Secure Provisioning of Cloud Services
based on SLA Management

SPECS Project - Deliverable 1.5.1

Integration Scenario

Version no. 1.1
19 July 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.:	D1.5.1
Deliverable title:	Integration Scenario
Deliverable nature:	Report
Dissemination level:	Public
Contractual delivery:	19 July 2016
Actual delivery date:	19 July 2016
Author(s):	Jolanda Modic (XLAB), Miha Stopar (XLAB)
Contributors:	Massimiliano Rak (CeRICT)
Reviewers:	Ruben Trapero (TUDA), Marina Bregou (CSA)
Task contributing to the deliverable:	T1.5
Total number of pages:	89

Executive summary

The main focus of this deliverable is a set of integration scenarios that are defined according to the framework requirements and design, interaction protocols, and validation scenarios. The document is coupled with D1.5.2, which demonstrates integration scenarios from the implementation and testing perspective.

In particular, this document presents:

- An overview of the SPECS framework: We briefly present the architecture of the framework and summarize its functionalities.
- The integration overview: We present the integration plan, which is based on implementation plans associated to different modules of the SPECS framework, End-user's requirements, and validation scenarios. We also introduce the template used to describe each integration scenario, which simplifies the integration process.
- Integration scenarios: We split the set of integration scenarios into two groups. First we present scenarios associated to the integration of SPECS core modules/components, and then we describe scenarios related to the integration of the SPECS applications with security mechanisms.

Table of contents

Deliverable information 2
Executive summary 3
Table of contents 4
Index of figures 6
Index of tables 7
1. Introduction 8
2. Relationship with other deliverables 9
3. SPECS framework overview 10
3.1. The SPECS flow 10
3.2. The SPECS framework 11
4. Integration overview 16
4.1. Integration plan 16
4.2. Integration scenario template 21
5. Integration scenarios 23
5.1. Integration of SPECS core components 23
5.2. Integration of SPECS applications 32
6. Conclusions 37
7. Bibliography 38
Appendix 1. SPECS validation scenarios 39
SST.1 Secure_Storage_Selection 39
SST.2 Secure_Storage_Brokering_with_Client_Crypto 40
SST.3 Secure_Storage_with_Defined_CSP 42
SST.4 Secure_Storage_Brokering_with_Client_Crypto_Alert 45
SST.5 Secure_Storage_Brokering_with_Client_Crypto_Violation 47
SWC.1 Secure_Web_Container_Selection 50
SWC.2 Secure_Web_Container_Brokering 51
SWC.3 Secure_Web_Container_TLS_Enhanced 54
SWC.4 Secure_Web_Container_TLS_Enhanced_Alert 56
SWC.5 Secure_Web_Container_TLS_SVA_Enhanced_Violation 59
SWC.6 Secure_Web_Container_TLS_Multitenancy 62
SWC.7 Secure_Web_Container_Web_Pool_Replication_Enhanced_Alert 65
SWC.8 Secure_Web_Container_Web_Pool_Replication_Enhanced_Violation 67
NGDC.1 Data_Center_Bursting_for_Storage_Resources 70
NGDC.3 Data_Center_Storage_Selection 71
CRO.1 Security_Tokens_Acquisition 72
CRO.2 Security_Tokens_Validation 74
CRO.3 Security_Tokens_Revocation 75
CRO.4 Credential_Management 76
CRO.5 User_Direct_Registration 78
CRO.6 User_Registration_External_Account 79
CRO.7 User_Authentication_External_Account 80
CRO.8 Metric_Definition 82
CRO.9 Security_Mechanism_Development 83
CRO.10 SPECS_Application_Development 84
AAA.1 Identity_Management_Set-up 85
AAA.2 User_Registration 86

AAA.3 User_Access_Internal_Account 87
AAA.4 User_Access_External_Account.....88

Index of figures

Figure 1. Relationship with other deliverables 9
Figure 2. High level SPECS flow 10
Figure 3. High level SPECS architecture 12
Figure 4. Coverage of SPECS components and applications with validation scenarios 18

Index of tables

Table 1. The architecture of the Negotiation module 13
Table 2. The architecture of the Enforcement module 13
Table 3. The architecture of the Monitoring module 14
Table 4. The architecture of the SLA Platform 14
Table 5. The architecture of the Enabling Platform 15
Table 6. The architecture of the Vertical Layer 15
Table 7. SPECS security mechanisms 15
Table 8. SPECS user stories and validation scenarios 16
Table 9. SPECS user stories and associated SPECS applications 17
Table 10. Integration of SPECS core components 19
Table 11. Integration of SPECS applications 21
Table 12. Integration scenario specification template 22
Table 13. Integration scenario *Core-A1* 23
Table 14. Integration scenario *Core-B1* 24
Table 15. Integration scenario *Core-AB1* 24
Table 16. Integration scenario *Core-AB2* 25
Table 17. Integration scenario *Core-AB3* 25
Table 18. Integration scenario *Core-AB4* 26
Table 19. Integration scenario *Core-C1* 26
Table 20. Integration scenario *Core-C2* 27
Table 21. Integration scenario *Core-ABC1* 27
Table 22. Integration scenario *Core-ABC2* 28
Table 23. Integration scenario *Core-D1* 28
Table 24. Integration scenario *Core-CD1* 28
Table 25. Integration scenario *Core-ABCD1* 29
Table 26. Integration scenario *Core-ABCD2* 29
Table 27. Integration scenario *Core-ABCD3* 30
Table 28. Integration scenario *Core-ABCD4* 30
Table 29. Integration scenario *Core-ABCD5* 31
Table 30. Integration scenario *Core-ABCD6* 31
Table 31. Integration scenario *Core-ABCD7* 31
Table 32. Integration scenario *Core-ABCD8* 32
Table 33. Integration scenario *Core-ABCD9* 32
Table 34. Integration scenario *App-A1* 33
Table 35. Integration scenario *App-A2* 33
Table 36. Integration scenario *App-A3* 34
Table 37. Integration scenario *App-A4* 34
Table 38. Integration scenario *App-B1* 34
Table 39. Integration scenario *App-C1* 35
Table 40. Integration scenario *App-D1* 35
Table 41. Integration scenario *App-D2* 35
Table 42. Integration scenario *App-E1* 36
Table 43. Integration scenario *App-F1* 36

1. Introduction

The SPECS framework orchestrates a variety of complex processes to automatically manage the SLA life-cycle. On top of that, SPECS offers a wide set of security features through a large set of applications. Therefore, the SPECS framework is composed of a large number of components separated into a set of modules. In order to facilitate a fast development of the highly modular SPECS framework and enable a fast and easy detection of potential coding issues, the continuous integration process has been adopted in the project.

There are two documents of the task T1.5 that aim at presenting the SPECS integration process. Starting from the elicited requirements and the derived validation scenarios, in this document, we define and present a set of integration scenarios. The SPECS integration plan and associated integration scenarios are defined in a way that enables an efficient development of all SPECS validation applications. The complementing deliverable D1.5.2 presents the SPECS integration process from the technical point of view. It reports about the tools used during the integration and discusses the deployment aspects of the defined integration scenarios.

The document is structured as follows. In Section 2, relationships between this document and other deliverables of the project are discussed. Section 3 briefly summarizes the processes orchestrated by the SPECS framework and its architecture. The detailed SPECS integration plan is presented in Section 4, and the associated integration scenarios are defined in Section 5. The document concludes with a brief summary in Section 6. In order to help the reader with all the information needed to understand the integration plan, in Appendix 1 we provide the final set of validation scenarios defined in the context of the project.

2. Relationship with other deliverables

The integration task is tightly connected to almost all activities of the project. In order to define the integration plan and integrate all pieces of the framework, input is needed from all technical tasks and deliverables as depicted in Figure 1.

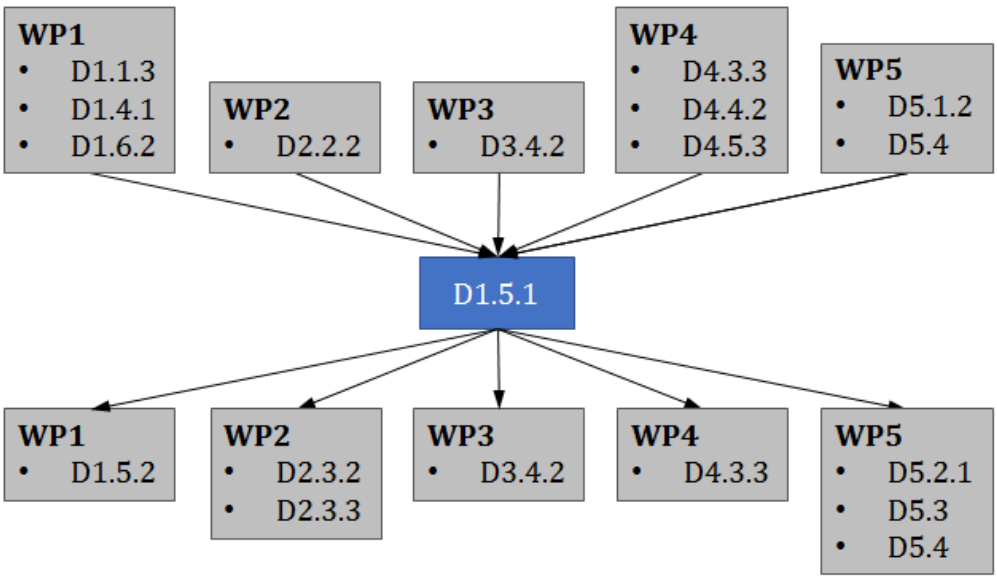


Figure 1. Relationship with other deliverables

The architectural and functional overview of the SPECS framework is extracted from the deliverable D1.1.3. Details related to the Enabling Platform are provided by the deliverable D1.6.2.

Deliverables D1.4.1 and D4.4.2 serve as the supporting documentation for the components of the Vertical Layer, whereas deliverables D2.2.2, D3.4.2, and D4.3.3 serve as guidelines for integrating the components of the Negotiation module, the Monitoring module, and the Enforcement Module, respectively.

Integration scenarios are built on top of the user stories and validation scenarios defined in D4.4.2, D5.1.2 and D5.4, whereas the integration and system testing activities are conducted as defined in the deliverable D4.5.3.

Implementation and testing aspects of integration scenarios defined in this deliverable are presented in D1.5.2. Feedback from the integration activities are provided to the developers of the core modules for which the final prototypes are discussed in deliverables D2.3.2, D2.3.3, D3.4.2, and D4.3.3, and to developers of the validation application presented in deliverables D2.3.2, D2.3.3, D5.2.1, D5.3, and D5.4.

3. SPECS framework overview

Before we discuss the SPECS integration plan and define integration scenarios, we briefly present the architecture of the SPECS framework, the SPECS flow that the elements of the framework orchestrate, and their dependencies. The detailed description of the SPECS design and behaviour is available in the deliverable D1.1.3.

3.1. The SPECS flow

The high level illustration of the SPECS flow is shown in Figure 2. It outlines the main operations performed by the elements of the SPECS framework during the SLA life-cycle and presents the role of the SPECS application. In the following we summarize the main steps.

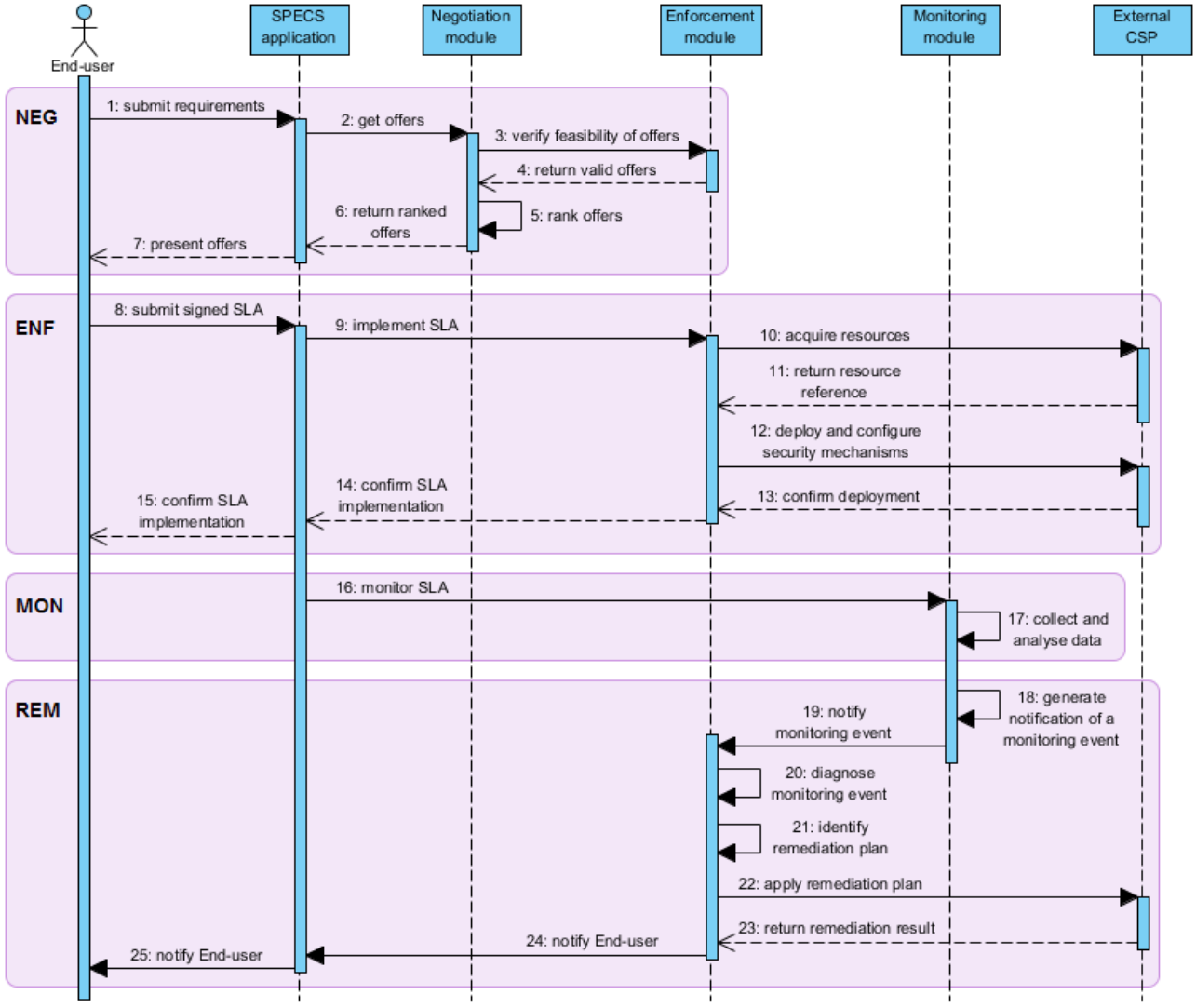


Figure 2. High level SPECS flow

The SPECS flow starts with the negotiation process (**NEG**), which enables End-users to negotiate the desired security level of the cloud service they wish to acquire. During this phase, the End-user selects the desired cloud service and the required security features. SPECS creates a set of feasible SLA Offers, ranks them according to the End-user's security requirements, and presents them to the End-user. The negotiation process ends with the End-

user either submitting a signed version of one of the presented SLA Offers or terminating the process by declining all of them (in this case the End-user can start a new cycle of the negotiation phase).

If the End-user selects and signs one of the presented SLA Offers, the SPECS application triggers the start of the SLA enforcement process (**ENF**). This phase consists of acquiring resources and deploying and configuring SPECS security mechanisms according to the EU's signed SLA.

After the successful SLA implementation, the End-user's SLA enters the monitoring phase (**MON**). The monitoring adapters on external resources continuously collect monitoring data and send it to the SPECS Monitoring module, which stores it and filters it.

Whenever the Monitoring module detects a possible SLA alert or an SLA violation, a notification is created and sent to the Enforcement module. Notification invokes the start of the remediation phase (**REM**). SPECS analyses the event and identifies a remediation plan. The execution of the remediation plan can result in a success (the alert/violation is resolved) or a failure (SPECS is unable to automatically mitigate the SLA alert or recover from the SLA violation). In any case, SPECS notifies the End-user through the SPECS application. The End-user now has an opportunity to either (i) accept the risk and continue with the SLA monitoring (if possible), (ii) terminate the SLA, or (iii) renegotiate the SLA, which triggers another (adjusted) version of the negotiation phase (**NEG**).

Note that, all signed SLAs and the information regarding the security services offered by SPECS are managed by the SLA Platform module which is excluded from the flow diagram for the sake of simplicity.

Further details about the individual phases are available in deliverables of the dedicated workpackages: negotiation in WP2, enforcement and remediation in WP4, while monitoring in WP3. The description of the role of the SLA Platform and the detailed summary of all phases are available in the deliverable D1.1.3, and the interfaces are presented in the deliverable D1.3.

3.2. The SPECS framework

The description of the SPECS flow can be extended with the details of the artifacts orchestrating the phases of the SLA life-cycle. The focus of this section is just that: to briefly summarize the SPECS architecture and provide a link to the SPECS flow for each element of the SPECS framework. Such a description provides for a better understanding of the SPECS integration process reported in Section 4 and integration scenarios defined in Section 5.

The SPECS framework, orchestrated by the SPECS application, comprises the following modules (as depicted in Figure 3):

- **Negotiation module**: Orchestrates the SLA negotiation process.
- **Enforcement module**: Supports the SLA negotiation process and oversees the SLA implementation and remediation phases.
- **Monitoring module**: Provides SLA monitoring functionalities.

- **SLA Platform**: Comprises components to manage SLAs and to provide information about the offered cloud security services.
- **Vertical Layer**: Offers cross-cutting services used by all core modules (e.g., user management, auditing)
- **Enabling Platform**: Serves as the deployment and execution environment for all core modules.

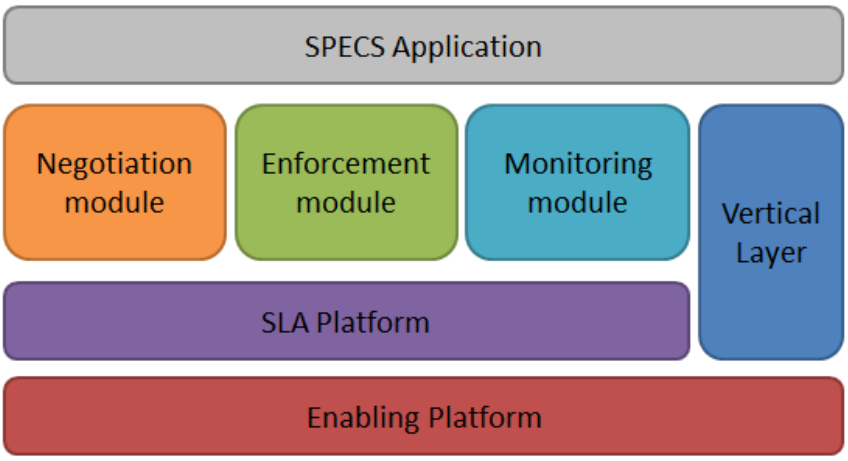


Figure 3. High level SPECS architecture

In the remainder of this section we present the design of each module and summarize the main functionalities and existing compile-time dependencies for the comprising components. Additionally, for each module, we provide a link to deliverables where the interested reader can find further details.

Note that the SPECS Data Model, which is a library that contains common classes shared among SPECS components, is presented in deliverable D1.4.1.

The Negotiation module comprises three components as reported in Table 1. In Table 2 we report the design of the Enforcement module. The architecture of the Monitoring module is presented in Table 3. The SLA Platform and the Enabling Platform are reported in Table 4 and Table 5, respectively. The components of the Vertical Layer are discussed in Table 6. Finally, the security mechanisms (which are developed under the umbrella of the Enforcement module) are reported in Table 7.

Note that two components of the Enforcement module, namely the Broker that orchestrates the acquisition of external cloud resources and the Chef[1] Server that automates management of the acquired resources and deployed SPECS components, both presented in the deliverable D4.2.2, are integrated with the Implementation component. Therefore in Table 2 we report

[1] The underlaying technology used in SPECS to deploy and configure the SPECS platform at the start-up and the SPECS security mechanisms during runtime. For details see D4.2.2 and [3].

their combined functionalities. Similarly, in the integration plans and integration scenarios, we consider them altogether (as the Implementation component).

SPECS component	Functionalities	Dependencies	Deliverables
SLO Manager	• Manages customization of an SLA Template according to the End-user's security requirements. • Builds SLA Offers according to the generated supply chains.	• Supply Chain Manager • Data Model	• D2.1.2 (requirements) • D2.2.2 (design) • D2.3.1, D2.3.2, D2.3.3 (implementation) • D1.3 (API)
Supply Chain Manager	• Prepares the input according to the customized SLA Template and triggers the creation of supply chains.	• Data Model	
Security Reasoner	• Evaluates and ranks SLA Offers.	• Data Model	

Table 1. The architecture of the Negotiation module

SPECS component	Functionalities	Dependencies	Deliverables
Planning	• Builds supply chains according to the SLA Templates customized with End-user's security requirements. • Builds an implementation plan according to the signed SLA.	• Auditing • Data Model	• D4.1.2 (requirements) • D4.2.2 (design) • D4.3.x (implementation) • D1.3 (API)
Implementation (with Broker and Chef Server)	• Executes implementation plans (acquires resources, deploys and configures SPECS components) provided by the Planning component. • Executes remediation plans (performs reconfigurations) provided by the RDS component.	• Broker • Auditing • Data Model	
Diagnosis	• Performs the analysis of potential SLA alerts and violations.	• Auditing • Data Model	
Remediation Decision System (RDS)	• Builds remediation plans to recover from SLA alerts and violations analysed by the Diagnosis component.	• Auditing • Data Model	

Table 2. The architecture of the Enforcement module

SPECS component	Functionalities	Dependencies	Deliverables
Event Hub	• Routes events collected from the Monitoring Adapters deployed on external resources.	• Event Archiver	• D3.2 (requirements)

Event Aggregator	• Consumes events according to the configured aggregation rules.	/	• D3.3 (design) • D3.4.x (implementation) • D1.3 (API)
Event Archiver	• Stores all events that are exchanged through the Event Hub.	/	
Monitoring Policy Filter (MoniPoli)	• Filters events to detect potential SLA alerts and violations.	• Event Hub	
SLO Metrics Exporter (SLOM Exporter)	• Forwards notifications of SLA alerts and violations received from the MoniPoli to the Enforcement module.	/	
CTP	• Forwards monitoring data relevant to the End-users to the SPECS Application.	/	

Table 3. The architecture of the Monitoring module

SPECS component	Functionalities	Dependencies	Deliverables
SLA Manager	• Manages signed SLAs during their life-cycle.	/	• D1.2 (requirements) • D1.4.1 (design) • D1.4.2 (implementation) • D1.3 (API)
Service Manager	• Manages information (metadata) related to security mechanisms able to enforce and monitor security services offered to End-users.	• Data Model	
Metric Catalogue	• Stores and maintains information related to security metrics.	/	
Interoperability Layer	• Enables interoperation and decoupling among SPECS components by offering a single access point for all APIs.	/	

Table 4. The architecture of the SLA Platform

SPECS component	Functionalities	Dependencies	Deliverables
Launcher	• Provides a collection of services responsible for the Enabling Platform infrastructure deployment and resource registration.	/	• D1.2 (requirements) • D1.1.3 (design) • D1.6.x (implementation, API)
Custom OS	• A custom, lightweight operating system enriched with a set of remote runtime operation services that are able to setup, on demand, the environment for the hosted components.	/	
Core Repository	• A collection of Chef recipes that define the deployment and runtime processes of the SPECS core	/	

	components.		
Mechanisms Repository	• A collection of Chef recipes that define the deployment and runtime processes of the SPECS security mechanisms.	/	

Table 5. The architecture of the Enabling Platform

SPECS component	Functionalities	Dependencies	Deliverables
User Manager	• Manages registration and basic authentication and authorization features of SPECS users.	/	• D1.4.1 (design) • D1.4.2 (implementation) • D1.3 (API)
Auditing	• Provides logging functionalities to all components of the SPECS framework.	• Data Model	
Security Tokens	• Protects internal communication among SPECS components	• Data Model	• D4.4.x (requirements, design, implementation, and API)
Credential Service	• Provides a service to securely store and manage credentials needed to access external resources.	• Data Model	

Table 6. The architecture of the Vertical Layer

SPECS component	Functionalities	Dependencies	Deliverables
Secure Web Server (WebPool)	• Provisions web servers and offers security assurances through redundancy and diversity.	/	• D4.1.2 (requirements) • D4.2.2 (design) • D4.3.x (implementation) • D1.3 (API)
Transport Layer Security (TLS)	• Offers different configurations of the TLS protocol.	/	
Software Vulnerability Assessment (SVA)	• Provides configurable software vulnerability scanners and enables periodic vulnerability scans.	/	
Database and Backup (DBB)	• Offers secure storage with backup with the capability of detecting violations related to write-serializability and read-freshness.	/	
End-2-End Encryption (E2EE)	• Extends the DBB mechanisms with client-side encryption.	/	
DoS Detection and Mitigation (DoS)	• Offers detection, classification, and mitigation of DoS attacks.	/	
AAA	• Offers federated identity and access management features over resources and services of different applications.	/	

Table 7. SPECS security mechanisms

4. Integration overview

Purpose of the SPECS integration plan is to present the order in which SPECS components are integrated so as to enable an efficient development of SPECS applications.

In the following Subsection 4.1 we present a detailed integration plan that is based on the End-user's requirements (i.e. user stories and validation scenarios) and the SLA life-cycle. The integration plan is structured in a way that enables an efficient implementation of the example applications. In Subsection 4.2 we introduce a template that is used in Section 5 to describe details of each defined integration scenario.

The technical aspects of the SPECS integration process are presented in deliverable D1.5.2.

4.1. Integration plan

The SPECS integration plan is built on top of five user stories and associated validation scenarios defined in tasks T5.1, T4.2, T5.4, and reported in Table 8. All SPECS validation scenarios are also presented in Appendix 1.

User story	Validation scenario (ID, Name)	
Secure Storage	SST.1	*Secure_Storage_Selection*
	SST.2	*Secure_Storage_Brokering_with_Client_Crypto*
	SST.3	*Secure_Storage_with_Defined_CSP*
	SST.4	*Secure_Storage_Brokering_with_Client_Crypto_Alert*
	SST.5	*Secure_Storage_Brokering_with_Client_Crypto_Violation*
Secure Web Container	SWC.1	*Secure_Web_Container_Selection*
	SWC.2	*Secure_Web_Container_Brokering*
	SWC.3	*Secure_Web_Container_TLS_Enhanced*
	SWC.4	*Secure_Web_Container_TLS_Enhanced_Alert*
	SWC.5	*Secure_Web_Container_TLS_SVA_Enhanced_Violation*
	SWC.6	*Secure_Web_Container_TLS_Multitenancy*
	SWC.7	*Secure_Web_Container_Web_Pool_Replication_Enhanced_Alert*
	SWC.8	*Secure_Web_Container_Web_Pool_Replication_Enhanced_Violation*
Next-Generation Data Centers	NGDC.1	*Data_Center_Bursting_for_Storage_Resources*
	NGDC.3	*Data_Center_Storage_Selection*
Crosscutting	CRO.1	*Security_Tokens_Acquisition*
	CRO.2	*Security_Tokens_Validation*
	CRO.3	*Security_Tokens_Revocation*
	CRO.4	*Credential_Management*
	CRO.5	*User_Direct_Registration*
	CRO.6	*User_Registration_External_Account*
	CRO.7	*User_Authentication_External_Account*
	CRO.8	*Metric_Definition*
	CRO.9	*Security_Mechanism_Development*
	CRO.10	*SPECS_Application_Development*
Secure Storage with Identity Management	AAA.1	*Identity_Management_Set-up*
	AAA.2	*User_Registration*
	AAA.3	*User_Access_Internal_Account*
	AAA.4	*User_Access_External_Account*

Table 8. SPECS user stories and validation scenarios

In the first year we defined the Security-Oriented Dashboard user story that revolved around an End-user that wants to evaluate Cloud Service Providers (CSPs) on the basis of provided security requirements. The user story was detailed with two validation scenarios (in deliverable D5.1.1). In year 2, a deeper analysis highlighted similarities between these two validation scenarios and validation scenarios defined for other user stories, hence both scenarios were discarded. Nevertheless, we have developed an application that covers their main aspects.

Note that the first three user stories (Secure Storage, Secure Web Container, and Next-Generation Data Centers) implement all phases of the SLA life-cycle, whereas the other validation scenarios refer to inter-application context; these scenarios are out of scope of a single user story but are needed for its execution.

As discussed in deliverables D5.1.2, D4.4.2, and D5.4, and seen in Table 9, we have five user stories that are implemented with five validation applications. With the Secure Storage and the ngDC applications, the End-user can acquire a file storage system enhanced with a set of specific security features as requested by the End-user. These two applications implement similar user stories in a different context; the Secure Storage application is integrated with XLAB's Koofr [1] in the context of the IM1[2], whereas the ngDC application integrates EMC's hardware storage solutions and the ViPR [2] software layer in the context of the IM2[3]. Further details are provided in deliverables D5.2.1 and D5.2.2 (the Secure Storage) and D5.3 (the ngDC).

User story	Validation application	Deliverables
Secure Storage	Secure Storage	D5.1.2, D5.2.2
Next-Generation Data Centers	ngDC	D5.3
Secure Web Container	Secure Web Container	D5.1.3, D1.5.2
Security-Oriented Dashboard	Security Reasoner	D2.3.1
Secure Storage with Identity Management	AAA-as-a-Service	D5.4

Table 9. SPECS user stories and associated SPECS applications

A pool of virtual machines, on which an End-user can run a set of her/his own applications, can be acquired with the Secure Web Container application. The End-user can also negotiate the desired security level that is enforced by SPECS. Further details are available in D5.1.3 and D1.5.2.

The Security Reasoner application implements techniques to compare the security levels offered by different CSPs. The application helps End-users to decide on the best CSP that can fulfil her/his security requirements. Further details are available in D2.3.1.

The AAA-as-a-Service application comprises a set of functionalities, offered "as-a-Service", associated to the identity management and access control. For further details see D5.4.

[2] Interaction Model 1: Deployed on a private/public provider, where the SPECS Owner is a third-party. See D1.2.
[3] Interaction Model 2: The SPECS framework is on the same resource that host the target service. See D1.2.

In addition to the introduced user stories, we develop the Metrics Catalogue application which comprises a database and an interface to manage data for all security metrics available in SPECS. For details see deliverables D5.1.3 and D1.5.2.
In order to plan integration activities in a way that enables an efficient development of the SPECS applications, we analyse the coverage of SPECS artifacts with validation scenarios. The mapping is presented with a matrix in Figure 4. The matrix presents SPECS components (where the SLAP, NEG, ENF, MON, VL, and SM represent the SLA Platform, Negotiation module, Enforcement module, Monitoring module, Vertical Layer, and Security Mechanisms, respectively) and applications (APP) involved in validation scenarios (VS) reported in Table 8.

Note that the components of the Enabling Platform enable the execution of all processes in SPECS, thus, they are covered by all validation scenarios. For the sake of simplicity, we leave the components of the Enabling Platform out of the coverage matrix.

	SLAP				NEG			ENF				MON							VL				SM							APP					
VS	SLA Manager	Service Manager	Metric Catalogue	Interoperability Layer	SLO Manager	Supply Chain Manager	Security Reasoner	Planning	Implementation	Diagnosis	RDS	Event Hub	Event Aggregator	MoniPoli Filter	SLOM Exporter	CTP	Event Archiver	Nmap	Auditing	Security Tokens	Credential Service	User Manager	WebPool	SVA	TLS	DoS	AAA	DBB	E2EE	Secure Web Container	Metric Catalogue	Security Reasoner	Secure Storage	ngDC	AAAaaS
SST.1	x	x		x	x	x	x	x										x		x		x										x	x		
SST.2	x	x		x	x	x	x	x	x			x	x	x				x	x	x	x	x						x	x				x		
SST.3	x	x		x	x	x	x	x	x			x	x	x				x	x	x	x	x						x	x				x		
SST.4	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						x	x				x		
SST.5	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						x	x				x		
SWC.1	x	x		x	x	x	x	x										x		x		x										x			
SWC.2	x	x		x	x	x	x	x	x			x	x	x				x	x	x	x	x	x							x		x			
SWC.3	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x				x					
SWC.4	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						x					
SWC.5	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x					
SWC.6	x	x		x	x	x	x	x	x			x	x	x				x	x	x	x	x	x		x	x				x					
SWC.7	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x							x					
SWC.8	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x							x					
NGDC.1	x	x		x	x	x	x	x										x		x		x												x	
NGDC.3	x	x		x	x	x	x	x										x		x		x										x		x	
CRO.1																				x										x			x	x	
CRO.2																				x										x			x	x	
CRO.3									x											x										x			x	x	
CRO.4																				x	x									x			x	x	
CRO.5																						x								x			x	x	
CRO.6																						x								x			x	x	
CRO.7																						x								x			x	x	
CRO.8			x																												x				
CRO.9		x	x																																
CRO.10	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
AAA.1																											x								x
AAA.2																											x								x
AAA.3																											x								x
AAA.4																											x								x
SUM	16	17	3	16	16	16	16	16	13	8	8	12	12	12	8	8	8	16	12	20	13	19	8	3	4	3	5	5	5	15	2	5	13	10	5

Figure 4. Coverage of SPECS components and applications with validation scenarios

As seen from the matrix in Figure 4, the Web Container and the Secure Storage applications cover the highest proportion of validation scenarios (the Web Container implements 15 and the Secure Storage implements 13 out of 25 validation scenarios). Therefore, the integration activities are planned so that these two applications are developed first, with the minimum need for mock-ups, and covering the basic steps of core functionalities.

In order to enable the basic steps of negotiation, enforcement, monitoring, and remediation, we integrate the components of the core modules (SLAP, NEG, ENF, and MON) and the Vertical Layer (VL) gradually as shown in Table 10.

Integration Scenario	Default SPECS App.	SLAP			NEG			ENF				MON							VL			
		SLA Manager	Service Manager	Interoperability Layer	SLO Manager	Supply Chain Manager	Security Reasoner	Planning	Implementation	Diagnosis	RDS	Event Hub	Event Aggregator	MoniPoli Filter	SLOM Exporter	CTP	Event Archiver	Nmap	Auditing	Security Tokens	Credential Service	User Manager
Core-A1		x			x																	
Core-B1			x		x	x		x														
Core-AB1		x	x		x	x		x														
Core-AB2		x	x		x	x		x	x													
Core-AB3		x	x		x	x		x	x			x		x								
Core-AB4	x	x	x		x	x		x	x			x		x								
Core-C1												x	x	x	x		x					
Core-C2												x	x	x	x	x	x					
Core-ABC1		x	x		x	x		x	x			x	x	x	x	x	x					
Core-ABC2	x	x	x		x	x		x	x			x	x	x	x	x	x					
Core-D1								x	x	x	x											
Core-CD1								x	x	x	x	x	x	x	x	x	x					
Core-ABCD1		x	x		x	x		x	x	x	x	x	x	x	x	x	x					
Core-ABCD2		x	x		x	x	x	x	x	x	x	x	x	x	x	x	x					
Core-ABCD3		x	x		x	x	x	x	x	x	x	x	x	x	x	x	x			x		
Core-ABCD4		x	x		x	x	x	x	x	x	x	x	x	x	x	x	x			x		x
Core-ABCD5		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x		x
Core-ABCD6		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x
Core-ABCD7		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x
Core-ABCD8		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Core-ABCD9	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 10. Integration of SPECS core components

For the integration of the core components of the SPECS framework we use the *default SPECS Application* introduced in deliverable D5.1.3. During the integration process we have developed three versions of the application.

The initial standalone version of the default SPECS Application is interfaced with mock-ups for other components so that it enables and automates all functionalities needed to negotiate, implement, monitor, and remediate an SLA.

As shown in Table 10, we sequentially integrate core components to enable separate phases of the SLA life cycle (to replace mock-ups with SPECS components and we verify the interfaces and the data flow among integrated artifacts). After each phase is enabled, we additionally test the integration with the default SPECS Application and thus develop the second version of it (scenarios *Core-AB4* and *Core-ABC2*).

Replacing all mock-ups in the default SPECS Application with integrated SPECS components (i.e. when the *Core-ABCD9* integration scenario is deployed), completes the core integration and the customized version of the default SPECS Application (the third and final version) is ready to be integrated with SPECS security mechanisms; this way we develop specific SPECS applications (e.g., Secure Web Container, Secure Storage).

Note that all integration tests run on top of SPECS Testbed (see D1.6.1 and D1.6.2), which is our default Enabling Platform. Since (i) the components of the Enabling Platform are required for the execution of every integration scenario and (ii) they do not interoperate with other SPECS components directly, we do not explicitly mention it in Table 10.

In the following we provide a description of the integration scenarios defined in Table 10.

The first phase of the SLA life-cycle is the SLA negotiation. To enable the elicitation of security requirements from End-users, we execute the *Core-A1* scenario which integrates components that support presentation of security offers to End-users and customization of SLA Templates.

In parallel, we integrate the Service Manager, SLO Manager, Supply Chain Manager, and Planning components (*Core-B1*), which orchestrate generation of valid supply chains based on the customized SLA Templates.

Afterwards, we execute integration scenario *Core-AB1*, which integrates components from the *Core-A1* and *Core-B1* integration scenarios. All involved components support the basic version of the SLA negotiation process (negotiation without the ranking of SLA Offers). Later we add basic functionalities related to the SLA enforcement phase by integrating the Implementation component (*Core-AB2*), the Monitoring Policy Filter component, and the Event Hub component (*Core-AB3*). Execution of the integration scenario *Core-AB4*, which extends the *Core-AB3* scenario with the default SPECS application, completes the integration of components responsible for the SLA negotiation and SLA enforcement phase.

Independently, we integrate components of the Monitoring module (scenarios *Core-Cx*). Later we merge the *Core-AB3* and *Core-C2* integration scenarios to enable the flow up to the SLA monitoring phase. The resulting integration scenario *Core-ABC1* is later extended with the default SPECS application (*Core-ABC2*).

The next step is to enable the SLA remediation phase. To this end, all Enforcement components are integrated in the *Core-D1* scenario. Afterwards we execute the *Core-CD1* and *Core-ABCD1* integration scenarios. The first one combines monitoring and remediation components (merges *Core-C2* and *Core-D1*), and the latter one merges components involved in the SLA negotiation, SLA enforcement, SLA monitoring, and SLA remediation phases (merges *Core-ABC1* and *Core-CD1*).

When the basic SPECS flow is integrated, we add the remaining components according to the number of validation scenarios they cover and the functionalities they support (integration scenarios *Core-ABCDx*). These scenarios complete the integration process for core components of the SPECS framework.

The specific SPECS applications (APP) are continuously integrated with security mechanisms (SM) and some core components (of the SLA Platform – SLAP, and Negotiation module – NEG) as presented in Table 11.

Development of the Secure Web Container application is defined with the *App-Ax* scenarios, which gradually integrate the security mechanisms offered through this application. The order is defined based on the number of validation scenarios that they cover. With the *App-B1* and *App-C1* scenarios we present integration of the security mechanisms with the Secure Storage and the ngDC application. Finally, the *App-Dx*, *App-E1*, and *App-F1* integration scenarios demonstrate development of the remaining SPECS applications.

Note that the ngDC application does not integrate any of SPECS security mechanisms, but uses EMC's ViPR solution to offer secure storage.

Integration Scenario	SLAP	NEG	SM								APP					
	Metric Catalogue	Security Reasoner	WebPool	TLS	SVA	DoS	AAA	DBB	E2EE	ViPR	Secure Web Container	Secure Storage	ngDC	AAAaaS	Metric Catalogue	Security Reasoner
App-A1			x								x					
App-A2			x	x							x					
App-A3			x	x	x						x					
App-A4			x	x	x	x					x					
App-B1								x	x			x				
App-C1										x			x			
App-D1							x			x				x		
App-D2							x	x	x	x				x		
App-E1	x														x	
App-F1		x														x

Table 11. Integration of SPECS applications

The template used for a detailed description of scenarios is presented in the next subsection.

4.2. Integration scenario template

All integration scenarios defined in Table 10 and Table 11 are later (in Section 5) detailed according to the template shown below in Table 12.

For each integration scenario we give a brief natural language description outlining the involved components and their roles. Since integration scenarios are built gradually, adding artifacts one by one, we report, for each scenario, a link to those scenarios that form a base for the current one and a list of additionally integrated artifacts. For the sake of completeness, we finally list all integrated SPECS artifacts (core components, security mechanisms, and SPECS applications).

Scenario ID		*The ID of the integration scenario as defined in Table 10 and Table 11.*
Description		*A natural language description of the integration scenario outlining the roles of the involved artifacts.*
Base scenarios (IDs)		*The list of IDs of integration scenarios that serve as the base for the presented one.*
Added artifacts		*A list of components/mechanisms/application that extend the associated base integration scenario;* *current scenario = base scenario + added artifacts.*
Core components	**SLAP**	*A list of integrated artifacts (as presented in Table 10 and Table 11).*
	NEG	
	ENF	
	MON	
	VL	
Security mechanisms		
SPECS applications		

Table 12. Integration scenario specification template

5. Integration scenarios

This section presents integration scenarios defined according to the SPECS integration plan discussed in Section 4.1. First, scenarios that detail integration of SPECS core components are presented (Section 5.1), then scenarios that specify the integration of SPECS applications are listed (see Section 5.2). We use the template introduced in Section 4.2.

Note that all deployment details associated to each integration scenario are presented in deliverable D1.5.2.

The technical details associated to each artifact are available in the relevant prototype deliverables (for the SLA Platform and the Vertical Layer see D1.4.2, for the Negotiation module see D2.3.1 and D2.3.2, for the Monitoring module see D3.4.1 and D3.4.2, for the Enforcement module and security mechanisms see D4.3.x, and for the SPECS applications see D5.1.3, D5.2.2, D5.3, and D5.4).

5.1. Integration of SPECS core components

The SPECS integration process starts with the SLA Manager and the SLO Manager components. The SLA Manager component enables the management of SLAs (creation of SLAs in the negotiation phase and their retrieval during runtime) and is therefore involved in all phases of the SLA life-cycle. The SLO Manager component enables the customization of SLA Templates with security requirements from End-users, and builds SLA Offers according to the supply chains that are generated for the customized SLA Templates. Therefore, the SLO Manager is the core component involved in the SLA (re)negotiation phase. Hence, the integration of the SLO Manager and the SLA Manager components defines the main scenario that serves as the base for many other integration scenarios. For details see Table 13.

Scenario ID		*Core-A1*
Description		This scenario integrates the SLA Manager component (SLA Platform) and the SLO Manager component (Negotiation module) which provide basic functionalities for the creation and management of SLAs.
Base scenarios (IDs)		/
Added artifacts		SLA Manager, SLO Manager
Core components	**SLAP**	SLA Manager
	NEG	SLO Manager
	ENF	/
	MON	/
	VL	/
Security mechanisms		/
SPECS applications		/

Table 13. Integration scenario *Core-A1*

When the End-user specifies the desired security features and SPECS prepares an SLA Template accordingly, the Service Manager component is needed to provide information about security mechanisms that are able to enforce and monitor the SLA. Furthermore, the Supply Chain Manager component is needed to take the customized SLA Template and prepare the input for the Planning component that builds supply chains (identifies CSPs and determines the number of resources needed to deploy mechanisms that enforce and monitor

the SLA). Details for the scenario that integrates these two components are reported in Table 14.

Scenario ID		*Core-B1*
Description		This scenario integrates the Service Manager component (SLA Platform), the SLO Manager and the Supply Chain Manager (Negotiation module), and the Planning component (Enforcement module). The Supply Chain Manager component (which depends on the SLO Manager) prepares the input and triggers the Planning component to build supply chains according to the SLA Template and the information provided by the Service Manager.
Base scenarios (IDs)		/
Added artifacts		Service Manager, SLO Manager, Supply Chain Manager, Planning
Core components	**SLAP**	Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning
	MON	/
	VL	/
Security mechanisms		/
SPECS applications		/

Table 14. Integration scenario *Core-B1*

The basic version of the negotiation phase (all steps except for the ranking of SLA Offers) is available with the merge of the *Core-A1* and *Core-B1* integration scenarios. As discussed above, *Core-A1* integrates components that orchestrate elicitation of End-user's requirements and generation of SLA Offers according to the set of built supply chains, and *Core-B1* integrates components involved in the process of building supply chains. Details for the *Core-AB1* scenario covering the complete (basic) negotiation phase are provided in Table 15.

Scenario ID		*Core-AB1*
Description		This scenario integrates the *Core-A1* and *Core-B1* scenarios. Involved artifacts enable the complete negotiation phase (the basic version without ranking SLA Offers).
Base scenarios (IDs)		*Core-A1*, *Core-B1*
Added artifacts		/
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning
	MON	/
	VL	/
Security mechanisms		/
SPECS applications		/

Table 15. Integration scenario *Core-AB1*

After the negotiation phase, the Planning and Implementation components are responsible for the SLA implementation which includes (i) building an implementation plan according to the signed SLA and associated supply chain and (ii) acquisition of cloud resources, deployment of security mechanisms, and their configuration. Added functionalities of the Planning component (building implementation plans) and functionalities of the Implementation

component (acquisition and deployment) are incorporated in the *Core-AB2* scenario, reported in Table 16, which extends the *Core-AB1* integration scenario.

Scenario ID		*Core-AB2*
Description		This scenario extends the *Core-AB1* integration scenario with the Implementation component (Enforcement module) responsible for the acquisition and configuration of cloud resources.
Base scenarios (IDs)		*Core-AB1*
Added artifacts		Implementation
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning, Implementation
	MON	/
	VL	/
Security mechanisms		/
SPECS applications		/

Table 16. Integration scenario *Core-AB2*

In order to enable the complete flow up to the SLA monitoring phase, we extend the Core-AB2 integration scenario with the Monitoring Policy (MoniPoli) Filter component that is configured during the SLA implementation phase to filter all collected monitoring events and extract potential SLA alerts and violations during runtime. Since the MoniPoli Filter component directly depends on the Event Hub component, we integrate it as well. For details see Table 17.

Scenario ID		*Core-AB3*
Description		This scenario extends the *Core-AB2* integration scenario with the MoniPoli Filter component (Monitoring module) that is configured during the SLA implementation phase and is responsible for the identification of possible SLA alerts and violations.
Base scenarios (IDs)		*Core-AB2*
Added artifacts		MoniPoli Filter, Event Hub
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning, Implementation
	MON	MoniPoli Filter, Event Hub
	VL	/
Security mechanisms		/
SPECS applications		/

Table 17. Integration scenario *Core-AB3*

The *Core-AB3* integration scenario is finally tested with the default SPECS Application. The resulting *Core-AB4* integration scenario is detailed in Table 18.

Scenario ID		*Core-AB4*
Description		This scenario extends the *Core-AB3* integration scenario with the default SPECS Application. The involved artifacts enable the basic version of the SPECS flow up to the SLA monitoring phase (SLA negotiation and SLA implementation).
Base scenarios (IDs)		*Core-AB3*
Added artifacts		Default SPECS Application
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning, Implementation
	MON	MoniPoli Filter, Event Hub
	VL	/
Security mechanisms		/
SPECS applications		Default SPECS Application

Table 18. Integration scenario *Core-AB4*

In parallel with previous scenarios, we define the *Core-C1* scenario (in Table 19) where we integrate

- the Event Hub component, which collects events from various monitoring adapters and routes them towards other components of the Monitoring module,
- the MoniPoli Filter component, which filters events to extract only those that can potentially represent SLA alerts and violations,
- the Event Aggregator component, which consumes events from the Event Hub and aggregates them according to the predefined aggregation rules,
- the SLOM Exporter component, which sends notifications about the potential SLA alerts and violation to the Enforcement module according to the results produced by the MoniPoli Filter component, and
- the Event Archiver component, which stores all collected monitoring data.

Scenario ID		*Core-C1*
Description		This scenario integrates the components of the Monitoring module that enable collecting, aggregating, filtering and archiving events, and notifying possible SLA alerts and violations to the Enforcement module.
Base scenarios (IDs)		/
Added artifacts		Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver
Core components	**SLAP**	/
	NEG	/
	ENF	/
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver
	VL	/
Security mechanisms		/
SPECS applications		/

Table 19. Integration scenario *Core-C1*

Later we add the CTP component, which securely exports the collected monitoring data to the SPECS Application to be presented to End-users. The details of the associated *Core-C2* integration scenario are provided in Table 20.

Scenario ID		*Core-C2*
Description		This scenario extends the *Core-C1* scenario with the CTP component (Monitoring module), which exports monitoring data relevant to the End-user to the SPECS Application. The set of integrated components enables all monitoring functionalities.
Base scenarios (IDs)		*Core-C1*
Added artifacts		CTP
Core components	**SLAP**	/
	NEG	/
	ENF	/
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	/
Security mechanisms		/
SPECS applications		/

Table 20. Integration scenario *Core-C2*

With the definition of the next integration scenario (namely *Core-ABC1*, reported in Table 21), which merges the *Core-AB3* and *Core-C2* integration scenarios, we cover the complete SPECS flow, up to the SLA remediation phase (we integrate the basic SLA negotiation, SLA implementation, and SLA monitoring phases). As presented above, the *Core-AB3* scenario integrates components that oversee the steps of the SLA negotiation and SLA implementation, while *Core-C2* integrates components responsible for the monitoring functionalities.

Scenario ID		*Core-ABC1*
Description		This scenario integrates the *Core-AB3* and *Core-C2* scenarios. Involved artifacts enable the basic SLA negotiation (without ranking SLA Offers), SLA implementation, and SLA monitoring phases.
Base scenarios (IDs)		*Core-AB3*, *Core-C2*
Added artifacts		/
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning, Implementation
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	/
Security mechanisms		/
SPECS applications		/

Table 21. Integration scenario *Core-ABC1*

Afterwards we extend the Core-ABC1 integration scenario with the default SPECS Application to test the interfaces. The resulting Core-ABC2 integration scenario is reported in Table 22.

In parallel to the "monitoring" integration scenarios, we define a scenario that integrates components orchestrating the SLA remediation phase. Integration of the Diagnosis component, which is responsible for analysing SLA alerts and violations, the RDS component, which is responsible for preparing remediation plans, and the Planning and Implementation components which execute it, is further detailed in Table 23.

Scenario ID		*Core-ABC2*
Description		This scenario extends the *Core-ABC1* scenario with the default SPECS Application. The involved artifacts enable the basic version of the SLA negotiation, SLA implementation, and SLA monitoring.
Base scenarios (IDs)		*Core-ABC1*
Added artifacts		Default SPECS Application
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning, Implementation
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	/
Security mechanisms		/
SPECS applications		Default SPECS Application

Table 22. Integration scenario *Core-ABC2*

Scenario ID		*Core-D1*
Description		This scenario integrates the Diagnosis and the RDS components (Enforcement module), which analyse monitoring events and prepare remediation plans according to the performed analysis.
Base scenarios (IDs)		/
Added artifacts		Planning, Implementation, Diagnosis, RDS
Core components	**SLAP**	/
	NEG	/
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	/
	VL	/
Security mechanisms		/
SPECS applications		/

Table 23. Integration scenario *Core-D1*

The next *Core-CD1* scenario merges the *Core-C2* and *Core-D1* scenarios, which connect monitoring functionalities with the SLA remediation phase. Further details are available in Table 24.

Scenario ID		*Core-CD1*
Description		This scenario merges the *Core-C2* and *Core-D1* integration scenarios. Involved components enable the monitoring and remediation steps.
Base scenarios (IDs)		*Core-C2*, *Core-D1*
Added artifacts		/
Core components	**SLAP**	/
	NEG	/
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	/
Security mechanisms		/
SPECS applications		/

Table 24. Integration scenario *Core-CD1*

The first integration scenario that enables the basic version of the entire SPECS flow is denoted by *Core-ABCDE1* and is reported in Table 25. It merges the scenario *Core-ABC1* covering the negotiation and implementation steps, with the *Core-CD1* scenario covering the monitoring and remediation phases.

Scenario ID		*Core-ABCD1*
Description		This scenario merges the *Core-ABC1* and *Core-CD1* integration scenarios, and enables the basic version of the entire SPECS flow (all steps except SLA ranking).
Base scenarios (IDs)		*Core-ABC1*, *Core-CD1*
Added artifacts		/
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	/
Security mechanisms		/
SPECS applications		/

Table 25. Integration scenario *Core-ABCD1*

Once the basic version of the entire SPECS flow is integrated, we gradually add remaining components of the framework. First we add the Security Reasoner component, which provides SLA evaluating and ranking functionalities. The defined *Core-ABCD2* integration scenario (reported in Table 26) covers the entire SPECS flow.

Scenario ID		*Core-ABCD2*
Description		This scenario extends the *Core-ABCD1* scenario by integrating the Security Reasoner component (Negotiation module). By including functionalities related to the evaluation and ranking of the SLAs, this scenario enables the complete SPECS flow.
Base scenarios (IDs)		*Core-ABCD1*
Added artifacts		Security Reasoner
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	/
Security mechanisms		/
SPECS applications		/

Table 26. Integration scenario *Core-ABCD2*

With the *Core-ABCD3* integration scenario, the SPECS flow is upgraded with Security Tokens mechanism that protects communication channels among internal SPECS components. Further details are reported in Table 27.

Scenario ID		*Core-ABCD3*
Description		This scenario extends the *Core-ABCD2* scenario by integrating the Security Tokens mechanism (component of the Vertical Layer), which is responsible for the security of interactions among SPECS components.
Base scenarios (IDs)		*Core-ABCD2*
Added artifacts		Security Tokens
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	Security Tokens
Security mechanisms		/
SPECS applications		/

Table 27. Integration scenario *Core-ABCD3*

In the next step we add another component of the Vertical Layer, namely the User Manager component that provides access control functionalities to the SPECS users. The associated *Core-ABCD4* integration scenario is reported in Table 28.

Scenario ID		*Core-ABCD4*
Description		This scenario extends the *Core-ABCD3* scenario by integrating the User Manager component (Vertical Layer), which oversees authentication and authorization functionalities to SPECS users.
Base scenarios (IDs)		*Core-ABCD3*
Added artifacts		User Manager
Core components	**SLAP**	SLA Manager, Service Manager
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	Security Tokens, User Manager
Security mechanisms		/
SPECS applications		/

Table 28. Integration scenario *Core-ABCD4*

The Interoperability Layer component that provides a single access point for all SPECS APIs is integrated in the *Core-ABCD5* scenario. Further details are provided in Table 29.

The next two scenarios integrate the following two artifacts. First, the Credential Service mechanism, that provides functionalities to store and manage SPECS Owner credentials needed to access the external resources (*Core-ABCD6*, Table 30). Second, the Auditing component, that offers logging functionalities to all components of the SPECS framework (*Core-ABCD7*, Table 31).

The final two scenarios integrate the Nmap mechanism that monitors availability of SPECS components (scenario *Core-ABCD8* in Table 32) and the default SPECS Application. We

perform this test to execute the final verification of the correctness of the SPECS flow implementation (scenario *Core-ABCD9* in Table 33).

Scenario ID		*Core-ABCD5*
Description		This scenario extends the *Core-ABCD4* scenario by integrating the Interoperability Layer component (Vertical Layer), which offers the single access point to all SPECS APIs.
Base scenarios (IDs)		*Core-ABCD4*
Added artifacts		Interoperability Layer
Core components	**SLAP**	SLA Manager, Service Manager, Interoperability Layer
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	Security Tokens, User Manager
Security mechanisms		/
SPECS applications		/

Table 29. Integration scenario *Core-ABCD5*

Scenario ID		*Core-ABCD6*
Description		This scenario extends the *Core-ABCD5* scenario by integrating the Credential Service mechanism (Vertical Layer), which stores and manages SPECS Owner credentials to access the external resources.
Base scenarios (IDs)		*Core-ABCD5*
Added artifacts		Credential Service
Core components	**SLAP**	SLA Manager, Service Manager, Interoperability Layer
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	Security Tokens, User Manager, Credential Service
Security mechanisms		/
SPECS applications		/

Table 30. Integration scenario *Core-ABCD6*

Scenario ID		*Core-ABCD7*
Description		This scenario extends the *Core-ABCD6* scenario by integrating the Auditing component (Vertical Layer), which offers logging functionalities to all components of the SPECS framework.
Base scenarios (IDs)		*Core-ABCD6*
Added artifacts		Auditing
Core components	**SLAP**	SLA Manager, Service Manager, Interoperability Layer
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP
	VL	Security Tokens, User Manager, Credential Manager, Auditing
Security mechanisms		/
SPECS applications		/

Table 31. Integration scenario *Core-ABCD7*

Scenario ID		*Core-ABCD8*
Description		This scenario extends the *Core-ABCD7* scenario by integrating the Nmap mechanism (Monitoring module), which monitors availability of internal components of the SPECS framework.
Base scenarios (IDs)		*Core-ABCD7*
Added artifacts		Nmap
Core components	**SLAP**	SLA Manager, Service Manager, Interoperability Layer
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP, Nmap
	VL	Security Tokens, User Manager, Credential Manager, Auditing
Security mechanisms		/
SPECS applications		/

Table 32. Integration scenario *Core-ABCD8*

Scenario ID		*Core-ABCD9*
Description		This scenario extends the *Core-ABCD8* scenario by integrating the default SPECS Application.
Base scenarios (IDs)		*Core-ABCD8*
Added artifacts		Default SPECS Application
Core components	**SLAP**	SLA Manager, Service Manager, Interoperability Layer
	NEG	SLO Manager, Supply Chain Manager, Security Reasoner
	ENF	Planning, Implementation, Diagnosis, RDS
	MON	Event Hub, MoniPoli Filter, Event Aggregator, SLOM Exporter, Event Archiver, CTP, Nmap
	VL	Security Tokens, User Manager, Credential Manager, Auditing
Security mechanisms		/
SPECS applications		Default SPECS Application

Table 33. Integration scenario *Core-ABCD9*

5.2. Integration of SPECS applications

As discussed in Section 4.1 (and further elaborated in deliverable D5.1.2), we have a wide set of SPECS applications:

- Secure Web Container
- Secure Storage
- ngDC
- AAAaaS
- Metric Catalogue
- Security Reasoner

Three of them (Secure Web Container, Secure Storage, ngDC) are built on top of the SPECS framework and SPECS security mechanisms. They offer different cloud services through security SLAs. In this section, we present scenarios that define the process of integration of the SPECS framework and security mechanisms with these applications. The deployment of the Secure Web Container integration scenarios is presented in deliverable D1.5.2. Deployment tests for the Secure Storage and the ngDC applications are reported in dedicated deliverables D5.2.2 and D5.3, respectively.

The AAAaaS SPECS application offers identity management and access control functionalities as-a-Service. It integrates open source tools, adjusted to SPECS needs. The technical aspects of the integration scenarios defined later in this section are further elaborated in the dedicated deliverable D5.4.

The last two scenarios defined in this section describe the integration of components to support development of the Metric Catalogue and Security Reasoner applications. Deployment of these scenarios is further discussed in deliverable D1.5.2.

In the remainder of this section we present integration scenarios defined for the testing of the developed SPECS applications.

With the Secure Web Container application, the End-user can acquire external resources (VMs). SPECS enhances the security level of this service by offering resilience to security incidents through redundancy and diversity (implemented with the WebPool security mechanism).

Additionally, End-users are offered further security features implemented by the SVA, TLS, and DoS security mechanisms.

The first integration scenario (*App-A1*, reported in Table 34) integrates the basic version of the Secure Web Container application, built on top of the SPECS framework, with the WebPool mechanism. Later we gradually integrate remaining mechanisms; the TLS in *App-A2* (see Table 35), the SVA in *App-A3* (see Table 36), and the DoS in *App-A4* (see Table 37).

Scenario ID	*App-A1*
Description	This scenario integrates the Secure Web Container application with the SPECS WebPool security mechanism.
Base scenarios (IDs)	/
Added artifacts	Secure Web Container application, WebPool
Core components	All
Security mechanisms	WebPool
SPECS applications	Secure Web Container

Table 34. Integration scenario *App-A1*

Scenario ID	*App-A2*
Description	This scenario extends the *App-A1* integration scenario with the TLS security mechanism. It integrates the Secure Web Container application with the WebPool and TLS security mechanisms.
Base scenarios (IDs)	*App-A1*
Added artifacts	TLS
Core components	All
Security mechanisms	WebPool, TLS
SPECS applications	Secure Web Container

Table 35. Integration scenario *App-A2*

Scenario ID	*App-A3*
Description	This scenario extends the *App-A2* integration scenario with the SVA security mechanism. It integrates the Secure Web Container application with the WebPool, TLS, and SVA security mechanisms.
Base scenarios (IDs)	*App-A2*
Added artifacts	SVA
Core components	All
Security mechanisms	WebPool, TLS, SVA
SPECS applications	Secure Web Container

Table 36. Integration scenario *App-A3*

Scenario ID	*App-A4*
Description	This scenario extends the *App-A3* integration scenario with the DoS security mechanism. It integrates the Secure Web Container application with the WebPool, TLS, SVA, and DoS security mechanisms.
Base scenarios (IDs)	*App-A3*
Added artifacts	DoS
Core components	All
Security mechanisms	WebPool, TLS, SVA, DoS
SPECS applications	Secure Web Container

Table 37. Integration scenario *App-A4*

With the Secure Storage application, the End-user can acquire cloud storage with external providers. SPECS enhances the security level of this service by providing (i) backup, write-serializability, and read-freshness with the DBB mechanism, and (ii) end-to-end encryption with the E2EE mechanism. We define one scenario for the Secure Storage application; integration of the basic Secure Storage application with the DBB and E2EE security mechanisms (*App-B1*, reported in Table 38).

Scenario ID	*App-B1*
Description	This scenario integrates the Secure Storage application (built on top of the SPECS framework) with the SPECS DBB and E2EE mechanisms.
Base scenarios (IDs)	/
Added artifacts	Secure Storage (application), DBB, E2EE
Core components	All
Security mechanisms	DBB, E2EE
SPECS applications	Secure Storage

Table 38. Integration scenario *App-B1*

The ngDC application is for the use in a private cloud environment in which the CSP hosting SPECS can either offer to End-users its private storage or broker the storage from external resources. The application is built using EMC storage hardware solutions and the ViPR software layer integrated with the SPECS framework. We define one scenario for the development of the ngDC application; integration of the basic ngDC application with the ViPR solution (*App-C1*, reported in Table 39).

Scenario ID	*App-C1*
Description	This scenario integrates the ngDC application with the SPECS DBB and E2EE mechanisms.
Base scenarios (IDs)	/
Added artifacts	ngDC application, ViPR
Core components	All
Security mechanisms	ViPR
SPECS applications	ngDC

Table 39. Integration scenario *App-C1*

The final integration scenarios *App-D1* (Table 40), *App-D2* (Table 41), *App-E1* (Table 42), and *App-F1* (Table 43) present the development of the AAAaaS, Metric Catalogue, and the Security Reasoner applications, respectively. The first application (associated to scenarios *App-D1* and *App-D2*) is built on top of the AAA security mechanism and provides federated identity and access management functionalities enriched with the DBB and E2EE mechanisms. The second application (associated to scenario *App-E1*) is built on top of the Metric Catalogue component, which stores and maintains information related to security metrics. The thirds application (associated to scenario *App-F1*) is built on top of the Security Reasoner component, which ranks SLA Offers by applying assessment algorithms that allow evaluation of the level of security associated to each SLA Offer.

Scenario ID	*App-D1*
Description	This scenario integrates the AAAaaS application with the AAA security mechanism.
Base scenarios (IDs)	/
Added artifacts	AAAaaS application, AAA
Core components	All
Security mechanisms	AAA, ViPR
SPECS applications	AAAaaS

Table 40. Integration scenario *App-D1*

Scenario ID	*App-D2*
Description	This scenario extends the *App-D1* integration scenario with the DBB and E2EE mechanisms.
Base scenarios (IDs)	*App-D1*
Added artifacts	DBB, E2EE
Core components	All
Security mechanisms	AAA, DBB, E2EE, ViPR
SPECS applications	AAAaaS

Table 41. Integration scenario *App-D2*

Scenario ID		*App-E1*
Description		This scenario integrates the Metric Catalogue application with the Metric Catalogue component (part of the SLA Platform).
Base scenarios (IDs)		/
Added artifacts		Metric Catalogue application, Metric Catalogue
Core components	**SLAP**	Metric Catalogue
Security mechanisms		/
SPECS applications		Metric Catalogue

Table 42. Integration scenario *App-E1*

Scenario ID		*App-F1*
Description		This scenario integrates the Security Reasoner application with the Security Reasoner component (part of the Negotiation module).
Base scenarios (IDs)		/
Added artifacts		Security Reasoner application, Security Reasoner (component)
Core components	**NEG**	Security Reasoner
Security mechanisms		/
SPECS applications		Security Reasoner

Table 43. Integration scenario *App-F1*

6. Conclusions

This document introduces a set of integration scenarios to be used as a part of the SPECS integration testing activities. For the sake of clarity, it also gives a brief summary of the architecture of the entire SPECS framework and its behaviour.

The presented integration scenarios have been defined in a way that enables an efficient development of the SPECS applications (i.e. implementation of SPECS validation scenarios) with minimum need for mock-ups. The components of the SPECS framework are gradually integrated to sequentially enable steps determined by the SLA life-cycle.

The complementing deliverable D1.5.2 provides the technical aspects associated to the SPECS integration process (i.e. how we execute integration scenarios, what tools are used, etc.). It specifies the deployment properties for each core integration scenario and the deployment properties for the integration scenarios for the Secure Web Container, the Metric Catalogue, and the Security Reasoner applications. Execution of the integration scenarios for the Secure Storage, ngDC, and the AAAaaS applications is further discussed in deliverables D5.2.2, D5.3, and D5.4, respectively.

Apart from the description of the execution of the defined integration scenarios, the complementing prototype deliverable D1.5.2 also reports results on the security assessment of the framework. Performance and scalability of SPECS components is discussed in dedicated prototype deliverables (for the SLA Platform in D1.5.2, for the Negotiation module in D2.3.2, for the Monitoring module in D3.4.2, and for the Enforcement module in D4.5.3).

7. Bibliography

[1] "Koofr", available online: http://koofr.eu/, last access in January 2016.

[2] "EMC ViPR", available online: http://www.emc.com/vipr, last access in January 2016.

[3] "Chef", available online: http://www.chef.io, last access in January 2016.

Appendix 1. SPECS validation scenarios

This section presents the final version of all validation scenarios defined in the project (in tasks T5.1, T4.2, and T5.4). We group scenarios according to the user stories.

SST.1 Secure_Storage_Selection

General Information			
ID		*SST.1 - Secure_Storage_Selection*	
Version		*2.0*	
User Story		*STO*	*Secure Storage*
Invocation Chain		*IM1-P, IM3*	*Interaction Model 1- SPECS acting the role of Partner*
Scenario Steps			
General Description		*The End-user aims at acquiring a secure storage service from a cloud provider, which fulfils specific security-related requirements. To achieve this, the End-user negotiates the desired features with SPECS.* *In this validation scenario, the desired features are entirely implemented by an external CSP, while SPECS only provides to the End-user the functionalities to search, rank and select a service, which are compliant with her/his requirements. Moreover, in this scenario, the End-user signs an SLA with the selected provider.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*	
	Trigger		
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*	
	Postconditions	*A supply chain compliant to the End-user requirements is built.*	
3	Phase	*SLA Negotiation*	
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*	
	Preconditions	*A secure storage service that fulfils the specific security requirements is known to SPECS.*	
	Trigger		

	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application. The CSPs also add the cost of each service offer.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The service offer is associated with an SLA published by an external CSP.* *The End-user either:* *1. Accepts and signs the SLA offered by the external CSP;* *2. Does not select any SLA Offer from the list and repeats the whole process from step 1 (possibly specifying a different set of requirements);* *3. Does not select any SLA Offer from the list and exits the application.*
	Postconditions	*In case 1 - the signed SLA is stored by SPECS. The End-user is enabled to invoke the desired service on the external CSP with the configuration information included in the SLA.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_4, SLA_5*	

SST.2 Secure_Storage_Brokering_with_Client_Crypto

General Information		
ID	*SST.2 - Secure_Storage_Brokering_with_Client_Crypto*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data.* *To enable this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End encryption security mechanism.* *In this scenario, SPECS also provides monitoring functionalities.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*

	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service, which fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*

	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install, as well as their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7*	

SST.3 Secure_Storage_with_Defined_CSP

General Information		
ID	*SST.3 - Secure_Storage_with_Defined_CSP*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*

Scenario Steps		
General Description		*The End-user aims at storing encrypted data on a known remote cloud provider which offers a Database-as-a-service capability. The End-user asks SPECS for End-to-End Encryption capability, needed to locally encrypt her/his data.* *To enable this service, the End-user also gives SPECS her/his credentials on the chosen provider; SPECS securely manages these credentials and uses them to log into the chosen provider and store the End-user's data.* *In this scenario, SPECS also provides monitoring functionalities.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	*The external CSP offering the Database-as-a-Service chosen by the End-user is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the needs of using a specific CSP as Database-as-a-Service provider and having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, the specific CSP defined by the End-user is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	

	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install as well as their configurations.*
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.* *The credentials of the End-user on the external CSP have been acquired.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module acquires the storage service with the credentials of the End-user on the external CSP and deploys and configures monitoring agents. The SPECS Enforcement module activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	

SLA	*SLA_1, SLA_3, SLA_6, SLA_7*

SST.4 Secure_Storage_Brokering_with_Client_Crypto_Alert

General Information			
ID		*SST.4 - Secure_Storage_Brokering_with_Client_Crypto_alert*	
Version		*2.0*	
User Story		*STO*	*Secure Storage*
Invocation Chain		*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data.* *To enable this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End Encryption security mechanism.* *In this scenario, SPECS also provides monitoring functionalities.* *In this scenario, an alert is raised since the Encryption Server component is detected to be down and, since no data is sent from the End-user during the down time, no violation occurs.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*	
	Trigger		
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*	
	Postconditions	*A supply chain compliant to the End-user requirements is built.*	
3	Phase	*SLA Negotiation*	
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*	

	Preconditions	*A secure storage service that fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*

	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events due to the deviation of some metrics from set thresholds (since the the Encryption Server component is down).*
	Actions	*The SPECS Enforcement module analyses monitoring events and classifies it as an alert. The root cause of the monitoring event is determined (the Encryption server component is detected to be down, but no data has been sent from the End-user during the down time; thus no violation occurs).*
	Postconditions	*A report on the alert and on the root cause of the monitoring event is created.*
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*The SPECS Enforcement module reacts by restarting the component before any encrypted files are sent to the server.*
	Postconditions	*The alert is solved.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_10, SLA_11*	

SST.5 Secure_Storage_Brokering_with_Client_Crypto_Violation

General Information		
ID	*SST.5 - Secure_Storage_Brokering_with_Client_Crypto_violation*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data.* *To achieve this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End Encryption security mechanism.* *In this scenario, SPECS also provides monitoring functionalities.* *In this scenario, a violation is detected since the Encryption Server component is detected to be down.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	

4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*End-user, SPECS Monitoring module, SPECS Enforcement module*

	Preconditions	*The End-user has sent files to encrypt to the server while it is down*
	Trigger	*The SPECS Monitoring module generates monitoring events due to the deviation of some metrics from set thresholds (since the Encryption Server component is down).*
	Actions	*The SPECS Enforcement module analyses monitoring events and detects a violation. The root cause analysis of the monitoring event is determined (the Enforcement module determines that the SLA violation occurred due to the Encryption Server component being down).*
	Postconditions	*A report on the violation and on the root cause of the monitoring event is created.*
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*SPECS notifies the violation to the End-User through the SPECS Application. The SPECS Enforcement module searches for alternatives for the End-user by building new services.*
	Postconditions	*The SLA is no more fulfilled.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_12*	

SWC.1 Secure_Web_Container_Selection

General Information		
ID	*SWC.1 - Secure_Web_Container_Selection*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-P*	*Interaction Model 1- SPECS acting the role of Partner*
Scenario Steps		
General Description	*The End-user aims at acquiring a web container service fulfilling specific security requirements (e.g., availability, resilience to attacks). To enable this service, the End-user negotiates the desired features with SPECS.* *In this validation scenario, the desired features are already provided by a CSP, and SPECS only returns to the End-user the reference to such provider.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user is an expert customer since she/he is able to evaluate each individual metric with respect to her/him own security requirements.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface using the expert interface, in order to enter/specify in a specific way her/his security requirements. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*

	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is offered by at least one external CSP, known to SPECS.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application. The CSPs also add the cost of each service offer.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The service offer is associated with an SLA published by an external CSP. The End-user either:* *1. Accepts and signs the SLA offered by the external CSP;* *2. Does not select any SLA Offer from the list and repeats the whole process from step 1 (possibly specifying a different set of requirements);* *3. Does not select any SLA Offer from the list and exits the application.*
	Postconditions	*In case 1 - the signed SLA is stored by SPECS. The End-user is enabled to invoke the desired service on the external CSP with the configuration information included in the SLA.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users		*U_1 (CSC:User)*
Target services		*TS_4 (Infrastructure as a Service)*
SPECS services		*See Appendix B of D5.1.2*
SLA		*SLA_1, SLA_3. SLA_4, SLA_5*

SWC.2 Secure_Web_Container_Brokering

General Information	
ID	*SWC.2 - Secure_Web_Container_Brokering*

Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*The End-user aims at acquiring a web container service fulfilling specific security requirements (e.g., availability, resilience to attacks). To enable this service, the End-user negotiates the desired security features with SPECS.* *In this validation scenario, the desired features are already provided by a CSP, but SPECS acts as a broker by acquiring the resources on behalf of the End-user (registered on SPECS) and by setting up some monitoring functionalities in order to monitor the fulfilment of the SLA.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*	
	Trigger		
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls- She/he also specifies the desired metrics and sets the related SLOs.* *The End-user accesses the Security Metric Catalogue in order to have additional and detailed information about the specific chosen metrics.*	
	Postconditions	*A supply chain compliant to the End-user requirements is built.*	
3	Phase	*SLA Negotiation*	
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*	
	Preconditions	*A web container, which fulfils the specific security requirements, is offered by at least one external CSP, known to SPECS.*	
	Trigger		
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*	
	Postconditions		
4	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SLA Platform*	
	Preconditions	*The End-user shall be logged on SPECS.*	

	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_4 (Infrastructure as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	

SLA	*SLA_1, SLA_3, SLA_6, SLA_7*

SWC.3 Secure_Web_Container_TLS_Enhanced

General Information			
ID		*SWC.3 - Secure_Web_Container_TLS_enhanced*	
Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires the adoption of the TLS protocol to protect the network communications, and of the DoS detection and mitigation features. To enable this service, the End-user negotiates the desired features with SPECS.* *In this validation scenario, a bare web container is offered by a CSP, while the TLS protocol and the DoS detection and mitigation features are provided by SPECS through the activation of proper mechanisms. SPECS acquires the resources on behalf of the End-user (registered on SPECS), deploys and activates the TLS and DoS related mechanisms, and sets up elated monitoring functionalities. In this scenario, an alert regarding a DoS attack is generated, and SPECS reacts by activating proper mitigation strategies. The scenario ends without any other alert.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*	
	Trigger		
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*	
	Postconditions	*A supply chain compliant to the End-user requirements is built.*	
3	Phase	*SLA Negotiation*	
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*	
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS.* *An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and the TLS and DoS detection and mitigation tools are offered as SPECS security mechanisms.*	
	Trigger		

	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. TLS, DoS detection and DoS mitigation components are identified among SPECS Enforcement security components.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP, while the TLS, DoS detection and DoS mitigation are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*

	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events related to detection of DoS attack by the DoS Monitoring component.*
	Actions	*The SPECS Enforcement module analyses monitoring events and, relying upon the attack classification functionalities provided by the SPECS DoS Mitigation component, classifies it as an alert.*
	Postconditions	
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	*Some mitigation strategies are available.*
	Trigger	*An alert has been detected.*
	Actions	*The SPECS Enforcement module reacts by activating proper mitigation strategies, defined by the SPECS DoS Mitigation component.*
	Postconditions	*The alert is solved and the SLA is completed since neither alerts nor violations occur.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_4 (Infrastructure as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_10, SLA_11, SLA_8*	

SWC.4 Secure_Web_Container_TLS_Enhanced_Alert

General Information		
ID	*SWC.4 - Secure_Web_Container_SVA_enhanced_alert*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires the adoption of a Software Vulnerability Assessment (SVA) tool to protect the web container environment. To enable this service, the End-User negotiates the desired features with SPECS.* *In this validation scenario, the bare web container is offered by a CSP, while the SVA tools are provided by SPECS. SPECS acquires the resources on behalf of the End-user (registered on SPECS), deploys and activates the needed SVA agents and sets-up related monitoring functionalities.* *In this scenario, an alert is generated due to the existence of some critical vulnerability in the installed software. SPECS reacts by updating the software version to remove the vulnerability. The scenario ends without any other alert.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS.* *An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and SVA agents are offered as SPECS security mechanisms.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. SVA agents are identified among SPECS Enforcement security components.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	

	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP, while the SVA agents are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA (including the installation of SVA agents on the plain VM). The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events related to the deviation of some metrics from set thresholds (e.g., number of exposed vulnerabilities).*

	Actions	*The SPECS Enforcement module makes an analysis of monitoring events and classifies them as an alert.*
	Postconditions	
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	*The new version of the vulnerable software is available.*
	Trigger	*An alert regarding a vulnerability threat has been detected*
	Actions	*The SPECS Enforcement module reacts by activating the available redressing technique (it checks the presence of new versions, and updates the vulnerable software).*
	Postconditions	*The alert is solved.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_4 (Infrastructure as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_10, SLA_11*	

SWC.5 Secure_Web_Container_TLS_SVA_Enhanced_Violation

General Information		
ID	*SWC.5 - Secure_Web_Container_TLS_SVA_enhanced_violation*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a web container from an Infrastructure-as-a-Service CSP, represented by a VM hosting the web server, which fulfils specific security-related requirements. In particular, the End-user requires the adoption of Software Vulnerability Assessment (SVA) tools to protect the Web Server environment. To enable this service, the End-user negotiates the desired features with the SPECS.* *In this validation scenario, the VM (without SVA) is provided by an Infrastructure-as-a-Service CSP, while the SVA agents are installed by SPECS. SPECS acquires the resources on behalf of the End-user (registered on SPECS), it adds the SVA agents, and sets up some monitoring functionalities in order to detect the presence of exposed vulnerabilities.* *This scenario includes the raising of an alert regarding a vulnerability assessment report, which corresponds to a violation of the agreed SLA. SPECS reacts by renegotiating the SLA; the End-user asks for the adoption of the TLS protocol to protect the Web Server communications. The renegotiated SLA is hence signed and properly monitored by SPECS.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	

	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS.* *An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and SVA agents are offered as SPECS security mechanisms.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. SVA agents are identified among SPECS Enforcement security components.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offer,s which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the SVA agents are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*

	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA (including the installation of SVA agents on the plain VM). The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events related to the deviation of some metrics from set thresholds (e.g., number of exposed vulnerabilities).*
	Actions	*The SPECS Enforcement module makes an analysis of monitoring events and classifies them as a violation.*
	Postconditions	
10	Phase	*SLA Remediation*
	Actor	*SPECS Application, SPECS Enforcement module*
	Preconditions	*No remedies can be applied by SPECS; renegotiation is needed.*
	Trigger	*A violation of the signed SLA has been detected.*
	Actions	*SPECS notifies the violation to the End-User through the SPECS Application. The SPECS Enforcement module searches for alternatives for the End-user by building new services.*
	Postconditions	*The SLA is no more fulfilled*
11	Phase	*Renegotiation*
	Actor	*End-user, SPECS Application, SPECS Negotiation module*
	Preconditions	
	Trigger	

	Actions	*The End-user asks for the adoption of Transport Layer Security (TLS) protocol to protect the Web Server communications. The renegotiation follows the same activities described in steps 1 to 4.*
	Postconditions	*The renegotiated SLA is signed.*
12	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available.*
	Trigger	
	Actions	*The implementation of the SLA follows the same activities described in steps 5 to 7.*
	Postconditions	
13	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	*The monitoring policy has been updated to include thresholds related to the SLA.*
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_4 (Infrastructure as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	SLA_1, SLA_3, SLA_6, SLA_7, SLA_13, SLA_14, SLA_17, SLA_19	

SWC.6 Secure_Web_Container_TLS_Multitenancy

General Information		
ID	*SWC.6 - Secure_Web_Container_TLS_multitenancy*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*Two End-users aim at acquiring different web container services fulfilling specific security requirements. In addition, both End-users require the adoption of the TLS protocol to protect the communications of Web Servers. To enable this service, the first End-user negotiates the desired features with SPECS. The VM (without TLS) is provided by a CSP, while the TLS protocol is added by SPECS by setting up the appropriate resources (e.g., reverse proxy).* *The second End-user negotiates the desired features with SPECS. A different VM (without TLS) is provided by a CSP (either the same or a different one) while the TLS protocol is added by SPECS reusing, for scalability purposes, the same resources configured for the first End-user.* *This validation scenario considers the multi-tenancy in the usage of shared resources between End-users.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user (first), SPECS application, SPECS Negotiation module*

	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The first End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user (first), SPECS application*
	Preconditions	
	Trigger	
	Actions	*The first End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS.* *An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and the TLS and DoS detection and mitigation tools are offered as SPECS security mechanisms.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. TLS, DoS detection and DoS mitigation components are identified among SPECS Enforcement security components.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user (first), SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the TLS, DoS detection and DoS mitigation are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	

	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate, and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Negotiation*
	Actor	*End-user (second), SPECS application, SPECS Negotiation module, SPECS Enforcement module, SLA Platform*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.* *The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The second End-user accesses the SPECS application interface, asking for a secure web container which fulfils the specific security requirements.* *The negotiation follows the same activities described in steps 1 to 4.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
10	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform, SPECS Monitoring Module*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform*
	Trigger	

	Actions	*The SPECS application invokes the SPECS Enforcement module which prepares and implements the plan that implements the signed SLA. It configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.* *The implementation of the SLA follows the same activities described in steps 5 to 7 but the TLS protocol is added by reusing, for scalability purposes, the same resources adopted for the first End-user.*
	Postconditions	
11	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates the monitoring policies.*
	Postconditions	*The signed SLA is fulfilled since neither alerts nor violations occur.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_4 (Infrastructure as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	SLA_1, SLA_3, SLA_6, SLA_7	

SWC.7 Secure_Web_Container_Web_Pool_Replication_Enhanced_Alert

General Information		
ID	*SWC.7 - Secure_Web_Container_Web_Pool_Replication_enhanced_alert*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires a specific level of redundancy and session persistence among web container replicas. To enable this service, the End-User negotiates the desired features with SPECS.* *In this validation scenario, the bare web containers are offered by a CSP, while the redundancy features with session persistence among replicas is provided by SPECS through the WebPool mechanism. SPECS acquires the resources on behalf of the End-user (registered on SPECS), adds the WebPool mechanism's components, and sets-up proper resources to handle HTTP requests through proxy functionalities, in order to forward the requests to one of the available web container replicas. In this scenario, the proxy functionality is added, by SPECS, on a dedicated VM.* *In this scenario, an alert is generated because one of the replicas slows down, thus risking compromising the desired level of redundancy. SPECS reacts by isolating the replica and by restarting it. The scenario ends without any other alert*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*

	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. In particular, the End-user requires the adoption of a web pool mechanism to ensure session persistence among web container replicas*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*An Infrastructure-as-a-Service provider that offers VMs that fulfil the specific requirements is known to SPECS. The web pool mechanism is offered as a SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified; the web pool mechanism is identified among SPECS security mechanisms.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the web pool mechanism is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform, SPECS Monitoring Module*
	Preconditions	
	Trigger	

	Actions	*SPECS acquires the VMs on behalf of the End-user on the external CSP and adds the web pool components. SPECS also sets up proper resources to handle HTTP request through proxying functionality in order to forward the requests to one of the available web containers. SPECS launches the related monitoring services.*
	Postconditions	
6	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates the monitoring policies.*
	Postconditions	
7	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	*A redressing technique can be adopted according to the signed SLA, and is available as SPECS security mechanisms.*
	Trigger	*An alert regarding the level of redundancy is raised by the enforcement diagnosis, after the notification of a monitoring event by the SPECS Monitoring module.*
	Actions	*SPECS updates the implemented forwarding policy (redressing technique) and removes the affected web container from the pool of available web containers*
	Postconditions	*The discovered alert is solved and no more alerts are generated.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_4 (Infrastructure as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_8, SLA_9, SLA_10, SLA_11*	

SWC.8 Secure_Web_Container_Web_Pool_Replication_Enhanced_Violation

General Information		
ID	*SWC.8 - Secure_Web_Container_Web_Pool_Replication_enhanced_violation*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*An End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires a specific level of redundancy and session persistence among web container replicas. To achieve this service, the End-User negotiates the desired features with SPECS.* *In this validation scenario, the bare web containers are offered by a CSP, while the redundancy features with session persistence among replicas is provided by SPECS through the WebPool mechanism. SPECS acquires the resources on behalf of the End-user (registered on SPECS), adds the WebPool mechanism's components, and sets-up proper resources to handle HTTP requests through proxy functionalities, in order to forward the requests to one of the available web container replicas. In this scenario, the proxy functionality is added, by SPECS, on a dedicated VM.* *In this scenario, an alert is generated because one of the replicas goes down, thus compromising the desired level of redundancy. SPECS reacts by isolating the replica and by removing it from the pool of replicas. The SLA is violated since the level of redundancy is not preserved.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features (in particular, the End-user requires the adoption of a web pool mechanism to ensure session persistence among web container replicas) by selecting the capabilities she/he is interested in and by specifying the related security controls. The End-user also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*An Infrastructure-as-a-Service provider that offers VMs that fulfil the specific requirements, is known to SPECS. The web pool mechanism is offered as a SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified; the web pool mechanism is identified among SPECS security mechanisms.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	

4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the web pool mechanism is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform, SPECS Monitoring Module*
	Preconditions	
	Trigger	
	Actions	*SPECS acquires the VMs on behalf of the End-user on the external CSP and adds the web pool components, and sets up proper resources to handle HTTP request through proxying functionality in order to forward the requests to one of the available web containers. SPECS launches the related monitoring services.*
	Postconditions	
6	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates the monitoring policies.*
	Postconditions	
7	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*An alert regarding a vulnerability threat on a web container is raised by the enforcement diagnosis, after the notification of a monitoring event by the SPECS Monitoring module.*
	Actions	*SPECS removes the affected web container from the pool of available web containers.*
	Postconditions	
8	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	*A violation of the signed SLA is detected by the enforcement diagnosis.*
	Actions	*SPECS notifies the violation to the End-user.*
	Postconditions	*The SLA is no more fulfilled*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	

Target services	*TS_4 (Infrastructure as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_12*

NGDC.1 Data_Center_Bursting_for_Storage_Resources

General Information			
ID		*NGDC.1 - Data_Center_Bursting_for_Storage_Resources*	
Version		*1.1*	
User Story		*ngDC*	*Next Generation Data Center*
Invocation Chain		*IM2-CSP*	*Interaction Model 2 - SPECS acting in the role of CSP*
Scenario Steps			
General Description		*A CSP hosting its own next generation Data Center (ngDC), acting within a Cloud Service Customer (CSC) role, aims at using the SPECS framework to perform Cloud bursting in order to extend its Secure Storage as a Service (SStaaS) capabilities. This occurs during a period of increased storage demand, which exceeds the CSP's own ngDC storage capabilities.* *The CPS considers its storage as first class storage due to the fine grained control it has over all the security parameters. The CSP will allocate the first class storage to End-users that do not require high-security capabilities enabled. Otherwise, it will allocate storage acquired from an external provider through SPECS. The entire process is transparent to the End-user.* *Note, while a CSP acquiring storage resources from an external 3rd party CSP is typically defined as an End-user, it is not in the context of a SPECS defined in this way. That is, the CSP intends to resell its acquired external storage resources and so it is considered a CSC (in the context of SPECS). For ease of exposition 'customer' is used as a common reference to either a CSC or End-user of the CSP hosting the ngDC.*	
Steps			
1	Phase	*Negotiation*	
	Actor	*CSC (CSP is acting within a CSC role)*	
	Preconditions	*The CSC monitors the current state of its ngDC in terms of its on-premise storage resources.*	
	Trigger	*Capacity threshold reached.*	
	Actions	*The CSC asks its locally hosted SPECS for an external CSP offering SStaaS, which fulfils its specific security requirements. These security requirements might be based on either or both the CSC's own security requirements or that of the CSC's own customers. Examples of security requirements are the data geo-location, the Drive type, RAID level, etc.*	
	Postconditions		
2	Phase	*Negotiation*	
	Actor	*SPECS Negotiation module*	
	Preconditions	*An external CSP that fulfils the specific secure storage requirements must already be present within the locally hosted CSC's SPECS SLA Repository.*	
	Trigger		
	Actions	*SPECS searches for possible supply chains compliant with the specified secure storage requirements, evaluates if the external CSP fulfils the End-User requirements SPECS will allocate directly the resource, otherwise it will allocate resource on the local storage platform.*	

	Postconditions	
3	Phase	*Negotiation*
	Actor	*CSC*
	Preconditions	
	Trigger	
	Actions	*The CSC selects one supply chain from the retrieved list and signs the SLA with the external CSPs that form part of the SPECS supply chain.*
	Postconditions	
Graphical Model		
Coverage Information		
Users	*U_1 (CSC:user)*	
Target services	*TS_3 (Data Storage as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA Lifecycle	*SLA_1, SLA_3, SLA_4, SLA_5, SLA_6*	

NGDC.3 Data_Center_Storage_Selection

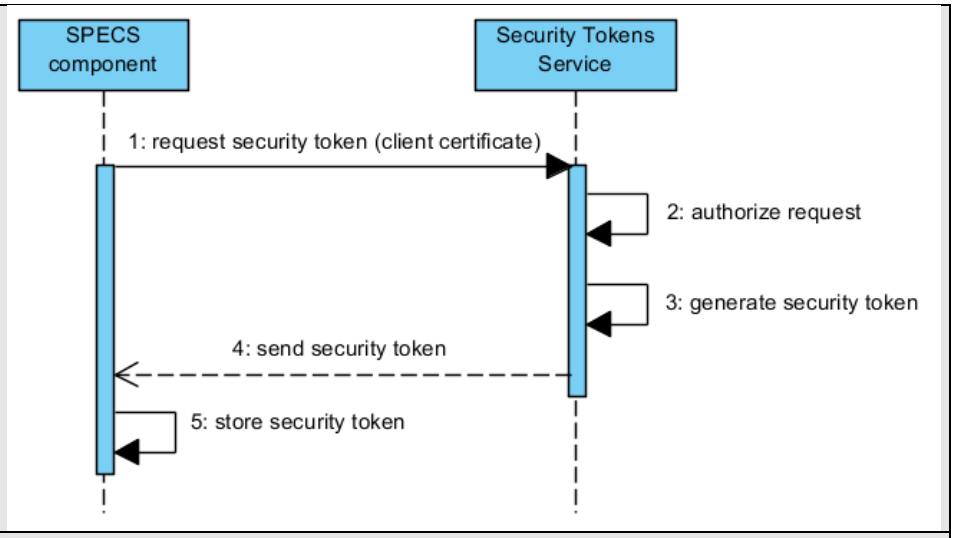
General Information		
ID	*NGDC.3 – Data_Center_Storage_Selection*	
Version	*1.0*	
User Story	*NgDC*	*Next Generation Data Center*
Invocation Chain	*IM2-CSP*	*Interaction Model 2- SPECS acting the role of CSP*
Scenario Steps		
General Description	*A CSP owning SPECS and hosting its own ngDC, acting within a CSC role, aims at using the SPECS framework to perform Cloud bursting in order to extend its Secure SStaaS capabilities. This occurs during a period of increased storage demand, which exceeds the CSP's own ngDC storage capabilities.* *In this validation scenario, the desired features are entirely implemented by an external CSP, while SPECS only offers the End-user the ability to search, rank and select a service, which is compliant to her/his requirements. Moreover, in this scenario, SPECS supports the End-user in signing an SLA with the selected provider.*	
Steps		
1	Phase	*SLA Negotiation*
	Actor	*CSC (CSP is acting within a CSC role)*
	Preconditions	*The End-user has good security knowledge; she/he is able to express qualitatively requirements at a low-level of abstraction.*
	Trigger	
	Actions	*The CSC asks its locally hosted SPECS for an external CSP offering SStaaS, which fulfils its specific security requirements. These security requirements might be based on either or both the CSC's own security requirements or that of the CSC's own customers. Examples of security requirements are the data geo-location, the Drive type, RAID level, etc.* *The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*

	Actor	*CSC, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. The End-user also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is known to SPECS.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application. The CSPs also add the cost of each service offer.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The service offer is associated with an SLA published by an external CSP. The End-user either:* *1. Accepts and signs the SLA offered by the external CSP;* *2. Does not select any SLA Offer from the list and repeats the whole process from step 1 (possibly specifying a different set of requirements);* *3. Does not select any SLA Offer from the list and exits the application.*
	Postconditions	*In case 1 - the signed SLA is stored by SPECS. The End-user is enabled to invoke the desired service on the external CSP with the configuration information included in the SLA.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3. SLA_4, SLA_5*	

CRO.1 Security_Tokens_Acquisition

General Information		
ID	*CRO.1 - Security_Tokens_Acquisition*	
Version	*2.0*	
User Story	*n.d.*	

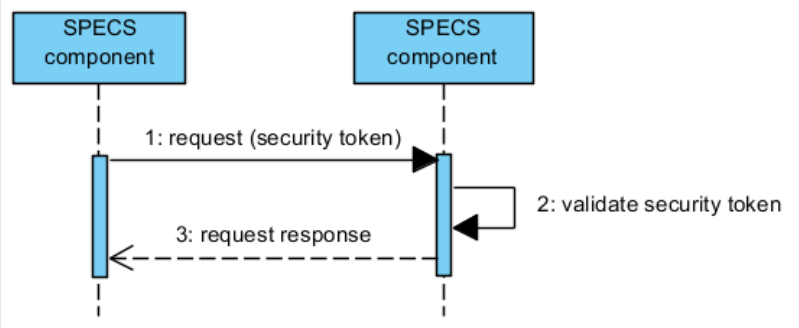
Invocation Chain		*n.d.*	
Scenario Steps			
General Description		*Each invocation of a SPECS component API must be authenticated and authorized through a proper mechanism based on security tokens.* *In this validation scenario, the acquisition of a security token is shown.*	
Steps			
1	Phase	*Token Acquisition*	
	Actor	*SPECS component*	
	Preconditions	*The component has a valid client certificate.*	
	Trigger	*The SPECS component would like to call some SPECS service.*	
	Actions	*The SPECS component sends a request to the Security Tokens Service and asks for a security token. It authenticates with its client certificate.*	
	Postconditions		
2	Phase	*Token Acquisition*	
	Actor	*Security Tokens Service*	
	Preconditions	*The SPECS component authenticated with valid a client certificate.*	
	Trigger		
	Actions	*The Security Tokens Service authorizes the request for a security token.*	
	Postconditions		
3	Phase	*Token Acquisition*	
	Actor	*Security Tokens Service*	
	Preconditions	*The client is authorized to request a security token.*	
	Trigger		
	Actions	*The Security Tokens Service generates a security token containing the subject and list of services the token is eligible to access and returns it to the client.*	
	Postconditions		
4	Phase	*Token Acquisition*	
	Actor	*SPECS component*	
	Preconditions	*The request for a security token was granted.*	
	Trigger		
	Actions	*The SPECS component stores the security token to the token vault noting the token's expiration time.*	
	Postconditions		
4	Phase	*Token Acquisition*	
	Actor	*SPECS component*	
	Preconditions	*The SPECS component has a valid security token*	
	Trigger		
	Actions	*The SPECS component calls some SPECS service, attaching the security token to the request. When making REST API calls, the security token is put in the HTTP header named X-AUTH-TOKEN. All communication among components is encrypted by using secure HTTPS connection.*	
	Phase		

Graphical Model	

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*Not Applicable.*
SPECS services	*See Appendix B of D5.1.2*
SLA	*Not Applicable.*

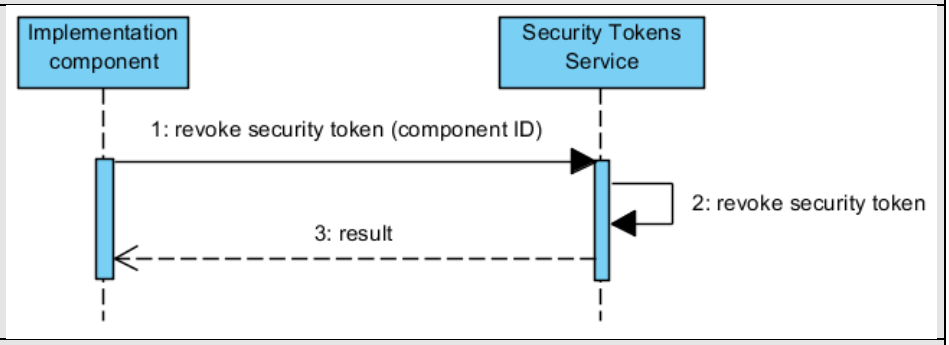
CRO.2 Security_Tokens_Validation

General Information			
ID		*CRO.2 - Security_Tokens_Validation*	
Version		*2.0*	
User Story		*n.d.*	
Invocation Chain		*n.d.*	
Scenario Steps			
General Description		*Each invocation of a SPECS component API must be authenticated and authorized through a proper mechanism based on security tokens. In this validation scenario, the validation of a security token is shown.*	
Steps			
1	Phase	*Token Validation*	
	Actor	*SPECS component*	
	Preconditions	*The SPECS component has a valid security token.*	
	Trigger		
	Actions	*The SPECS component calls another SPECS component, attaching the security token to the request.*	
	Postconditions		
2	Phase	*Token Validation*	
	Actor	*SPECS component*	
	Preconditions		
	Trigger		
	Actions	*The SPECS component uses the security-tokens-client library to validate and decode the token.*	
	Postconditions		

3	Phase	*Token Validation*
	Actor	*SPECS component*
	Preconditions	*The security token is valid.*
	Trigger	
	Actions	*The SPECS component authorizes the request based on the information in the security token using XACML authorization engine.*
	Postconditions	
Graphical Model		
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*Not Applicable.*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*Not Applicable.*	

CRO.3 Security_Tokens_Revocation

General Information		
ID	*CRO.3 - Security_Tokens_Revocation*	
Version	*2.0*	
User Story	*n.d.*	
Invocation Chain	*n.d.*	
Scenario Steps		
General Description	*In this validation scenario, the revocation of a security token is shown.*	
Steps		
1	Phase	*Token Revocation*
	Actor	*Implementation component*
	Preconditions	
	Trigger	*The SLA is terminated.*
	Actions	*The Implementation component sends request to the Security Tokens Service to revoke the security tokens issued to a specific SPECS component. The Implementation component is authenticated by its certificate.*
	Postconditions	
2	Phase	*Token Revocation*
	Actor	*Security Tokens Service*
	Preconditions	*The revoke request is authenticated and authorized.*
	Trigger	
	Actions	*The Security Tokens Service finds the tokens issued to the specified SPECS component, marks them as revoked and adds them to the token revocation list.*

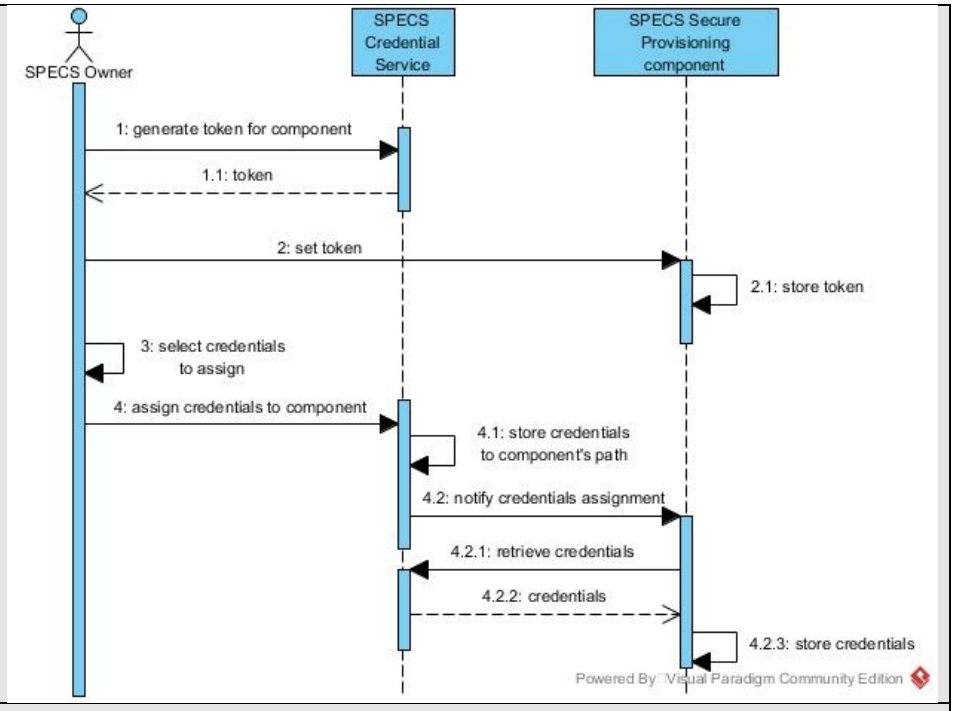
	Postconditions	
3	Phase	*Token Revocation*
	Actor	*All SPECS components*
	Preconditions	
	Trigger	*Periodical update of the token revocation list*
	Actions	*SPECS components periodically pull delta token revocation list and update local token revocation list cache. The revoked tokens are propagated to the local token revocation list caches.*
	Postconditions	
4	Phase	*Token Revocation*
	Actor	*Blocked component*
	Preconditions	*The revoked tokens were propagated to local token revocation list caches.*
	Trigger	
	Actions	*The blocked component calls some other SPECS component with security token attached. The target component validates the token, finds out that the token is on the revocation list and denies the request.*
	Postconditions	
Graphical Model		

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*Not Applicable.*
SPECS services	*See Appendix B of D5.1.2*
SLA	*Not Applicable.*

CRO.4 Credential_Management

General Information		
ID	*CRO.4 - Credential_Management*	
Version	*3.0*	
User Story	*n.d.*	
Invocation Chain	*n.d.*	
Scenario Steps		

General Description		*The SPECS Credential Management service handles the authentication/authorization requests coming from non-human clients on behalf of End-users and that are targeted to a CSP.* *In this scenario, we report all the steps needed to assign, to the Secure Provisioning component, the credentials to access an external CSP for resource acquisition.* *The reported interactions involve the SPECS Owner, the Secure Provisioning component and the Credential Service, which exposes a web interface to the SPECS Owner to manage credentials (i.e. the Credential Management Application). The Credential Service includes a Credential Client and a Credential Manager: the Credential Client acts as an interface to the Credential Manager, which offers the functionalities to manage credentials. All communications with the Credential Manager must be authenticated by means of a token.*
Steps		
1	Phase	*Configuration of the component to use the Credential Service*
	Actor	*SPECS Owner, SPECS Secure Provisioning component, SPECS Credential Service*
	Preconditions	*The SPECS Credential Service has been properly installed and initialized.* *The SPECS Owner has a valid token to communicate with the Credential Service.* *The Secure Provisioning component has been configured with a Credential Client at set-up.*
	Trigger	
	Actions	*The SPECS Owner accesses the web interface of the SPECS Credential Service and generates a token for the Secure Provisioning component. The token is passed to the component via the API exposed by the related Credential Client.*
	Postconditions	*The SPECS Secure Provisioning component has a valid token to communicate with the SPECS Credential Service*
2	Phase	*Credentials assignment*
	Actor	*SPECS Owner, SPECS Credential Service component, SPECS Secure Provisioning component*
	Preconditions	*The SPECS Owner has a valid token to communicate with the Credential Service.* *The credentials to access the CSP have been previously stored in the Credential Service by the SPECS Owner.*
	Trigger	
	Actions	*The SPECS Owner accesses the web interface of the SPECS Credential Service, selects a set of credentials and assigns them to the Secure Provisioning component.*
	Postconditions	*The credentials are copied to the component's path.* *The Secure Provisioning component is notified about the assignment via the API exposed by the related Credential Client.*
3	Phase	*Credentials retrieving*
	Actor	*SPECS Secure Provisioning component, SPECS Credential Service*
	Preconditions	*The SPECS Secure Provisioning component has a valid token to communicate with the SPECS Credential Service.*
	Trigger	
	Actions	*The SPECS Secure Provisioning component retrieves the credentials from the path and saves them locally.*
	Postconditions	

Graphical Model	

Coverage Information	
Users	*n.d.*
Target services	*Not Applicable.*
SPECS services	
SLA	*Not Applicable.*

CRO.5 User_Direct_Registration

General Information			
ID		*CRO.5 - User_Direct_Registration*	
Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*Some SPECS services are offered to the registered End-users. In this validation scenario, the registration is performed manually by the End-user, by inserting her/his personal information through the compilation of proper forms. The process ends with SPECS adding the registered user to the user list.*	
Steps			
1	Phase	*Registration*	
	Actor	*End-user*	
	Preconditions		
	Trigger		
	Actions	*The End-user fills the registration form with her/his personal information and submits it.*	
	Postconditions		
2	Phase	*Registration*	
	Actor	*SPECS AAA component*	

	Preconditions	*The End-user is not registered yet on SPECS*
	Trigger	
	Actions	*The SPECS user repository is updated by adding a new entry with the information of the End-user.*
	Postconditions	*The End-user's information is stored in the SPECS user repository.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*Not Applicable*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*Not Applicable*	

CRO.6 User_Registration_External_Account

General Information			
ID		*CRO.6 - User_Registration_External_Account*	
Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*Some SPECS services are offered to the registered End-users. In this validation scenario, the registration is performed by using a pre-existing external account (e.g., from an account of a social network or from an LDAP entry). The process ends with SPECS adding the registered user to the user list and linking it with the external account.*	
Steps			
1	Phase	*Registration*	
	Actor	*End-user*	
	Preconditions		
	Trigger		
	Actions	*The End-user submits an authentication request (through, for example, a SAML request) to the SPECS AAA component.*	
	Postconditions		
2	Phase	*Registration*	
	Actor	*End-user*	
	Preconditions	*The End-user has a valid account on the selected external authentication source.*	
	Trigger		
	Actions	*The End-user selects the external authentication source and performs the login with the credentials of the external account, retrieving her/his personal information.*	
	Postconditions		
3.1	Phase	*Registration*	
	Actor	*SPECS AAA component*	
	Preconditions	*The End-user is not registered yet on SPECS.*	
	Trigger		

	Actions	*The SPECS user repository is updated by adding a new entry with the information of the End-User from the external authentication source. A link to the external account is also created.*
	Postconditions	*The End-user's information, along with the link to the external account, is stored in the SPECS user repository.*
3.2	Phase	*Registration*
	Actor	*SPECS AAA component*
	Preconditions	*The End-user is already registered on SPECS, and the link with the external account has not yet been specified.*
	Trigger	
	Actions	*The link with the external account is created for the user entry.*
	Postconditions	*The link to the external account is stored in the SPECS user repository.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*Not Applicable*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*Not Applicable*	

CRO.7 User_Authentication_External_Account

General Information			
ID		*CRO.7 - User_Authentication_External_Account*	
Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*Some SPECS services are offered to the authenticated End-users. In this validation scenario, the authentication is performed by using a pre-existing external account (e.g., social accounts as Facebook, Twitter, or from an LDAP entry). When the user chooses to authenticate through an external source, SPECS checks that the external account is associated with a valid SPECS account. In this case, the user is authenticated. Otherwise SPECS asks if she/he wants to associate the external account to her/his existing SPECS account. In this latter case, the End-user must be preliminary authenticated on SPECS in order to prove the ownership of the SPECS account.*	
Steps			
1	Phase	*Authentication*	
	Actor	*End-user*	
	Preconditions		
	Trigger		
	Actions	*The End-user submits an authentication request (through, for example, an SAML request) to the SPECS AAA component.*	
	Postconditions		
2	Phase	*Authentication*	
	Actor	*End-user*	
	Preconditions	*The End-user has a valid account on the selected external authentication source.*	

	Trigger	
	Actions	*The End-user selects the external authentication source and performs the login with the credentials of the external account, retrieving her/his personal information.*
	Postconditions	*The End-user is authenticated on the external authentication source.*
3.1	Phase	*Authentication*
	Actor	*SPECS AAA component*
	Preconditions	*A SPECS account exists for the End-user. The SPECS account is already linked to the external account.*
	Trigger	
	Actions	*SPECS authenticates the End-user.*
	Postconditions	*The End-user is authenticated on SPECS.*
3.2	Phase	*Authentication*
	Actor	*SPECS AAA component*
	Preconditions	*A SPECS account exists for the End-user. The SPECS account is not yet linked to the external account.*
	Trigger	
	Actions	*SPECS asks the End-user to associate the external account to her/his existing SPECS account.*
	Postconditions	
4.2	Phase	*Authentication*
	Actor	*End-user*
	Preconditions	
	Trigger	
	Actions	*The End-user logs into SPECS with the credentials of the SPECS account.*
	Postconditions	*The End-user is authenticated on the external authentication source.*
5.2	Phase	*Authentication*
	Actor	*SPECS AAA component*
	Preconditions	
	Trigger	
	Actions	*The link with the external account is created for the user entry and SPECS authenticates the End-user.*
	Postconditions	*The link to the external account is stored in the SPECS user repository, and the End-user is authenticated on SPECS.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1(CSC:User)*
Target services	*Not Applicable*
SPECS services	*See Appendix B of D5.1.2*
SLA	*Not Applicable*

CRO.8 Metric_Definition

General Information			
ID		*CRO.8 - Metric_Definition*	
Version		*1.0*	
User Story		*n.d.*	
Invocation Chain		*n.d.*	
Scenario Steps			
General Description		*A SPECS user can easily manage a catalogue of security metrics and can also define her/his own security metric. In this scenario, the definition of a new security metric is shown.*	
Steps			
1	Phase	*Retrieve Metric*	
	Actor	*End-user, Security Metric Catalogue*	
	Preconditions	*The End-user shall be logged on SPECS*	
	Trigger		
	Actions	*The End-user accesses the section of SPECS in which the metric catalogue is stored. She/he finds the set of stored metrics and retrieves needed information in a structured way.*	
	Postconditions		
2	Phase	*Store Metric*	
	Actor	*End-user, Security Metric Catalogue*	
	Preconditions		
	Trigger		
	Actions	*The End-user compiles a form to define a new metric. Specifically, she/he chooses the type of the metric and compiles the appropriate fields. The End-user asks for the storing of the defined metrics.*	
	Postconditions	*The defined metric is added in the Metric Catalogue*	
3	Phase	*Store Metric*	
	Actor	*End-user, Security Metric Catalogue*	
	Preconditions		
	Trigger		
	Actions	*The End-user decides to update an already defined metric, by selecting the specific metric she/he wants to update. The chosen metric is shown in a structured way by the Security Metric Catalogue and the End-user can easily update the appropriate fields. The End-user asks for the storing of the updates.*	
	Postconditions	*The metric is updated in the Metric Catalogue*	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*	
Coverage Information			
Users		*U_1(CSC:User)*	
Target services		*Not Applicable*	
SPECS services		*See Appendix B of D5.1.2*	
SLA		*Not Applicable*	

CRO.9 Security_Mechanism_Development

General Information			
ID		*CRO.9 - Security_Mechanism_Development*	
Version		*1.0*	
User Story		*n.d.*	
Invocation Chain		*n.d.*	
Scenario Steps			
General Description		*A SPECS developer aims at developing a new SPECS security mechanism and integrating it into the SPECS framework. In this scenario, the development of a new security mechanisms and its integration into the SPECS framework is shown. Commercial-off-the-Shelf components are used.*	
Steps			
1	Phase	*Define Services*	
	Actor	*SPECS developer*	
	Preconditions		
	Trigger		
	Actions	*The SPECS developer defines the security properties that the security mechanism she/he wants to develop is able to grant, and the types of services to which the mechanism can be applied. Specifically, she/he identifies the security capabilities enforced by the mechanism and the associated security grants. She/he also defines the remediation process associated with the developed security mechanism.*	
	Postconditions		
2	Phase	*Define Mechanism Architecture*	
	Actor	*SPECS developer*	
	Preconditions		
	Trigger		
	Actions	*The SPECS developer identifies concretely the technologies and the solutions to be implemented through Chef recipes. Specifically, she/he maps each security metric to one basic measurement with which the system can identify possible violations. Each basic measurement is associated to at least one additional measurement.*	
	Postconditions		
3	Phase	*Define Remediation Process, RDS SPECS component*	
	Actor	*SPECS developer*	
	Preconditions		
	Trigger		
	Actions	*The SPECS developer identifies the set of recipes that RDS SPECS component will use to automate the SLA remediation.*	
	Postconditions		
4	Phase	*Prepare Mechanism Metadata*	
	Actor	*SPECS developer, SPECS SLA Platform*	
	Preconditions		
	Trigger		
	Actions	*The SPECS developer prepares the description of the mechanism behaviours, according to the SPECS security mechanism metadata. The developed description is stored in the SLA Platform in order to automate the SLA life cycle management process.*	

	Postconditions	
5	Phase	*Prepare Mechanism Cookbook*
	Actor	*SPECS developer*
	Preconditions	
	Trigger	
	Actions	*The SPECS developer prepares the cookbook which automates the security mechanism's execution. The cookbook is organized according to Chef rules. SPECS Monitoring Adapter must be developed accordingly.* *The SPECS developer tests the developed security mechanism.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_4, U_6 (CSN:developer)*	
Target services	*Not Applicable*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*Not Applicable*	

CRO.10 SPECS_Application_Development

General Information			
ID		*CRO.10 - SPECS_Application_Development*	
Version		*1.0*	
User Story		*n.d.*	
Invocation Chain		*n.d.*	
Scenario Steps			
General Description		*A SPECS developer aims at developing a new SPECS application. In this scenario, the development of a new SPECS application, using the default SPECS application as a template, is shown.*	
Steps			
1	Phase	*Cloud Service Definition*	
	Actor	*SPECS developer*	
	Preconditions		
	Trigger		
	Actions	*The SPECS developer defines the types of cloud services to deliver and prepares the related cookbooks. She/he needs to specify the mechanisms able to enforce specific security capabilities and/or monitor specific metrics, as well as she/he needs to provide proper mechanisms to automatically deploy and configure the target services themselves.*	
	Postconditions		
2	Phase	*Prepare Security Mechanisms*	
	Actor	*SPECS developer*	
	Preconditions		
	Trigger		
	Actions	*The SPECS developer selects, among available security mechanisms, those needed to offer the cloud services.*	
	Postconditions		

3	Phase	*Prepare SLA Template*
	Actor	*SPECS developer*
	Preconditions	
	Trigger	
	Actions	*The SPECS developer builds a WS-Agreement-compliant SLA template, which summarizes the security capabilities that can be offered and the related guarantees.*
	Postconditions	
4	Phase	*Deploy SLA Templates and Security Mechanisms*
	Actor	*SPECS developer, SLA Platform*
	Preconditions	
	Trigger	
	Actions	*The SPECS developer deploys the security mechanisms in order to make them available to the SPECS application. All the cookbooks must be registered with the Chef Server in order to enable the SPECS Enforcement module to implement the SLA, and the mechanisms' metadata must be registered in the SLA Platform in order to enable the SPECS application to retrieve the information and to implement the SLA.* *The SPECS developer tests the deployed SPECS application.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_4, U_5, U_6 (CSN:developer)*	
Target services	*Not Applicable*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*Not Applicable*	

AAA.1 Identity_Management_Set-up

General Information		
ID	*AAA.1 – Identity_Management_Set-up*	
Version	*1.0*	
User Story	*STOIM*	*Secure Storage with Identity Management*
Invocation Chain	*IM2-CSP*	*Interaction Model 2- SPECS acting the role of CSP*
Scenario Steps		
General Description	*In this scenario, a customer acquires the enhanced secure storage service from the SPECS Owner, configures the service and sets the access control policies for its End-users by using the identity management features offered by the service. Moreover, the provider configures the Identity Federation by identifying the supported identity providers.*	
Steps		
1	Phase	*Service acquisition*
	Actor	*Customer, SPECS Owner*
	Preconditions	
	Trigger	

	Actions	*The customer acquires the enhanced secure storage service from the SPECS Owner and is provided with access to an application for its configuration.*
	Postconditions	
2	Phase	*Identity Management Set-up*
	Actor	*Customer, AAA mechanisms component*
	Preconditions	
	Trigger	
	Actions	*The customer accesses the application and configures, via the AAA mechanisms offered, the storage service that will offer to End-users by identifying the access control policy. The customer sets different authorization roles for the users and configures the tools for authentication (e.g., LDAP, OAUTH) and authorization (e.g., XACML).*
	Postconditions	
3	Phase	*Identity Federation configuration*
	Actor	*Customer, AAA mechanisms component*
	Preconditions	
	Trigger	
	Actions	*The customer, via the AAA mechanisms offered with the service, identifies a set of external Identity Providers that he aims at supporting in an Identity Federation (e.g., Facebook)*
	Postconditions	
Graphical Model		*Not reported*

Coverage Information	
Users	*U_1(CSC:User)*
Target services	*Not Applicable*
SPECS services	*AAA mechanisms*
SLA	*Not Applicable*

AAA.2 User_Registration

General Information		
ID	*AAA.2 – User_Registration*	
Version	*1.0*	
User Story	*STOIM*	*Secure Storage with Identity Management*
Invocation Chain	*IM2-CSP*	*Interaction Model 2- SPECS acting the role of CSP*
Scenario Steps		
General Description	*In this scenario, an End-user of the enhanced secure storage service performs a registration by providing her/his data.*	
Steps		
1 – Phase	*Registration*	
1 – Actor	*End-user, AAA mechanisms component*	
1 – Preconditions	*The provider of the service (i.e. the customer that has acquired the service from the SPECS Owner in the User Story) has defined an access control policy.*	
1 – Trigger		

	Actions	*The End-user fills the registration form with her/his personal information and specifies the features of the storage service she/he is interested to use. The information is submitted to the AAA component.*
	Postconditions	
2	Phase	*Registration*
	Actor	*AAA mechanisms component, End-user*
	Preconditions	
	Trigger	
	Actions	*A new account is created for the End-user and submitted information is associated with it. The End-user is provided the credentials to access the application.*
	Postconditions	
Graphical Model		*Not reported.*
Coverage Information		
Users	*U_1(CSC:User)*	
Target services	*Not Applicable*	
SPECS services	*AAA mechanisms*	
SLA	*Not Applicable*	

AAA.3 User_Access_Internal_Account

General Information			
ID		*AAA.3 - User_Access_Internal_Account*	
Version		*1.0*	
User Story		*STOIM*	*Secure Storage with Identity Management*
Invocation Chain		*IM2-CSP*	*IM2-CSP*
Scenario Steps			
General Description		*In this scenario, an End-user requests the access to the storage system by using the account created when registering with the service;*	
Steps			
1	Phase	*Authentication*	
	Actor	*End-user, AAA mechanisms component*	
	Preconditions		
	Trigger		
	Actions	*The End-user submits an authentication request (through, for example, an SAML request) to the SPECS AAA component by using the account previously created during registration.*	
	Postconditions		
2	Phase	*Authentication*	
	Actor	*End-user, AAA mechanisms component*	
	Preconditions	*The End-user has a valid account on the application*	
	Trigger		
	Actions	*The AAA component checks the account in the internal repository (e.g., LDAP server) and authenticates the End-user, by applying the access control policy related to her/his role.*	
	Postconditions		

Graphical Model	*Not reported*
Coverage Information	
Users	*U_1(CSC:User)*
Target services	*Not Applicable*
SPECS services	
SLA	*Not Applicable*

AAA.4 User_Access_External_Account

General Information			
ID		*AAA.4 - User_Access_External_Account*	
Version		*1.0*	
User Story		*STOIM*	*Secure Storage with Identity Management*
Invocation Chain		*IM2-CSP*	*IM2-CSP*
Scenario Steps			
General Description		*In this scenario, an End-user requests access to the storage system by using an external account belonging to a supported Identity Provider. When the user chooses to authenticate through an external source, the application checks that the external account is associated with a supported identity provider. In this case, the user is authenticated.* *Otherwise the application asks if the End-user wants to associate the external account to her/his existing internal account. In this latter case, the End-user must first be authenticated on the application in order to prove the ownership of the internal account.*	
Steps			
1	Phase	*Authentication*	
	Actor	*End-user*	
	Preconditions	*The End-user has a valid account on the selected external authentication source.*	
	Trigger		
	Actions	*The End-user requests the access to the storage service by selecting an external authentication source and performs the login with the credentials of the external account, retrieving her/his personal information.*	
	Postconditions	*The End-user is authenticated on the external authentication source.*	
2.1	Phase	*Authentication*	
	Actor	*AAA component, End-user*	
	Preconditions	*An internal account exists for the End-user. The internal account is already linked to the external account.*	
	Trigger		
	Actions	*The AAA component checks if the external account is associated with any valid internal account and authenticates the End-user.*	
	Postconditions	*The End-user is authenticated on the application.*	
2.2	Phase	*Authentication*	
	Actor	*AAA component, End-user*	
	Preconditions	*An internal account exists for the End-user. The internal account is not yet linked to the external account.*	
	Trigger		

	Actions	*The AAA component checks if the external account is associated with any valid internal account and does not find any match. The AAA component asks the End-user to associate the external account to her/his existing internal account, if any exists.*
	Postconditions	*The internal account of the End-user is linked to this/he external account.*
3.2	Phase	*Authentication*
	Actor	*End-user, AAA component*
	Preconditions	
	Trigger	
	Actions	*The End-user logs into the application with the credentials of the internal account. The AAA component authenticates the End-user.*
	Postconditions	*The AAA component End-user is authenticated.*
4.2	Phase	*Account association*
	Actor	*SPECS AAA component*
	Preconditions	
	Trigger	
	Actions	*The link with the external account is created for the user entry by the AAA component.*
	Postconditions	*The link to the external account is stored in the AAA component repository*
Graphical Model		*Not reported*
Coverage Information		
Users	*U_1(CSC:User)*	
Target services	*Not Applicable*	
SPECS services		
SLA	*Not Applicable*	