three components, i.e., the SLA Manager, the Services Manager and the Interoperability layer. In the new version, we introduced the Metric Catalogue component to manage all the operations related to security metrics, which previously were handled by the Service Manager. The Service Manager, together with the Metric Catalogue component, offers the Services API described in Deliverable D1.3, while the SLA Manager and the Interoperability layer offer the SLA API and the Interoperability API (also described in D1.3), respectively.

As shown in the figure, the Metric Catalogue also exposes a Metric Catalogue API, which can be used independently of SPECS, exposing functions for the management of security metrics represented based on the RATAX standard format[3]. This updated API is described in D1.4.1.

A detailed discussion of the updates to the original SLA Platform architecture is reported in deliverable D1.4.1, which also contains the description of the internal architecture of each of the SLA Platform components. In Deliverable D1.4.2 instead, we report the SLA Platform implementation details.

[3] Cloud Computing Service Metrics Description, Draft, NIST Special Publication 500-xxx

4.2.2. Vertical layer

As reported in Deliverable D1.1.2, the **Vertical layer** offers some cross-cutting services used by the SLA Platform and other modules. The Vertical layer architecture has been updated, too. In particular, all functionalities related to storage and service management requirements are offered by the SLA Platform core components.
The final set of functionalities offered includes:

- *Management of users platform.*
- *Collection of logs for auditing purposes.*
- *Protection of internal communications among SPECS components.*
- *Management of credentials.*

Figure 9 shows the Vertical layer, which includes the following components:

- **User Manager:** Manages the SPECS platform's users and provides basic authentication and authorization features;
- **Auditing:** Manages logs for auditing purposes;
- **Security Tokens:** Protects communication channels among internal components via secure tokens;
- **Credential Service:** Provides a service to securely store and manage the credentials needed to access resources from External providers.

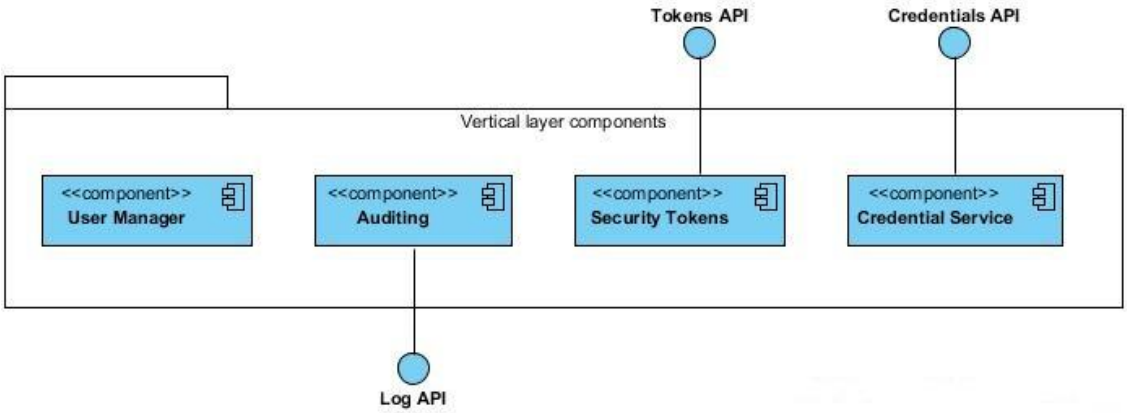

Figure 9: Vertical layer component diagram

Analogously to the SLA Platform architecture, the Vertical layer architecture has been updated compared to the previous version reported in Deliverable D1.1.2. Apart from updating the components' names (User Management, Audit, Tokens Management and Credential Management were the original names), we also removed some components (i.e., Notification Hub and Storage manager) as the offered functionalities have been allocated to other modules. Details about these updates are reported in deliverable D1.4.1.

With regards to the detailed design and implementation of the four Vertical layer components, they are split over different deliverables:

- The User Manager and the Auditing components are discussed in D1.4.1 (architecture) and D1.4.2 (implementation);
- The Credential Service and the Security Tokens components are described in D4.2.2 (architecture design) and in D4.4.2 (implementation).

As shown in Figure 9, the Vertical layer offers three APIs: the Log API, exposed by the Auditing component, which has already been detailed in Deliverable D1.3, the Credentials API and the Tokens API, which are discussed in Deliverable D4.4.2.

4.2.3. Enabling Platform module

The objective of the SPECS Enabling module is to provide a deployment and execution environment for the other SPECS modules.
It consists of two sub-modules, namely the SPECS Enabling Platform, which is a software stack that enables the SPECS platform deployment and its management at infrastructure level, and the SPECS Testbed supporting infrastructure, which represents the execution environment for the SPECS Enabling Platform.

A preliminary design of the Enabling Platform architecture has been reported in Deliverable D1.6.1 (released at month 18) and is still a work in progress, to complete at month 30 with deliverable D1.6.2.

In the following, we present the latest updates related to the definition of the general architecture of the Enabling Platform and to the design of its main internal components.
The current SPECS Enabling Platform architecture comprises the following four components (illustrated in Figure 10)[4]:

- **Launcher:** Provides a collection of services responsible for the Enabling Platform infrastructure deployment and resource registration;
- **Custom OS:** Referred to as **mOS** (multi-purpose operating system), this is a custom, lightweight operating system enriched with a set of remote runtime operations services that are able to setup, on demand, the environment for the hosted components using Chef technology[5];
- **Core Repository:** Represents a collection of Chef recipes that define the deployment and runtime processes of the core components;
- **Mechanisms Repository:** Represents a collection of Chef recipes that define the deployment and runtime processes of the security mechanism components.

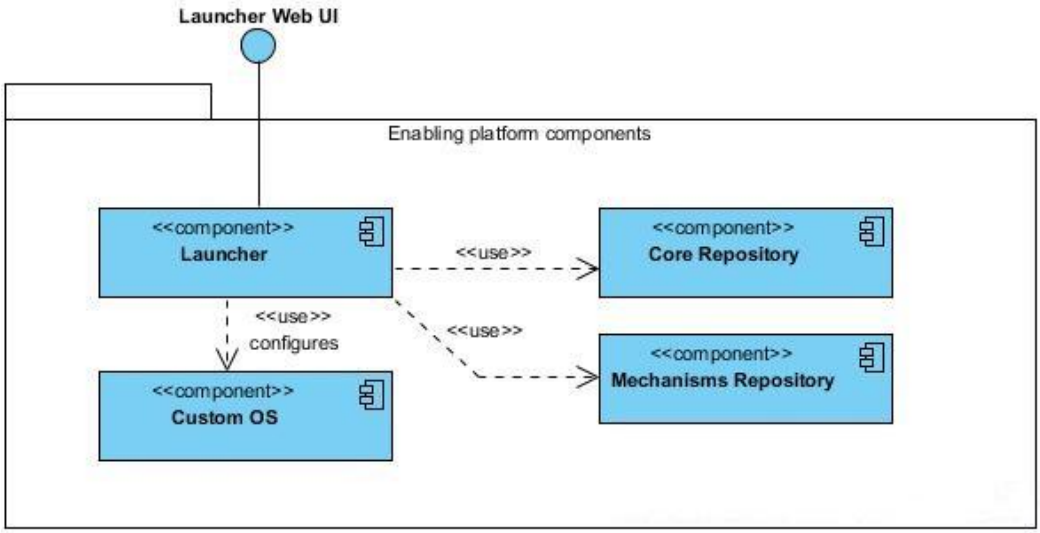

Figure 10: Enabling Platform component diagram

[4] Note that the previous solution proposed in Deliverable D1.6.1 identified three main architecture components (i.e., Custom OS, OS Bootstrapper and Cluster Manager),
[5] Chef , Automation for Web-Scale IT, http://www.chef.io

The functionalities provided in the previous version by the OS Bootstrapper (i.e., OS customization, resource discovery and management, resource configuration) and by the Cluster Manager (i.e., allocation of cloud resources acquired by external providers and deployment of services on such resources) have been redistributed and new ad-hoc components have been introduced for the management of Chef recipes.
The resulting refined architecture is reported in Figure 11, which outlines the internal design of the Launcher and the Custom OS components.
The Launcher is responsible for the initial deployment of the SPECS Enabling Platform to support the core services that will enable the platform functionalities. The Launcher consists of four sub-components:

- **User Interface**: A web user interface through which the End-user can configure and start-up the SPECS platform and the resources that will host the platform;
- **Cluster Manager**: A service responsible for the deployment and start-up of the SPECS platform; the configuration received from the User Interface component is implemented in three steps: (1) the requested cloud resources are acquired from the targeted cloud provider; (2) the Chef infrastructure is set up and the core repositories are imported into the Chef database; (3) the specific recipes are applied on the resources in order to bootstrap the SPECS platform core components;
- **Cloud Resource Allocator**: A service that is able to reserve and bootstrap cloud resources (running the SPECS Custom OS) using the specialized APIs of cloud providers;
- **Service discovery**: A distributed system able to automatically register and discover resources; it is used for the Chef infrastructure unattended setup;

The Custom OS is the official operating system that runs on the cloud resources. As specified in D1.6.1, it is based on OpenSUSE 13.x (custom built distribution) enriched with a set of services used for the deployment and setup of a Chef infrastructure. Such services are provided by the following three components:

- **Node Bootstrapper***:* Invoked by the Cluster Manager to deploy the required components for the Chef infrastructure;
- **Service discovery***:* A distributed system used to register automatically the resources enabled on the working instance or to discover other required resources distributed across the cluster nodes;
- **Chef component***:* Can be either a Chef server, the orchestrator of the Chef infrastructure, or a Chef client, the local service in charge of the deployment of the recipes.

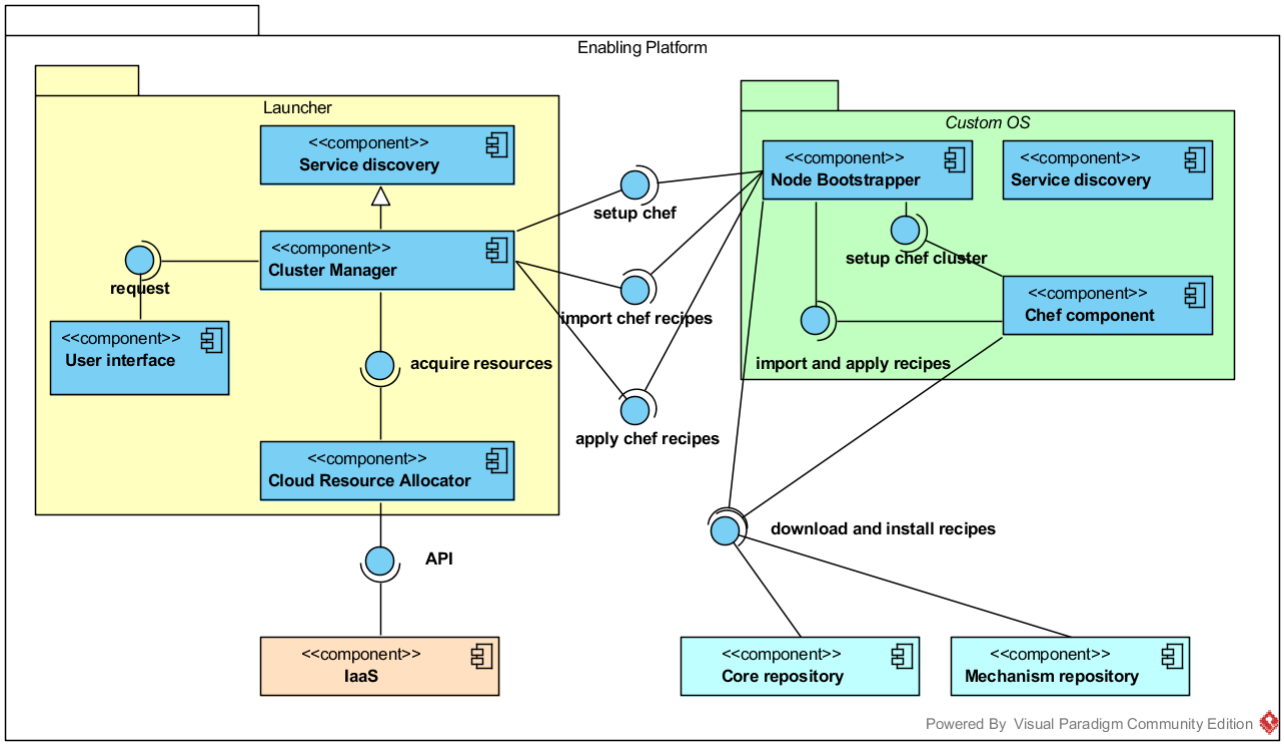

Figure 11: SPECS Enabling Platform detailed architecture

The SPECS Enabling Platform is hosted on the SPECS Testbed supporting infrastructure. The SPECS Testbed, whose architecture is depicted in Figure 12, is a private cloud provider powered by HP Helion[6] (formerly called Eucalyptus Cloud) cloud stack. It is hosted by the partner IeAT on its own private hardware resources. The architecture of the SPECS Testbed is presented in D1.6.1 and will be refined in D1.6.2 at M30.

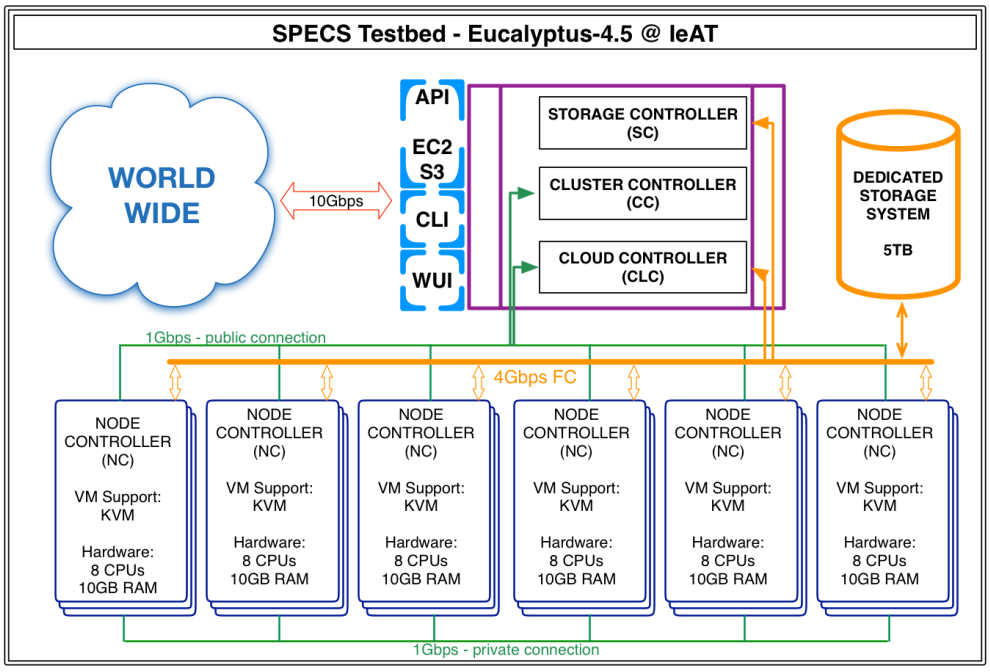

Figure 12: SPECS Testbed architecture

[6]http://www8.hp.com/us/en/cloud/helion-overview.html

4.2.4. Negotiation module

The Negotiation module manages the SLA negotiation and renegotiation phases. It manages the creation of the SLA by analysing the End-user requirements gathered through the SPECS application until its signature. The details of the negotiation module are reported in D2.2.2 and in the set of deliverables associated with task 2.3. The Negotiation module carries out the following operations:

- **Analysis of End-user's security requirements to generate the SLA**: The Negotiation module interacts with the SLA Platform to provide End-users with the list of available services (based on available templates). Through the SPECS application, the End-users will then specify their requirements for capabilities, controls and metrics.
- **Build SLA offers:** Once the End-user's security requirements are known, the Negotiation module triggers the generation of a set of supply chains, by the Enforcement module. With the list of valid (i.e., feasible) supply chains, the Negotiation module generates a set of SLA offers.
- **Rank of SLA offers:** In order to leverage the decision support capabilities of the SPECS framework, the Negotiation module ranks the SLA offers according to End-users' security requirements. SLA offers combine SPECS services and external CSPs' services. Hence End-users are provided with a very useful tool to know which is the SLA that best matches their requirements is.
- **Manage renegotiation**: SPECS considers two types of renegotiation, depending on the actor that triggers it. On the one hand, the CSP-triggered renegotiation occurs when an alert or a violation has been detected by the Enforcement module. These types of events entail a change in the SLA commitments, and a negotiation of new SLOs is required. On the other hand, the End-user-triggered renegotiation occurs when an End-users wants some change in the service (such as a new capability or a new value for an SLO); these changes invalidate the previous SLA and entail the renegotiation of a new one.

The Negotiation module (Figure 13) consists of three components that carry out the aforementioned activities, namely:

- **SLO Manager:** Manages the creation of SLA templates and it is also responsible for building SLA offers from the supply chains identified by the Enforcement module.
- **Supply Chain Manager:** Triggers the creation of supply chains in the Enforcement module (as described in D4.3.2), by providing it with the information on security requirements expressed by End-users (capabilities, controls and metrics).
- **Security Reasoner:** Responsible for the ranking of SLA offers, performed by applying dedicated assessment algorithms that allow to evaluate qualitatively and quantitatively the level of security associated to different SLAs, in order to compare them side-by-side.

Note that the Negotiation module architecture has not changed with respect to the previous version reported in deliverable D1.1.2. At the current state, only the SLO Manager and the Security Reasoner components have been implemented, and their description is reported in deliverable D2.3.1.
The SLO Manager component exposes the *Negotiation API*, whose complete documentation is reported in deliverable D1.3, while other components only offer internal interfaces.

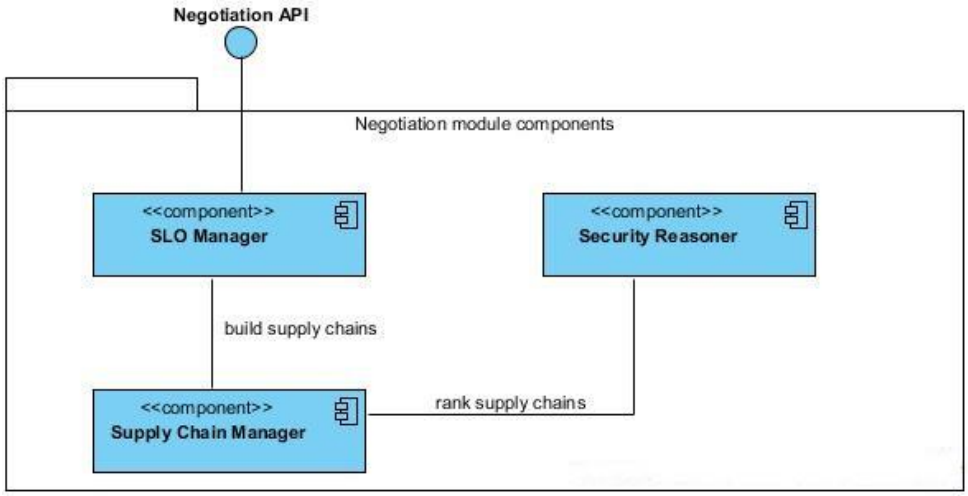

Figure 13: Negotiation module component diagram

4.2.5. Enforcement module

The Enforcement module orchestrates the actions needed in order to implement a signed SLA and ensure its fulfilment. This means that for each signed SLA the Enforcement module carries out the following activities:

- **Plans SLA implementation**: Translates End-user's security requirements specified in a signed SLA into a set of acquisition, allocation, and configuration parameters. During the negotiation process such a set of parameters is presented in the form of a supply chain. After an SLA has been signed, the Enforcement module builds an implementation plan according to the associated supply chain.
- **Implements signed SLA**: Provisions the resources and applies the security level specified in the signed SLA over the acquired resources. Acquisition and configuration of resources is conducted in accordance to the associated implementation plan.
- **Diagnoses monitoring events**: Analyses, classifies, and prioritizes all detected monitoring events that could represent a current or (possibly) a future SLA violation.
- **Remediates SLA alerts and violations**: Dynamically reconfigures resources after a detection of SLA alerts and violations according to the affected SLA and to the results of the diagnosis phase.

The entire enforcement process is orchestrated by a set of four core Enforcement components (depicted in Figure 14):

- **Planning**: Builds supply chains and implementation plans.
- **Implementation**: Executes implementation plans provided by the Planning and implements the remediation plans provided by the RDS.
- **Diagnosis**: Performs the analysis of possible alerts and violations.
- **Remediation Decision System (RDS)**: Builds remediation plans to recover from SLA alerts and violations analysed by the Diagnosis.

It should be noted that, as shown in Figure 14, the Implementation component explicitly includes two components, namely the **Broker** and **Chef Server**.

The role of the Chef Server has been discussed in D4.2.2, which showed the need for a tool able to perform the automatic deployment and configuration of software components for the security enforcement. The implementation plan consists of a set of *recipes*, stored in the Chef Server and used to install and configure security mechanisms required by the SLA to be

implemented. For this reason, the Chef Server has been explicitly included among Enforcement core components in this document.

Moreover, it should be noted note that in the previous version of the architecture design, the Broker (also known as Secure Provisioning) was considered as one of the available security mechanisms. Since the Broker does not actually offer any tangible security capability but is only used to acquire the resources needed for the implementation plan, we decided to include it among the Enforcement core components.

Further details on the internal architecture of the Enforcement core components are reported in deliverable D4.2.2, while deliverable D4.3.2 contains the information related to their current implementation.

The Enforcement components expose the *Enforcement API,* described in deliverable D1.3, enabling the generation of supply chains, the implementation of plans and the analysis of monitoring events.

The security services offered in SPECS during the negotiation process are enforced and monitored by the following set of security mechanisms (see Figure 14):

- **Secure Web Server (WebPool)**: Provisions web servers and offers security assurances through redundancy and diversity. Each acquired web server can be provisioned with a configurable amount of replicas (providing redundancy) of different types (assuring diversity).
- **Transport Layer Security (TLS)**: Offers different configurations of the TLS protocol.
- **Software Vulnerability Assessment (SVA)**: Provides configurable software vulnerability scanners and enables periodic updates of the list of published vulnerabilities, periodic vulnerability scans, periodic automatic checks for updates and upgrades of vulnerable libraries installed on acquired resources.
- **Database and Backup (DBB)**: Offers secure storage with the capability of detecting violations related to consistency among writes and reads of the stored data. Additionally, storage resources are extended with backups.
- **End-to-end Encryption (E2EE)**: Extends DBB mechanisms with providing client-side encryption of the stored data.
- **Authentication, Authorization, Accounting (AAA)**: Provides federated identity and access features over resources and services of different applications.
- **Denial of Service (DoS) Mitigation**: Offers detection, classification, and mitigation of DoS attacks.

These mechanisms are able not only to enforce suitable security capabilities, but also to monitor specific security metrics. To achieve this, each mechanism comes with a dedicated *adapter*, able to communicate with the Monitoring module and to send monitoring-related data to it. Additional security mechanisms can be developed and offered by SPECS.
Further details on the design of such negotiable security mechanisms are available in D4.2.2, while implementation and usage details are provided in D4.3.2. Compared to the set of security mechanisms discussed in D4.2.2, we added the *Database and Backup (DBB)* mechanism, which is fully described in D4.3.2. The AAA and DoS Mitigation mechanisms have not been finalized yet, and the related details will be available in D4.3.3.

In conclusion, it is worth mentioning that other components have been developed under the Enforcement umbrella, and are thus included in the diagram of Figure 14. They belong to the Vertical layer discussed in Section 4.2.2 and are the Auditing, Security Tokens and Credential Service components. The Security Tokens and Credential Service components are further discussed in D4.4.2, while the Auditing component is detailed in D1.4.1 and D1.4.2.

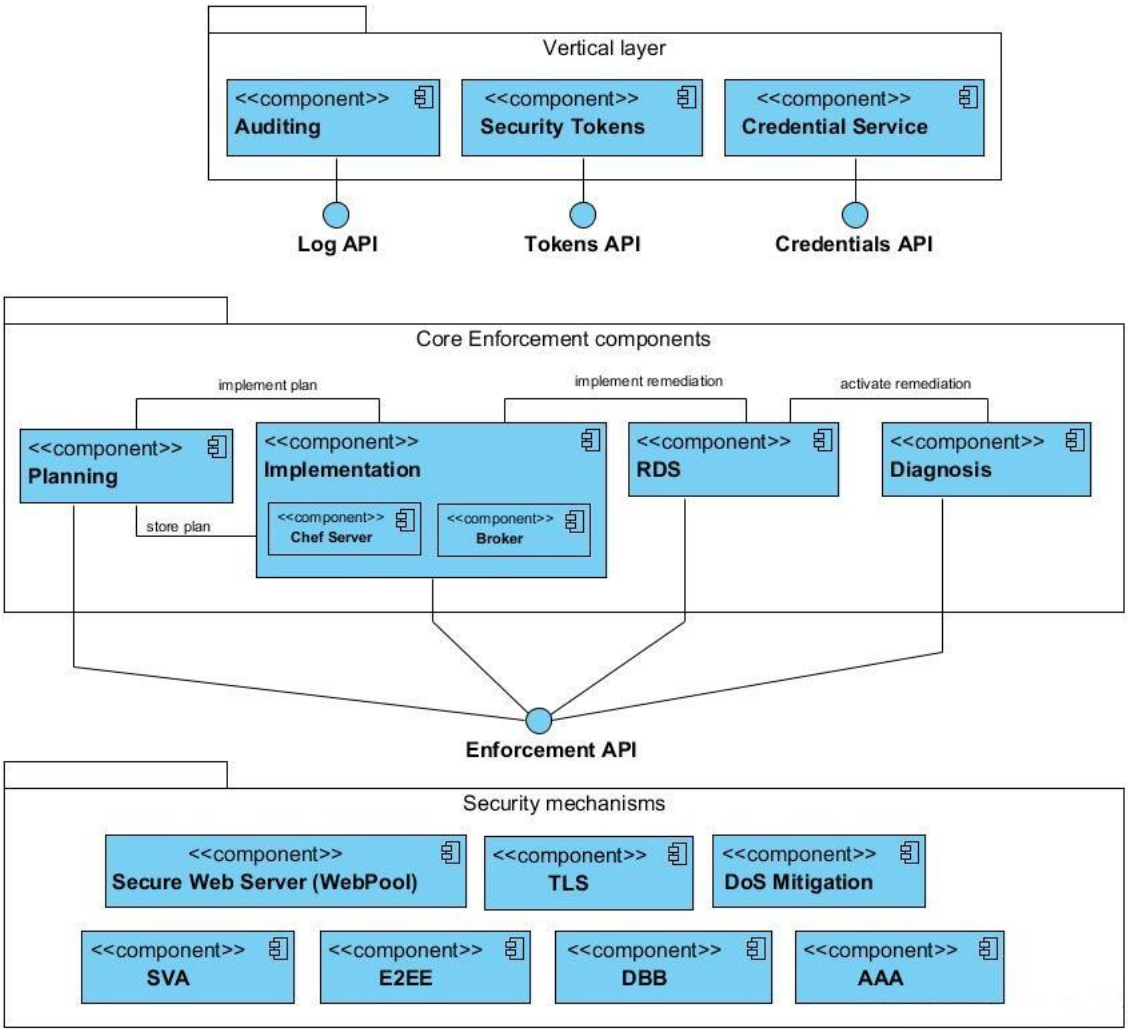

Figure 14: Enforcement module component diagram

4.2.6. Monitoring module

The monitoring module orchestrates the entire process of service monitoring by gathering raw information from the target services, translating it into events and, finally, into alerts or violations that need to be remediated.

The monitoring module is split into two parts (as shown in Figure 15):

- **Monitoring Core services:** In charge of events processing (routing, aggregation, archival, filtering) and alerts or violation notification in case of anomalies;
- **Monitoring Systems on the Target services:** Represents the components that are running on the target services and are in charge of collecting raw data and publishing events to the core services.

The **monitoring core services** part consists of several components that offer the basic monitoring functionalities:

- **Event Hub:** A message router that receives events collected from various monitoring adapters and route them towards to the Event Aggregator, Event Archiver and Monitoring Policy Filter;
- **Event Aggregator:** Consumes events from the Event Hub and according to the configured aggregation rules, aggregates the events and send them back to Event Hub for further utilisation;
- **Event Archiver:** Stores all the events that are exchanged through the Event Hub for later use;
- **Monitoring Policy Filter (Monipoli):** In charge of events filtering to extract only those events that potentially could be alerts or violations;
- **SLO Metrics Exporter:** Forwards notifications (alerts or violations) to the Enforcement core services;

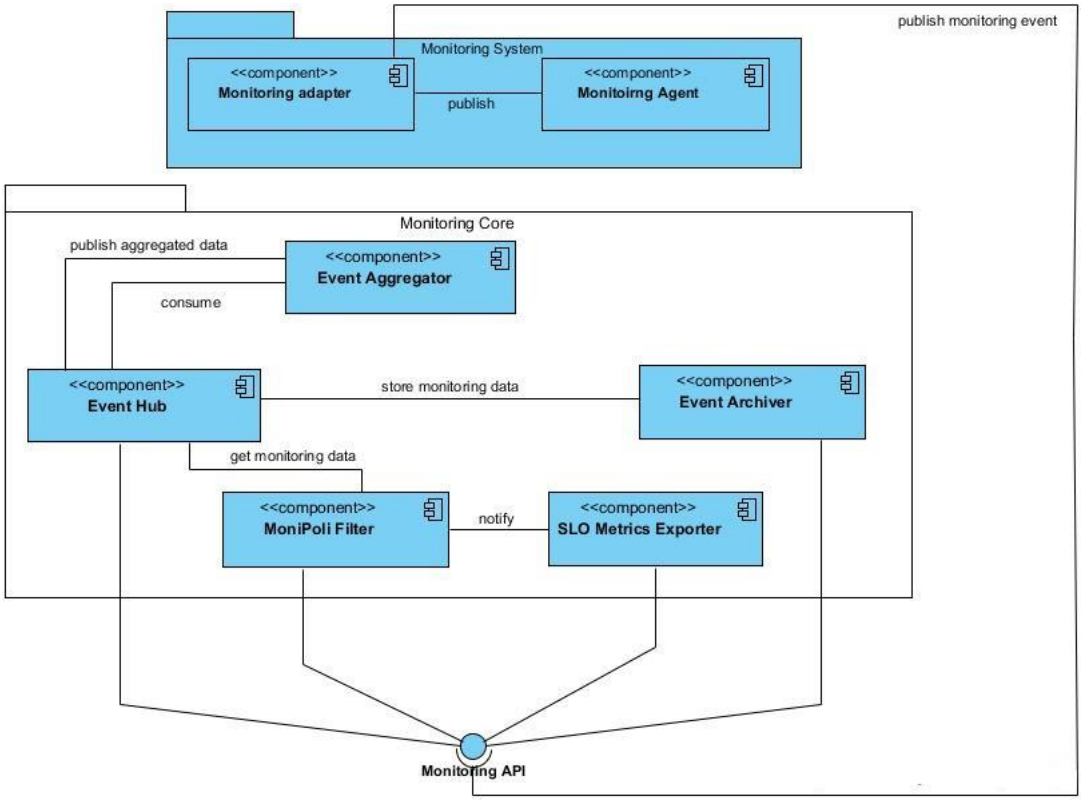

Figure 15: Monitoring module component diagram

It should be noted that the architecture of the Monitoring core has not changed as compared to the previous version, reported in D3.3. The design details related to the monitoring core components are included in D3.3, while D3.4.1 reports some updates regarding the implementation. The Monitoring module exposes the Monitoring API, fully described in Deliverable D1.3.

As anticipated in the previous section, the events processed by the monitoring core services are collected through the security mechanisms **adapters**. Monitoring adapters can be integrated into the mechanism and/or built on top of existing monitoring tools (such as OSSEC, OpenVAS and NMAP). They gather data from **monitoring agents**, which have the ability to search and extract raw information for the target services by scavenging the log files or other monitoring data. Monitoring adapters transform such raw monitoring information into SPECS events, which are successively published to the Event Hub.

Both components may be deployed on the Target Service, as shown in Figure 15, or may be deployed on other machines, depending on the rules posed by the External CSP providing the Target Services' resources.

4.2.7. Overall architecture

The overall architecture of the SPECS platform is depicted in Figure 16, which reports the platform modules and their main internal components. The figure shows also the APIs offered by the modules, namely the SLA API, the Services API, the Interoperability API, the Negotiation API, the Enforcement API, the Log API and the Monitoring API, which have been discussed in Deliverable D1.3. In addition to these, the figure also reports the Tokens API and Credentials API offered by the Vertical layer (described in D4.4.2), the Launcher Web UI offered by the Enabling Platform (discussed in D1.6.1) and the Metric Catalogue API (illustrated in D1.4.1).
Finally, the diagram in Figure 16 reports the main connections among components. It should be pointed out that, for clarity's sake, we do not report explicitly all connections (e.g., we do not report explicitly that the Interoperability API and the Log API are used by all components). Moreover, the connections with the SLA API and the Services API, which are the ones used by most components, are highlighted using different colours.

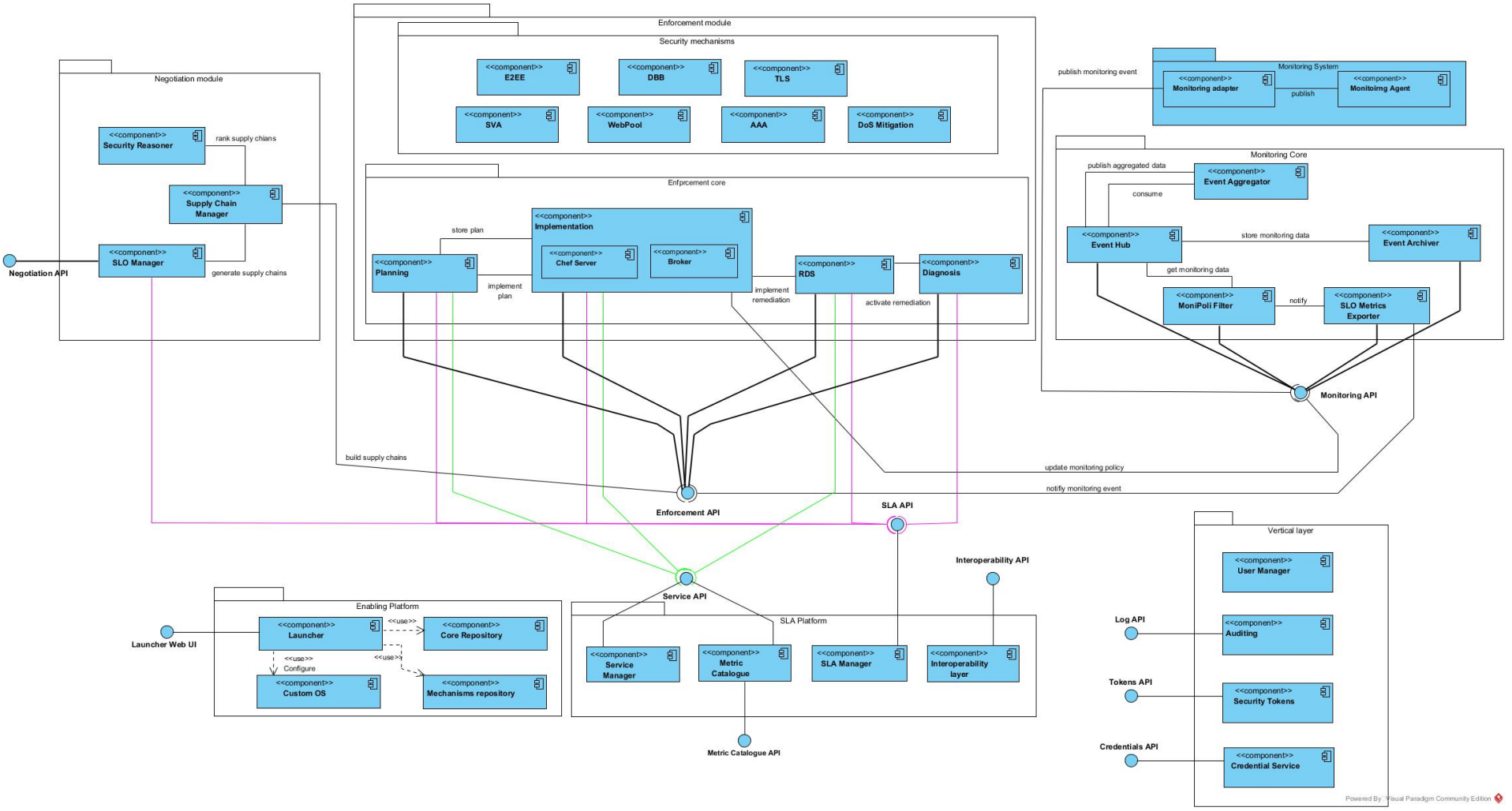

Figure 16: The full SPECS platform architecture

5. Using the SPECS framework

As already discussed (cf. D1.2 and D5.1.2) we introduced three different Interaction Models, which describe how the SPECS framework can be used to offer cloud services and which kind of responsibilities are involved.
Within the introduced Interaction Models, we are interested in specifying the role played by the SPECS owner, which may or may not take on the responsibility of signing the SLA related to the target service invocation. If yes, it acts as a *CSP*, otherwise it acts as a *Cloud Service partNer*(CSN).

In other words, an Interaction Model outlines (1) how the SPECS owner deploys the SPECS framework and (2) the SPECS framework usage options. As already discussed in the Deliverable D1.2, Interaction Models are important from a security and SLA point of view, as they imply different security issues in the whole supply chain and different responsibilities as far as the SLAs are concerned. In the design activities, we have reached the conclusion that these different scenarios affect the SLA signatures and resources involved, but do not affect the internal architecture of the SPECS framework. This also implies that the SPECS framework can be easily used and adapted to develop many different applications that can be guaranteed by Security SLAs.
In each Interaction Model, we put in evidence the *SPECS role*, i.e., the responsibility that the SPECS owner assumes over the SLA agreed with the End-user. The SPECS owner can be either the CSP (i.e., the one that provides the services and grants them, possibly paying penalties for SLA violations), or simply a Partner (who just helps the customer and the provider to find an agreement, but does not assume any responsibility).

In this section we present four different SPECS applications that correspond to four different business cases being developed by the SPECS consortium to build a close-to-market solution portfolio. The four business cases summarized in this section are related to the *SPECS Solutions Portfolio,* which is a set of software products being built by CERICT (Secure Web Container), CSA (SPECS-enhanced STAR Watch Premium), EMC (SPECS-enhanced ViPR) and XLAB (End-to-End Encryption), they exploit the three different Interaction Models. The Solutions Portfolio can be easily accessed from the SPECS website[7]. Final details and examples on how to develop new SPECS applications will be reported in deliverables associated to work package 5, namely D5.1.3, D5.2, D5.3 and D5.4.

5.1. Development with SPECS Framework

As anticipated in Section 3.2 and in Deliverable D5.1.1/D5.1.2, the SPECS framework main users are CSPs, which use the SPECS platform, services and applications to offer cloud services to their customers according to SLA life cycle, and Developers.

A developer can use the SPECS framework to develop new SPECS applications offering new services (according to the guide in deliverable D5.1.3). However, since each SPECS module works independently of the remainder of the platform, a developer could exploit its services accessing it directly. The deliverable D1.3 illustrates the standard flow of execution of the SPECS platform, also presenting the REST API used by each of the modules.
An example of application that uses a subset of the APIs offered by the platform modules is available in Deliverable D5.1.3 (the Metric catalogue application).

[7] Please refer to http://www.specs-project.eu/

The SPECS Framework is open source, and available through the *specs-team* account on the BitBucket open repository: https.//bitbucket.org/specs-team.
The framework, as illustrated in section 0, is made of several components; the specs-team account includes a repository for each component.
The repositories are named accordingly to the role that they have in the framework (*core* for the components of the platform, *mechanism* for the mechanisms, *app* for the SPECS applications, *utility* for the others) and the module to which they belong (*enabling_platform*, *slaplatform*, *negotiation*, *monitoring*, *enforcement*).
For example, the SLA manager, dealt with in Section 4.2.1, is available in the repository `specs-core-slaplatform-sla_manager`.
Each repository has a readme that briefly summarizes the component usage.
The deliverable D5.1.3 offers a guide for the development of new applications using the SPECS framework; the deliverables associated to the components offer the software documentation and the component description.
The SPECS web site (www.specs-project.eu) has dedicated sections to link the SPECS framework repositories and the user guide, so as to help developers in accessing the SPECS resources.
Moreover, the SPECS framework includes a set of facilities for the developers registered as members in the *specs-team* account: the continuous integration solution introduced in Deliverable D4.5.2 (based on bamboo, and available at https://bamboo.services.ieat.ro) and the code quality tool based on sonar (https://sonar.services.ieat.ro/).
The SPECS Testbed (see Section 4.2.3) hosts the Enabling Platform and helps in creating new instances of the SPECS Platform over the project resources (as described in D1.4.2).

5.2. Secure Web Container

This business case is based on a solution (offered by CERICT) targeting web developers looking to acquire a web container[8] for their applications, which fulfils a specific set of security requirements. The main challenge this solution is trying to face is to offer an easy-to-understand way to specify security capabilities during the negotiation of a Security SLA, to deploy in an automatic way the security features in the acquired web containers, to have grants related to the security capabilities and SLOs that have been negotiated. This application is described in details in D5.1.2 and D5.1.3 and provides an example of Interaction model 1, where the SPECS framework is hosted by a public or private CSP and the target service is hosted on some external CSP. The SPECS owner signs the Security SLA that includes all the security capabilities needed to fulfil End-user security requirements.
The main features associated with this business case are shown in Table 2.

Table 2. Secure Web Container features

Interaction Model	Advantages	Leveraged SPECS components
IM1	• A single interface to select among different offerings provided by different CSPs. • Mechanisms to enable web developers to specify the required security capabilities on the target web container.	• SLA Manager, Service Manager, SLO Manager, Supply Chain Manager • Planning, Implementation, Diagnosis and Remediation, SVA, WebPool, IDS, TLS

[8] The web container is represented by one or more Virtual Machines (VMs) provided by one or more IaaS CSPs.

Interaction Model	Advantages	Leveraged SPECS components
	• Automatic configuration mechanisms to enforce and monitor the requested security controls. • Automatic remediation of some detected security alerts/violations that may occur, associated to the SLA of the web container.	• Monitoring core

5.3. STAR Watch Premium

Another interesting example of SPECS usage is given by the Premium version of CSA STAR Watch[9] product that leverages specific components of the SPECS framework. STAR Watch delivers - in a database/machine readable format - the content of CSA's succinct yet comprehensive list of cloud-centric control objectives defined in the Cloud Controls Matrix (CCM) and the corresponding set of control assertion questions in the Consensus Assessments Initiative Questionnaire (CAIQ).

In particular, the STAR Watch Premium will leverage the platform security reasoning techniques to offer the ability to compare different CSPs by assessing their CCM responses with respect to the end user's security requirements. Table 3 presents further details related to STAR Watch Premium.

Table 3. STAR Watch Premium features

Interaction Model	Advantages	Leveraged SPECS components
IM1	• Prospective and current cloud customers can have a better level of transparency related to the CSPs delivering services to their organization. • Enables side-by-side comparison of CSPs based on a baseline set of end-user requirements. • Suitable for non-security expert users.	• The Premium version of STAR Watch will integrate SPECS security reasoner techniques. • CSP comparison features based on SPECS' Security SLA hierarchy.

5.4. End-to-end Encryption

An interesting example of Interaction Model 3 is provided by the XLAB business case, leveraging SPECS components to offer customers a secure storage solution allowing them to encrypt data in the cloud and not only detect but also prove violations related to modification and loss of stored data. The secure storage capability is implemented by the E2EE solution developed within the SPECS enforcement security mechanisms, furthermore diagnosis and remediation services will be used to detect possible violations to data stored, these are described in details in D4.3.2 and will be further exploited in D5.3. In Table 4 the main features of this application are reported.

Table 4. End-to-end Encryption features

Interaction Model	Advantages	Leveraged SPECS components
IM3	• Deploys a client-side encryption functionality enforcing *confidentiality* and *integrity*. • Provides detection and proof of violations related cloud storage. • Open source.	• This product leverages SPECS' enforcement mechanisms

[9] Please refer to https://cloudsecurityalliance.org/star/#_watch

Interaction Model	Advantages	Leveraged SPECS components
	• Equipped with a remediation functionality including an automated incident response mechanism.	

5.5. SPECS-enhanced ViPR

This business case developed by EMC enhances the security features offered by EMC's ViPR[10] storage controller, in particular when an end-user wishes to acquire storage resources via a CSP.

This application will be detailed in D5.2 and provides an example of Interaction Model 2, where the SPECS framework and the target service is hosted by the same private CSP. The SPECS owner signs the Security SLA that includes all the security capabilities needed to fulfil End-user security requirements and he also has the full control over the target service, ViPR in this case, and can deploy any desired security capability and monitoring systems.

Some specific features are mentioned in the following table.

Table 5. SPECS-enhanced ViPR features

Interaction Model	Advantages	Leveraged SPECS components
IM2	• SPECS adds a layer of control and intelligence on top of the ViPR interface while providing a user friendly interface to the customer. • Using the SPECS web interface, the customer itself can specify their storage requirements and sign an SLA with SPECS confirming the enforcement of those requirements. • SPECS automatically configures the storage and makes it available to the customer without the need for intervention from the admin staff.	• SLA Manager, Service Manager. • Brokering. • Monitoring.

[10] Please refer to https://www.emc.com/vipr

6. Security assessment and compliance to regulation

In this section, we report some final considerations related to the adoption of the SPECS solution. We first analyse the issues related to the assessment and certification of the services provided through SPECS, which is of particular interest for SPECS customers. Then, we discuss some compliance and liability aspects that may arise when using SPECS, due to existing regulations.

The SPECS architecture aims to provide security as a service (SaaS) with built-in monitoring capabilities. As such, the SPECS architecture should fulfil two objectives:

- Objective 1: Bring additional security features to end-users;
- Objective 2: Do not introduce security weaknesses that did not previously exist.

Today, one of the central ways to assess the security of a service is to verify its compliance with information security management standards. The most well-known standard for this purpose is ISO/IEC 27001 [11]. It offers a generic method for assessing information systems, but it is based on control objectives that are not designed for cloud specific needs. As an alternative, Cloud Security Alliance (a partner in the SPECS project) has designed the CCM (Cloud Control Matrix)[4] as a control framework that targets cloud specific needs. In September 2013, CSA teamed with the BSI to launch the CSA Start Certification scheme, actually combining the evaluation methodology developed in the context of ISO/IEC 27001 with the CCM as a control framework. Both ISO 27001 and STAR provide an assessment of an information system that goes far beyond technical points by encompassing also documentation quality, processes, compliance, governance, human-resources, etc. In other words, security is assessed in the context of an organization operating a "live" information system.

The SPECS architecture itself is therefore not the right subject for such a certification. However, if an organization decides to implement the SPECS architecture in order to offer a SaaS Brokerage service to real customers, then the issue of certification will become fully relevant on two fronts:

1) For the broker**,** as an organization offering SPECS enabled cloud-services.
2) For the customers who will use SPECS as part of an information system.

The broker will benefit from a certification in order to demonstrate that the SPECS service itself is secure (Objective 2). Such a certification will require an assessment of many elements that are out of scope of the architecture described in this documents, including physical security, contracts, compliance with local regulation that apply in relation with the geographical establishment of the broker. Nevertheless, the SPECS architecture provides a few artifacts in support of any form of certification: documentation of the SPECS software, description of authentication mechanism and software security quality controls (see deliverable D4.5.2).

For the customer, the choice of SPECS mechanisms should act as a facilitator for the certification of a cloud service, in order to gain trust in the security of its information system (Objective 1). We note however that the use of a SPECS mechanism alone is not a silver bullet: for example, if the customer uses the E2EE (encryption) mechanism, but it takes no precaution in order to secure the cryptographic keys that are used, the security of the system is at risk. The security of an information system should be evaluated as a whole, including both mechanisms provided by SPECS and measures implemented by the customer.

To get a feel of the potential advantage provided by SPECS to the customer, we will consider the set of controls identified in the CSA CCM along with the CSA CAIQ (a self-assessment questionnaire which details the CCM) [12]. Each control in the CCM is identified by a code and a title (e.g. "*IPY-01 – Interoperability & Portability / APIs*"). By scanning through the CCM and the CAIQ, we can identify controls for which SPECS mechanisms have the potential to act as "facilitators", in combination with the customer's own security measures, as reported in the following table:

- The secure provisioning (Broker):
  - *Facilitates BCR-01 - Business Continuity Management & Operational Resilience / Business Continuity Planning.*
  - *Facilitates IPY-01 Interoperability & Portability / APIs*
- The secure web server (WebPool):
  - *Facilitates BCR-11 - Business Continuity Management & Operational Resilience / Retention policy*
- Transport Layer Security:
  - *DSI-03 - Data Security & Information Lifecycle Management eCommerce Transactions.*
  - *Facilitates IVS 11 - Infrastructure & Virtualization Security / VMM Security - Hypervisor Hardening*
- Software Vulnerability Assessment:
  - *Facilitates AAC-02 - Audit Assurance & Compliance Independent Audits*
  - *Facilitates IVS-05 - Infrastructure & Virtualization Security / Management - Vulnerability Management*
  - *Facilitates TVM 03 - Threat and Vulnerability Management / Vulnerability & Patch Management*
- Database and Backup:
  - *Facilitates AIS-03 - Application & Interface Security / Data Integrity*
  - *Facilitates BCR-11 - Business Continuity Management & Operational Resilience / Retention policy*
- End-2-end Encryption:
  - *Facilitates AAC-03 - Audit Assurance & Compliance / Information System Regulatory Mapping*
  - *Facilitates EKM-2 - Encryption & Key Management / Key Generation*
  - *Facilitates EKM-3 - Encryption & Key Management / Storage and Access*
- Authentication, Authorization and Accounting:
  - *Facilitates IAM-12 - Identity & Access Management / User ID Credentials*
  - *Facilitates IVS-09 - Infrastructure & Virtualization Security / Segmentation*
- Denial of service mitigation:
  - *Facilitates IVS-13 - Infrastructure & Virtualization Security / Network Architecture*

In addition, SPECS as architecture with specific monitoring capabilities has the potential to act as a facilitator for the following CCM control:

- *BCR 09 "Business Continuity Management & Operational Resilience Impact Analysis".*

It is worth noting that the SPECS owner is responsible for offering SPECS services and he/she is the juridical subject signing the contract (SLA). Indeed, cloud regulation is a subject of much debate among government bodies and many potential risks may arise when using SPECS, due to existing regulations.

Depending on the adopted interaction model, SPECS applications may need to store and manage information such as user accounts, passwords, personal information and credentials used to buy resources from different external providers. In these cases, proper security policies must be actuated in order to take care of the consequent issues related to privacy, security and existing regulations. Depending on the local regulation or on the application environment in which SPECS operates, the requirements may drastically change.
Furthermore, SLA monitoring data are archived by the SPECS host (whether public cloud or hosted on organisations private cloud) for future auditing, reporting or investigative purposes. It is important that this process is transparent to the customer (in particular for cases where the customer terminates the contract with the SPECS host) as they may have some data privacy concerns about the information contained in the records.

SPECS cannot cover all the situations and, in any case, particular needs may arise in specific scenarios. The problem will be evaluated by the SPECS owner in accordance with local regulation and hosting CSP policies. Nevertheless, to support the SPECS owner, additional security mechanisms can be developed with SPECS for the management of the above discussed situations, which are easily integrated within the SPECS framework thanks to its modularity. Related to this, it is worth noticing that SPECS already offers a mechanism to securely manage credentials (see Credential Service, discussed in Deliverables D4.2.2 and D4.4.2).

Additionally, the protection of personal data (Personally Identifiable Information - PII) is a subject of much debate among government bodies, too. In general, new personal data protection laws are evolving to meet the new challenges faced by cloud models for data storage, distribution and processing. The objective of these new laws is to provide individuals with control and visibility over their personal data. It is important to highlight that the CSPs that host the SPECS platform should take into consideration such regulations when using the SPECS solution.

Furthermore, there is a particular concern regarding the adherence to data geo-location regulations, too. The transfer of data among regions with different legislation and regulations on data protection is a critical point. Many organisations are unwilling to have their data subject to multiple sets of laws and are concerned about the possibility of having their data location changed by CSPs. In providing assurance about data geo-location via an SLA, it is the responsibility of the SPECS owner to ensure that only data centres and storage resources that comply with the geo-location requirement are adopted. Failures in enforcing data geo-location restrictions could potentially put customers' data at risk. In addition to geo-location, there are other important compliance features that may be provided through SPECS so that customers can be assured that their services will perform optimally without significant risk of breaches or degradation due to attacks (e.g. DoS).

In conclusion, the SPECS platform provides potential mechanisms for securing cloud services, but the security level of a SPECS-enabled service and its compliance with regulation can only be assessed in a real-world instantiation of the architecture.

7. Conclusions

In this Deliverable, we presented the final architecture of the SPECS platform, derived from the refinement of the design activity presented in the previous version (D1.1.2). The final architecture is the result of a joint work conducted in several tasks during the last year related to a close-to-market analysis that has involved End-users in the validation of requirements gathered in the first year and from feedback received by the prototype implementation of preliminary designed solutions.

In particular, in the following table, already presented in the document, we report the links to all deliverables, available or in progress, where all the details on the design and implementation, are presented and discussed. The reader is referred to those documents to obtain the latest updates and details.

Module	**Component**	**Architecture Design Details**	**Implementation Details**
SLA Platform	SLA Manager	D1.4.1	D1.4.2
	Service Manager	D1.4.1	D1.4.2
	Metric Catalogue	D1.4.1	D1.4.2
	Interoperability layer	D1.4.1	D1.4.2
Vertical Layer	User Manager	D1.4.1	D1.4.2
	Auditing	D1.4.1	D1.4.2
	Credential Service	D4.2.2	D4.4.2
	Security Tokens	D4.2.2	D4.4.2
Enabling Platform	Launcher	D1.6.2	D1.1.3/D1.6.1 (D1.6.2)
	Custom OS	D1.6.1 (D1.6.2)	D1.6.1 (D1.6.2)
	Core Repository	D1.6.1 (D1.6.2)	D1.6.1 (D1.6.2)
	Mechanisms Repository	D1.6.1 (D1.6.2)	D1.6.1 (D1.6.2)
Negotiation Module	SLO Manager	D2.3.1 / (D2.3.2)	(D2.3.3)
	Supply Chain Manager	(D2.3.2)	(D2.3.3)
	Security Reasoner	D2.3.1 / (D2.3.2)	(D2.3.3)
Enforcement Module & Security Mechanisms	Planning	D1.1.2/D4.2.2	D4.3.2
	Implementation	D1.1.2/D4.2.2	D4.3.2
	Diagnosis	D1.1.2/D4.2.2	D4.3.2
	RDS	D1.1.2/D4.2.2	D4.3.2
	Broker	D4.2.2	D4.3.2
	WebPool	D4.2.2	D4.3.2
	TLS	D4.2.2	D4.3.2
	SVA	D4.2.2	D4.3.2
	DBB	D4.3.2	D4.3.2 / (D5.4)
	E2EE	D4.2.2	D4.3.2 / (D5.2)
	AAA	D4.2.2	(D4.3.3) / (D5.4)
	DoS	D4.2.2	(D4.3.3)
Monitoring Module	Event Hub	D3.3	D3.4.1
	Event Aggregator	D3.3	D3.4.1
	Event Archiver	D3.3	D3.4.1
	Monitoring Policy Filter	D3.3	D3.4.1
	SLO Metric Exporter	D3.3	D3.4.1

Interaction Protocols	(ALL) Module APIs	D1.3	D1.3

The final architecture including the SPECS main concepts, the modules and the technological choices is also result of a continuous observation of available solutions at the state of the art that we have been monitoring since the beginning of the project in order to always use the most suitable solutions. In addition to the architecture, we also presented four SPECS applications. Such applications outline the different interaction models related to the usage of SPECS and belong to the SPECS solution portfolio that is currently under development and will be fully available at M30.

Finally, we reported some considerations on the security assessment of the services offered through SPECS and on the compliance to regulation issues arising when using SPECS due to existing geographical regulations and laws.

At M24, a prototype of the SPECS framework is available; it can be downloaded from the SPECS repository at https://bitbucket.org/specs-team/profile/repositories, while a description of available applications is already reported on the website, in the Portfolio sub-menu at www.specs-project.eu. Among the applications in Portfolio, the Secure Web Container is already available and described in D5.1.2 and D5.1.3. It has been used as a validation application to cover a very high number of validation scenarios and demonstrate the feasibility of the proposed solutions.

8. References

[1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web services agreement specification (WS-Agreement)," in Global Grid Forum. The Global Grid Forum (GGF), 2004.
[2] International Organization for Standardization, "ISO/IEC NP 19086-1. Information Technology–Cloud computing–Service level agreement (SLA) framework and technology–Part 1: Overview and concepts," 2014
[3] National Institute of Standards and Technology, "NIST SP-800-53: Recommended Security Controls for Federal Information Systems,"2013.
[4] Cloud Security Alliance, "Cloud Control Matrix v3.0," https://cloudsecurityalliance.org/download/cloud-controls-matrix-v3/
[5] NIST, "NIST Special Publication 500-307 Draft: Cloud Computing Service Metrics Description," 2015.
[6] "Chef", 2014. [Online]. Available: http://www.getchef.com/
[7] https://aws.amazon.com/ec2/instance-types/
[8] http://www.apache.org/
[9] https://www.nginx.com/resources/wiki/
[10] https://oval.cisecurity.org/repository
[11] International Organization for Standardization, "ISO/IEC 27001:2013. Information technology - Security techniques - Information security management systems – Requirements, 2013.
[12] https://cloudsecurityalliance.org/group/consensus-assessments/