TNOVA

NETWORK FUNCTIONS AS-A-SERVICE OVER VIRTUALISED INFRASTRUCTURES

GRANT AGREEMENT NO.: 619520

Deliverable D6.01

Interim report on T-NOVA Marketplace implementation

Editor	Aurora Ramos (ATOS)
Contributors	Javier Melián (ATOS), Thomas Pliakas (CLDST), Yacine Rebahi (FOKUS), Paolo Comi (ITALTEL), Evangelos Markakis, George Alexiou, Athina Bourdena, Evangelos Pallis (TEIC), Piyush Harsh, Antonio Cimmino (ZHAW)
Version	1.0
Date	December 22nd, 2014
Distribution	(PU)

Executive Summary

This report outlines the outputs of the activities carried out in the first stages of the WP6 of the EU-FP7 T-NOVA Project: "Functions as-a-Service over Virtualised Infrastructures", WP that focuses on the implementation of the marketplace. Previous works in the project related with the current deliverable were: use cases definition and gathering requirements at system level, design of the overall architecture of the whole T-NOVA system [1], and requirements and specification of the T-NOVA Marketplace [2].

The marketplace in the Network Function Virtualization (NFV) scheme is an innovative concept that T-NOVA introduces with the aim of promoting the VNF (Virtual Network Functions) service offerings and facilitating the commercial activity and fluent interaction among the different business stakeholders identified, with the overall objective to bring the NFV closer to the market.

The T-NOVA Marketplace has been designed as a distributed platform placed on top of the overall architecture which, besides including the users front-end, it comprises BSS components as billing and accounting, and innovative modules as the T-NOVA Brokerage that facilitates auctioning among VNF developers.

In order to provide modularity the T-NOVA Marketplace will be developed with a Software Oriented Architecture (SOA) based on microservices that communicate with one another by means of RESTful APIs. By using this software architecture model, each component can be implemented separately in any technology and will be more easily integrated in the overall system.

This report gathers a reviewed and consolidated version of the the requirements identified in previous work [2] for each of the modules of the T-NOVA Marketplace, which are: business service catalog, brokerage module, user dashboard and SLA and billing management components. Next, a deep study of the state of the art has been elaborated focusing specifically in those previous solutions that may be applicable to T-NOVA approach, in order to find the most suitable background to build on when implementing the T-NOVA Marketplace: commercial solutions, standardization bodies and previous research projects. Finally, the T-NOVA framework has been explained for each of the modules as well as their architecture and the definition of the APIs, highliting the particularities for its implementation, including the study, identification and selection of the appropriate technologies and justificating the technical decisions and steps made so far to achieve our objectives.

Table of Contents

Index of Figures

Index of Tables

1. INTRODUCTION

1.1. Objectives and scope

This deliverable presents the current activities and interim results of the four active tasks in Work Package 6 of the T-NOVA project. The overall objective of WP6 is the implementation of the T-NOVA Marketplace and the tasks included in it are devoted to the implementation its different key components, based on the previous work done in the project for the T-NOVA Marketplace specification [3]:

- Task 6.1 is in charge of implementing the service description framework, including the implementation of the business service catalog.
- Task 6.2 is in charge of implementing the Brokerage module.
- Task 6.3 is in charge of implementing the dashboard. Besides the GUI (Graphical Users Interface) includes the implementation of the AA (Authentication and Authorization).
- Task 6.4 is in charge of implementing the SLA and billing components.

Current activities focus on the study, identification and selection of the appropriate technologies for implementation based on the requirements gathered in [3] and on the study of previous solutions in order to find the most suitable background to build on when implementing the T-NOVA Marketplace. The T-NOVA framework is also explained in this report specifically for each of the marketplace module as well as their architecture, highliting the implementation particularities, and justifying the technical decisions and steps made so far.

1.2. T-NOVA Marketplace highlevel overview

All features supported by the T-NOVA Marketplace will have to be compliance with the generic T-NOVA business scenario as depicted in Figure 1-1 which reflects the two main commercial relationship that are in T-NOVA: one between the Service Provider (SPs) and Function Providers (FPs) to acquire standalone VNFs to compose a Network Service (NSs) and the second one between the SP and the customer who acquire NSs.

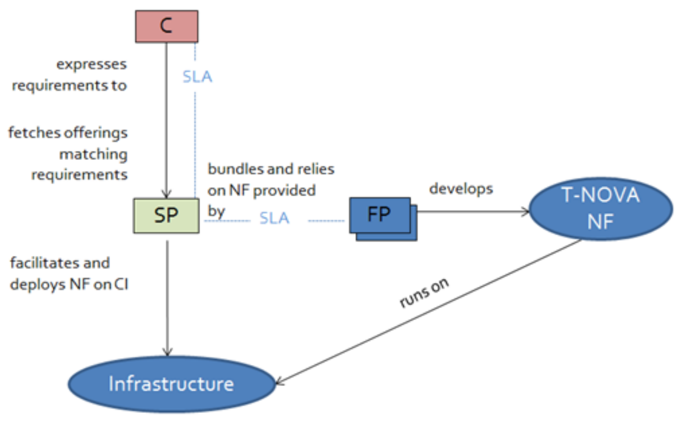

C
expresses requirements to
SLA
fetches offerings matching requirements
bundles and relies on NF provided by
SP
SLA
FP
develops
T-NOVA NF
facilitates and deploys NF on CI
runs on
Infrastructure

Figure 1-1 Business T-NOVA stakeholders relationships [2]

The Function Providers (FPs) that want to sell their VNFs through T-NOVA Marketplace will enter the system providing their VNFs information: VNF metadata including technical constraints, SLAs, price, etc.

The Service Provider (SP) that want to purchase VNFs in order to later sell NSs through T-NOVA enters the system. The SP will be able to compose services acquiring VNFs by means of a brokerage that will facilitate auctioning process among several FPs offering VNFs with similar features in achieve the lowest price offer. Then, the SP will be able to compose NSs bundling VNFs, and advertise them creating offerings that will include service description, SLA and price and will be exposure by means of the T-NOVA marketplace to the customer.

The customer will be able to search for the end-to-end network services offerings that can be composed by one or several VNFs, and with different SLA level and price in each offering.

In the event that there is not any available service offering matching a customer request, a new service composition could to take place triggered by the SP and trading mechanisms will be performed among FPs if several FPs offer similar VNFs dynamically.

The customer will be able to select offerings, and the SLA agreement procedure will be initiated: between customer and SP and consequently between SP and FPs; then the service provisioning will start.

All the related information SLAs, prices, etc. will be stored in the marketplace modules for later billing purposes.

Customer, SP and FP will be able to access their related information by means of the dashboard as it can be the service monitoring information, SLA fulfilment information and billing information.

1.3. WP6 inter-task dependencies

The outputs of the tasks within WP6 have a number of key dependencies within other tasks in the same WP along with tasks in other work packages including WP3 – Orchestrator platform and WP5 – Network Functions as it is outlined in tables 1-1 to 1-4. Therefore close cooperation and coordination between the dependent tasks will and be required to ensure the outputs are appropriate and maximise impact.

Dependent Task	Dependency
T3.1 – Orchestrator Interfaces	There is some information in the T-NOVA information model that has to be considered at both orchestrator and marketplace level.
T3.4 - Service Provisioning Management and Monitoring	The Network Service Descriptor in the orchestrator is related to the Service Description at martketplace level.
T5.1 Function Packaging and Repository	It would be preferable that the Network Service and the VNF offerings are described with analogous syntaxis and language by the SP and FPs respectively.
T6.3 - User Dashboard	The business service catalog will be browsed by the

	customer by means of the Dasboard. The SP will include the service offerings in the business service catalog by means of the dashboard.

Table 1-1 Outline of Task 6.1 (Service Description) inter-task dependencies

Dependent Task	Dependency
T3.1 – Orchestrator Interfaces	The brokerage module will know about the available VNFs by means of the orchestrator.
T3.3- Service Mapping	The VNFs may have special requirements to run over a compute node. This should be provided in the metadata, and also might yield different pricing.
T5.1 Function Packaging and Repository	The brokerage module will manage the VNF metadata and VNF descriptor to manage price, SLA offer and FP ID for the brokeraging process.
T6.1- Service Description	The brokerage needs to manage the Service Description scheme in order to use it in the brokering process.
T6.3 - User Dashboard	The trading process between the T-NOVA stakeholders managed by the brokerage will have to interface with the dashboard.
T6.4 SLA and billing	The trading mechanisms for the VNFs implemented by the brokerage will have to consider different SLA levels for each VNF. Also the price set as a result of an auctioning process will have to be store in the accounting for billing purposes.

Table 1-2 Outline of Task 6.2 (Brokerage module) inter-task dependencies

Dependent Task	Dependency
T3.1 – Orchestrator Interfaces	There will be a specific interface between Dashboard and Orchestrator in order to notify the orchestrator about a new/update/deleted Network Service (NS), to order a network service instantiation and to visualize monitoring information of the service by the stakeholders.
T3.4 - Service Provisioning Management and Monitoring	The dashboard will provide all the necessary visualization tools for the Service Management and monitoring.
T4.4 – Monitoring and Maintenance	It is assumed that some IVM metrics will be presented on the dashboard.
T6.1- Service Description	The business service catalog will be browsed by the customer by means of the Dasboard. The SP will include the service offerings in the business service catalog by means of the dashboard.
T6.2 - Brokerage module	The trading process between the T-NOVA stakeholders

	managed by the brokerage will have to interface with the dashboard.
T6.4 SLA and billing	The dashboard will integrate a GUI for the SLA agreement and monitoring procedures for the the different stakeholders, and to visualize all the billing information

Table 1-3 Outline of Task 6.3 (Users Dashboard) inter-task dependencies

Dependent Task	Dependency
T3.4 - Service Provisioning Management and Monitoring	SLA agreement and monitoring procedures strongly depend on the metrics that the orchestrator will be able to monitor.
T4.4 – Monitoring and Maintenance	SLA agreement and monitoring procedures strongly depend on IVM metrics.
T5.3 – Development of Network Functions	VNF developers will offer different parameters to be considered as part of different SLAs for their VNFs.
T6.2 - Brokerage module	The brokerage module performs the trading considering different SLA levels by means of VNF auctioning.
T6.3 - User Dashboard	The dashboard will integrate a GUI for the SLA agreement and monitoring procedures for the the different stakeholders, and to visualize all the billing information.

Table 1-4 Outline of Task 6.4 (SLA and billing) inter-task dependencies

1.4. Document structure

This report is structured as follows:

Firstly, section 2 provides an overview of the general architecture of the T-NOVA Marketplace and the decisions taken for its implementation.

Section 3 focuses in the work done about the way in which the service description and exposure is going to be performed in T-NOVA. For that, a previous study of services description languages has been included. Then the first draft of the T-NOVA information model is presented as well as the first technologic decisions to implement the business service catalog and its interfaces.

Section 4 is devoted to the brokerage module. It presents the consolidation of the requirements gathered previously for this compoment and some previous solutions for brokerage in the state of the art are collected. Next the T-NOVA brokeraging scenario is explained providing a first pscudo-code algorithm for its implementation. Finally the internal architecture of the brokerage module is addressed and the rationale about the possibility of using or not some existing brokerage solutions.

Section 5 focuses in the implementation of the T-NOVA Dashboard. After the requirements consolidation and overview of the main characteristics of T-NOVA Dashboard, the components of the dashboard are introduced, as well as the rationale of the technologies that are going to be used for its implementation. Section 5.6 addresses the permissions that will have to be managed by the Access Control module for the different stakeholders.

Section 6 is devoted to the study and implementation of the SLA procedures in T-NOVA. A survey on the state of the art is included, considering comercial solutions, research projects and standardization bodies. Then, the specific analysis for the T-NOVA SLA framework is explained, and finally the first technological decisions taken for their implementation and their interfaces definition.

Section 7 is devoted to the study and implementation of billing procedures, including the steps done so far for the implmention of the accounting module. Following the same structure as previous sections, it includes the study of the state of the art and a specific analysis for the T-NOVA billing framework. Furthermore the first technological decisions taken for their implementation are collected as well as the interfaces definition.

Finally, section 8 summaries the conclusions of the work presented in this report.

2. T-NOVA MARKETPLACE ARCHITECTURE

2.1. High level architecture

The T-NOVA Marketplace has been designed as a distributed platform placed on highest layer in the overall architecture [1] which, besides including the users front-end, it comprises BSS components as billing and accounting, and innovative modules as the T-NOVA Brokerage.

Figure 2-1 depictes the highlevel architecture of the T-NOVA Marketplace naming its internal and external interfaces which will be detailed along this report.

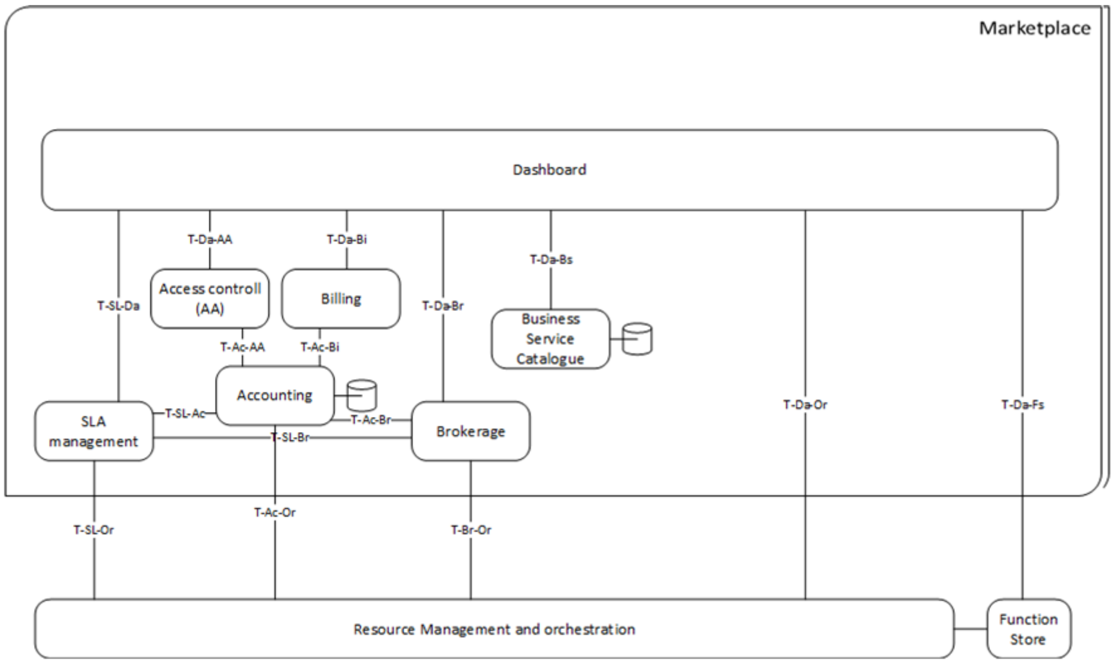


Figure 2-1 Marketplace interfaces

Table 1-1 provides a short description of the different modules that are part of the marketplace, indicating in each case the section of this report where extended information can be found.

Name	Description
Business Service Catalog	It stores all the available service offerings in the marketplace for service exposure (see section 3).
Brokerage Module	It is the entity which enables the interaction among actors for service request and brokerage/trading (see section 4).
Dashboard	It provides the user front-end, exposing in a graphical manner all customer-facing services. It will have three different views for each kind of the stakeholders accessing the system: SP dashboard, FP dashboard and customer dashboard (see section 4.8).

Access Control Module	It administers security in a multi-user environment, managing and enabling access authorization/control for the different T-NOVA stakeholders considering their roles and permissions (see section 5.6).
SLA Management Module	It establishes and stores the SLAs among all the involved parties and checking if the SLAs have been fulfilled or not will inform the accounting system for the pertinent billable items (penalties or rewarding) (see section 4.8).
Accounting Module	It stores all the information needed for later billing for each user: usage resources for the different services, SLAs evaluations, etc. (see section 7)
Billing Module	It produces the bills based on the information stored in the accounting module (see section 7).

Table 2-1 Marketplace components

The information that will be exchange by the different interfaces is briefly explained in Table 2-2 and Table 2-3. Further details about the operations that will be performed within each interface are collected along this report when defining the API operations supported by each marketplace component.

Marketplace External Interface	**Description**
T-Da-Or	It is used to notify the orchestrator about a new/update/deleted Network Service (NS), about a new/updated/deleted NS instantiation and to get the monitoring information of the service by the customer and SP.
T-Sl-Or	SLA module is notified with currently running NS metrics from the monitoring system in the orchestrator.
T-Br-Or	The brokerage module will use it to retrieves information about the available VNFs.
T-Ac-Or	The accounting is notified about state of each Network Service (NS) or VNF.
T-Da-Fs	It is used to upload VNF and metadata by the FPs.

Table 2-2 Marketplace external interfaces

Marketplace internal interfaces	**Description**
T-Ac-AA	It is used by the accounting module to access the "user profiles".
T-Ac-Bi	All the information needed for billing is stored in the accounting module.

T-Sl-Ac	SLA module accesses the accounting module to inform about SLA violations for penalties to be applied.
T-Br-Sl	This interface is used by the brokerage module to provide information to the SLA management module regarding the SLA agreed between SP and FPs as a result of the trading process.
T-Sl-Da	It is used to specify SLA by SP and FPs, and to negotiate and agree SLAs between the implied stakeholders.
T-Da-AA	It is used to provide and collect all the information necessary to authenticate the T-NOVA users or stakeholders.
T-Da-Bi	It is used to visualize billing information by the three stakeholders.
T-Da-Br	It is used to request VNFs, to facilitate auctioning among FPs, and optionally to request service offerings by the customer.
T-Br-Ac	With this interface the brokerage module will provide to the accounting system the appropriate information related to service selections and price.
T-Da-BSC	It is used to publish offerings by the customer, and to browse offerings by the customer.

Table 2-3 Marketplace internal interfaces

2.2. Marketplace implementation

In order to enhance the T-NOVA marketplace with the modularity specified in previous work [1] the T-NOVA Markeplace implementation will be based on a microservices software architecture [4].

The microservices software architecture model provides the necessary tools for each marketplace module to run separately as a standalone service. Hence, each module can manage its own database (if needed) or share a database with other module.

Furthermore, by using this software architecture model, each component can be implemented separately in any technology (e.g. Java, Python, etc.) and can be more easily integrated in the overall system which is the marketplace.

According to Figure 2-1 there are several components that communicate with one another and must be treated as individual services that provide and/or consume an API. For the implementation of the Marketplace we have identified one service for each of the different modules described in Table 2-1, each of them exposing a REST API (Figure 2-2).

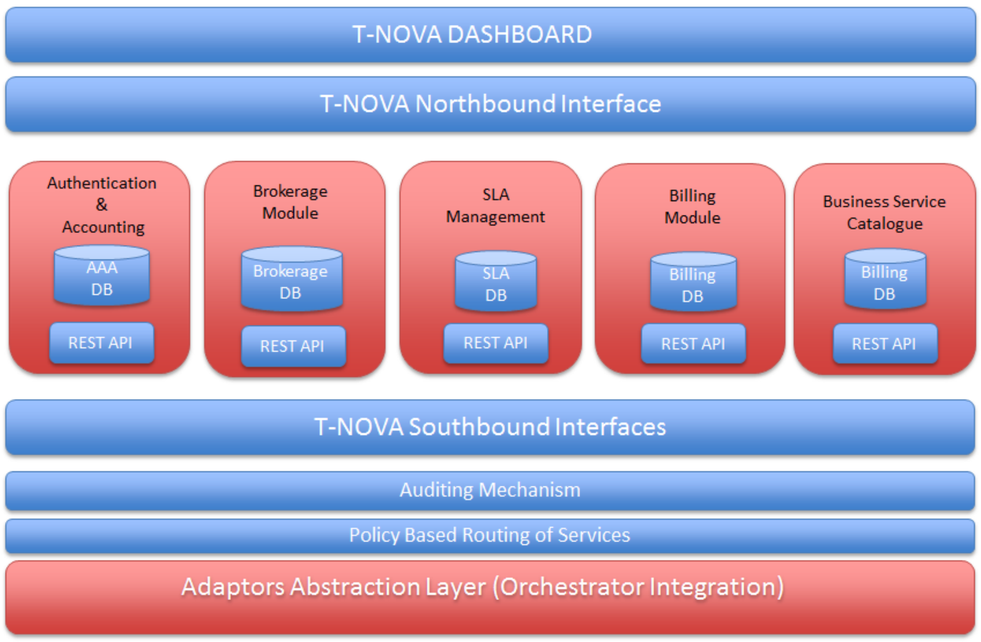

T-NOVA DASHBOARD
T-NOVA Northbound Interface
Authentication & Accounting
AAA DB
REST API
Brokerage Module
Brokerage DB
REST API
SLA Management
SLA DB
REST API
Billing Module
Billing DB
REST API
Business Service Catalogue
Billing DB
REST API
T-NOVA Southbound Interfaces
Auditing Mechanism
Policy Based Routing of Services
Adaptors Abstraction Layer (Orchestrator Integration)

Figure 2-2 Marketplace implementation

REST [5] defines a set of architectural principles by which it is posible to design web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages.

REST has emerged in the last few years alone as a predominant web service design model. In fact, REST has had such a large impact on the web that it has mostly displaced SOAP and WSDL-based interface design because it is a considerably simpler style to use.

The T-NOVA implementation of each REST web service will follow four basic design principles:

- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer JavaScript Object Notation (JSON).

Therefore, according to REST model all the T-NOVA Marketplace microservices will support the following methods requests:

- GET method requests to retrieve the information about module configured services.
- DELETE method requests to delete/discard the configured services.
- POST method requests to create the configured object.
- PUT method requests to update the configured services.

These operations are specified in the following sections for each of the modules.

3. Service Description and Exposure (Business Service catalog)

3.1. Introduction

This section explains the status of T6.1 – Service Description Framework, whose objective is to specify and implement the service description schema and information model for the T-NOVA services at marketplace level, including:

- Network services' asset description.
- Service offering advertisement.
- Service description storage.

The T-NOVA Marketplace will include a catalog in order to facilitate the exposure of the available services to the T-NOVA customer, decided after carefull assessment of the state-of-the-art [3]. This catalog is named "business service catalog" in line with TMForum [6] terminology as defined in its "integration framework", in which functional and non-functional aspects of a service based on service oriented principles are defined.

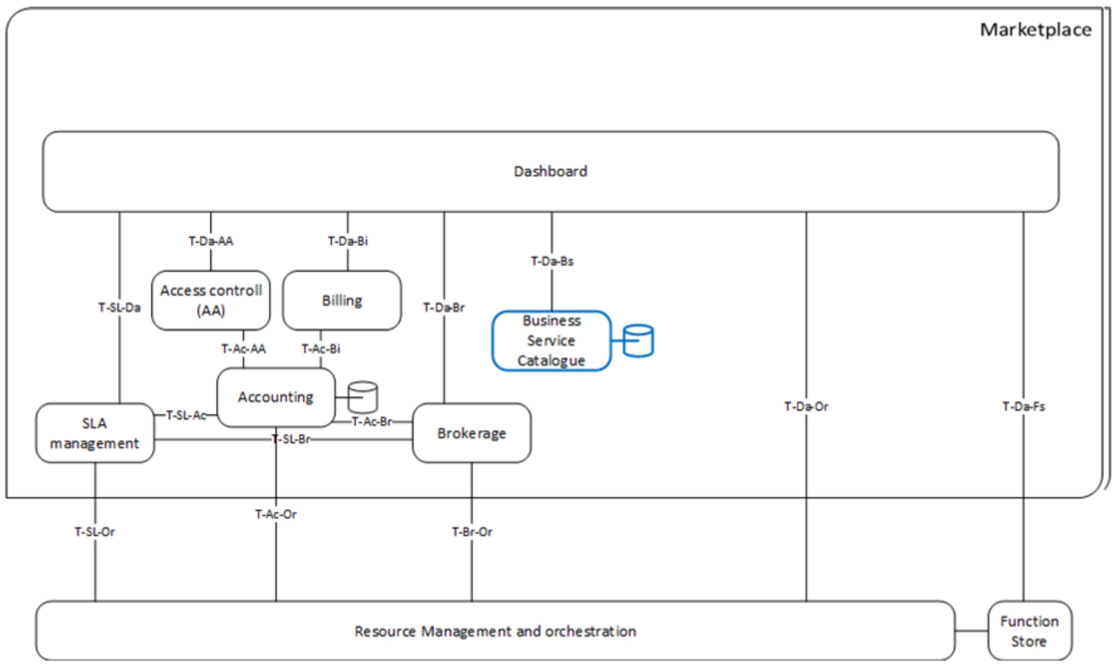


Figure 3-1 Business Service Catalog in T-NOVA marketplace

3.2. Requirements overview and consolidation

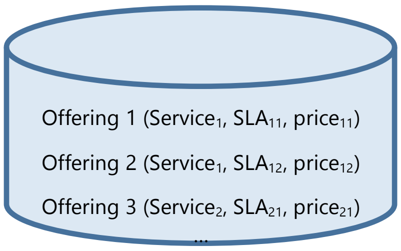
This section elaborates further on the requirements provided in [3] (see Table 3-1) and new ones that have emerged while working on the technical aspects for implementation. This effort is required, given the decision to include the development of a business service catalog in the T-NOVA marketplace.

Req. id	Domain	Requirement Name	Requirement Description
SC.1	Service continuity + Market/commercial operability	Services and SLAs description	The service catalogSHALL be able to store all the available NSs in the T-NOVA marketplace, specifying SLA level and price.
SC.2	Service continuity + Market/commercial operability	Services and SLAs description	The service catalog SHALL be browsable by the SP and customer

Table 3-1 . Business Service Catalog basic requirements

The requirements that we identified during the development process are:

- The business service catalog shall have entries containing what we have named "Offering". Each Offering will be composed of a high level service description + SLA offer (summary of the SLA specification performed by the SP) + price, as


Offering 1 (Service$_1$, SLA$_{11}$, price$_{11}$)
Offering 2 (Service$_1$, SLA$_{12}$, price$_{12}$)
Offering 3 (Service$_2$, SLA$_{21}$, price$_{21}$)
...

- Figure 3-2 shows.

Figure 3-2 Business Service Catalog

- The business service catalog shall be filled/updated with the Offerings information manually and offline by the SP, since it is a business process, after a new service is available in the system.
- The customer will be able to browse directly the business service catalog.
- The customer will be able to select one of the offerings which will create the service instantiation order in the orchestrator.

3.3. State of the art for service description in T-NOVA

3.3.1. Service description languages survey

The diversity of service description languages and vocabularies directly reflects the variety of notions and forms a network service takes across domains. Some network topology-oriented descriptions have been proposed to allow interoperability in network services. Among them, Network Description Language (NDL) [7] provides a language to describe optical networks based on the resource description framework (RDF); Virtual Resources and Interconnection Networks Description Language (VXDL) [8] defines a language for virtual resources interconnection network specification and modelling. It allows describing resources as well as network topology. These, together with perfSONAR [9]and cNIS [10] have joint efforts in network description standard proposed by Open Grid Forum, Network Markup Language (NML) [11].

OWL-S, WSMO and SA-WSDL [12] describe services through semantic information, but not common vocabulary and taxonomies for defining business aspects like the license model.

With a more business oriented perspective, Unified Service Description Language (USDL) [13] aims to embrace a wide approach considering business, operational and technical aspects of services. USDL reflects diverse non-functional aspects such as pricing and legal constraints, as well as, service interfaces for delivery and service level agreements together with details for the service functionality. USDL relies on Web Service Description Language (WSDL) [12], popular in Service Oriented Architectures (SOA).

Once the services are described and stored in the catalog, we will have several options in the state of the art to manage the retrieval of that information. Two of the most relevant ones are:

- Z39.50 [14] protocol for information retrieval, procedures and formats for a client to search a database provided by a server, retrieve database records and perform related information retrieval functions.

- Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [15] is a protocol that collects the metadata description of the records in an archive so that services can aggregate metadata from many archives.

3.3.2. Previous research projects

FIWARE [16] and XIFI [17] projects rely on WStore [18] to support their Marketplace and Repository. WStore is a store for selling services to both consumers and developers of Future Internet applications and services and for end-to-end managing of offerings and sales.

From the point of view of a service provider, the usage scenario describes the creation and publication of offerings. Before creating offerings containing service components, those components should be uploaded to the deployed WStore server (respository). To register those assets, the service component provider can use the Web interface provided by the WStore service or directly with the Web interface.

The RESERVOIR project [19] elaborated on top of Open Virtual Format (OVF) to define a Service Manifest that specifies the capacity requirements for an explicitly sized service application as agreed between the infrastructure provider and the service provider, and a set of elasticity rules that define the service adaptability criteria based on KPIs.

The OPTIMIS project [20] added to this, the consideration of legal constraints together with cost, risk, eco-efficiency and cost.

The ETICS project [21] elaborated on QoS-enabled interconnection models between network service providers. In order to do so, it defined the concept of end-to-end service specification that represents the orchestration of diverse atomic "Service Elements" offered by network service providers. Atomic Service Elements in this context could be composite in different ways (by means of centralized, hierarchical or interlinked contracts) in order to form end-to-end services adapted to specific constraints. Another issue has been addressed partially by the ETICS project: the question of the service offering availability. The issue is double: first having an accurate knowledge of the resource capacities is not trivial, second this is usually confidential information. On the other side, a service broker may be inefficient without such information and could infer it possibly in a pessimistic way. The ETICS consortium proposed to add a validity deadline to the service offering but did not study the impact on the service catalog resilience (many updates or not). Both push and pull models were analised for the brokering and publishing phase.

3.3.3. Standarization bodies

3.3.3.1. ETSI NFV

ETSI NFV group gives a first approach for the way the NS in the NFV scheme should be described at orchestration level for the MANO operation, containing E2E service description & KPIs, information about component VNFFGs and associated endpoints [22]. Not business connotations are provided by ETSI to this respect.

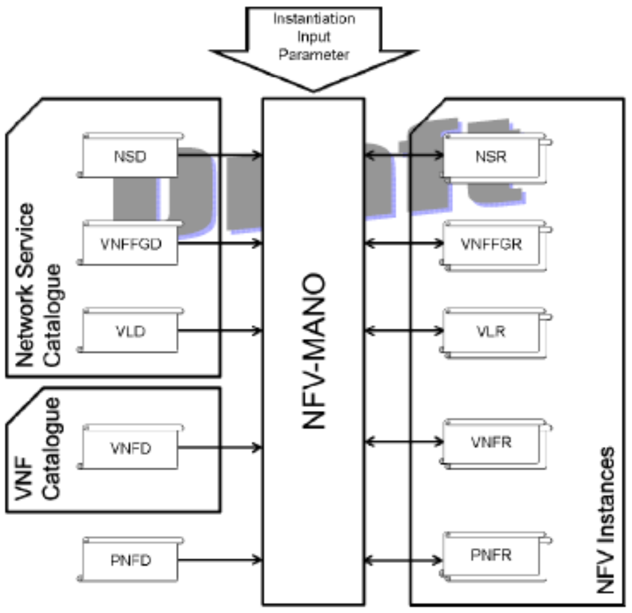
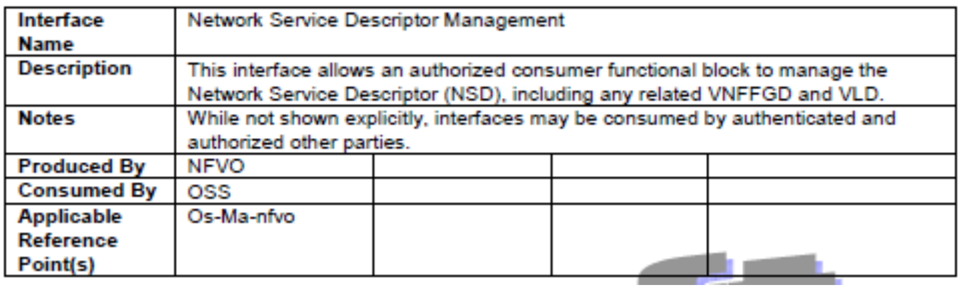


Figure 3-3 Information elements in NFV orchestration [22]

The interface that defines ETSI for NSD descriptor management with the OSS is depicted in Figure 3-4, which will be aligned with the operations that the Service Provider will provide to manage the NSD in the orchestrator.



Interface Name	Network Service Descriptor Management			
Description	This interface allows an authorized consumer functional block to manage the Network Service Descriptor (NSD), including any related VNFFGD and VLD.			
Notes	While not shown explicitly, interfaces may be consumed by authenticated and authorized other parties.			
Produced By	NFVO			
Consumed By	OSS			
Applicable Reference Point(s)	Os-Ma-nfvo			

Figure 3-4 ETSI MANO Os-Ma-nfvo interface

Table 3-2 collects the NSD fields provided by ETSI [22].

Identifier	Type	Cardinality	Description
Id	Leaf	1	ID of this Network Service Descriptor
vendor	Leaf	1	Provider or vendor of the Network Service
version	Leaf	1	Version of the Network Service Descriptor
vnfd	Reference	1..N	VNF which is part of the Network Service. This element is required, for example, when the NetworknService is being built top-down or instantiating the member VNFs as well.
vnffgd	Reference	0..N	VNFFG which is part of the Network Service. A Network Service might have multiple graphs, for example, for: 1. control plane traffic 2. management-plane traffic 3. User plane traffic itself could have multiple NFPs based on the QOS etc. The traffic is steered amongst 1 of these NFPs based on the policy decisions.
vld	Reference	0..N	Virtual Link which is part of the Network Service.
lifecycle_event	Leaf	0..N	Defines NS functional scripts/workflows for specific lifecycle events (e.g., initialization, termination, scaling)
vnf_dependen cy	Leaf	0..N	Describe dependencies between VNF. Defined in terms of source and target VNF i.e. target VNF "depends on" source VNF. In other words a source VNF must exist and connect to the service before target VNF can be initiated/deployed and connected. This element would be used, for example, to define the sequence in which various numbered network nodes and links within a VNF FG should be instantiated by the NFV Orchestrator.
monitoring_ parameter	Leaf	0..N	Represents a monitoring parameter which can be tracked for this NS. These can be network service metrics that are tracked for the purpose of meeting the network service availability contributing to SLAs (e.g. NS downtime). These can also be used for specifying different deployment flavours for the Network Service in Network Service Descriptor, and/or to indicate different levels of network service availability. Examples include specific parameters such as calls-persecond (cps), number-of-subscribers, no-of-rules, flows-persecond, etc.

			1 or more of these parameters could be influential in determining the need to scale-out
service_ deployment_ flavour	Element	1..N	Represents the service KPI parameters and its requirement for each deployment flavour of the NS being described. For example, there could be a flavor describing the requirements to support a vEPC with 300k calls per second. There could be another flavour describing the requirements to support a vEPC with 500k calls persecond.
auto_scale_policy	Leaf	0..N	Represents the policy meta data, which may include the criteria parameter & action-type. The criteria parameter should be a supported assurance parameter. An example of such a descriptor could be: • Criteria parameter: calls-per-second, • Action-type: scale-out to a different flavour ID
connection_point	Element	1..N	This element describes a Connection Point which acts as an endpoint of the Network Service. This can, for example, be referenced by other elements as an Endpoint.
pnfd	Reference	0..N	PNFs which are part of the Network Service.
nsd_security	Leaf	0..1	This is a signature of nsd to prevent tampering. The particular hash algorithm used to compute the signature, together with the corresponding cryptographic certificate to validate the signature should also be included.

Table 3-2 ETSI NFV Network Service Descriptor fields [22]

3.3.3.2. TMForum

The TMForum by means of its *Information Framework (SID, Shared Information/Data model)* [23] provides a common reference model for enterprise information that service providers, software providers and integrators use to describe management information. It is used to develop databases and provide a glossary of terms for business processes. The framework is intended to reduce integration costs and project management time and cost by minimizing the number of necessary changes to underlying architecture during the launch of a new product or service offering. More concretely, SID:

- Provides detailed models in an object-oriented form that can be used to further define MANO service and resource concepts.
- Provides a detailed model of how services and resources are managed, including definition of metrics to represent key characteristics and behaviour of services and resources as well as SLAs.
- Provides a framework to design interfaces and APIs for various business and operational processes.

3.3.3.3. Conclusions

Diverse standardisation efforts and research projects have elaborated on service description languages and vocabularies directly reflecting diversity and heterogeneity of acceptations that the service term has across different domains. The problem of service availability should also be considered both in a descriptive way (what is the information disclosed) to be in line with the providers' expectations and in an auditing way. Based on the state of art analysis, the progress on the T-NOVA service description framework is explained in the following sections together with the technological decisions already made.

3.4. T-NOVA Service Description framework

Progressing on the specification work to define the information that is needed to be stored and exchanged by the T-NOVA marketplace and its main interfaces, orchestrator and function store, we have created a first draft of the high-level information data model which is depicted in Figure 3-5.

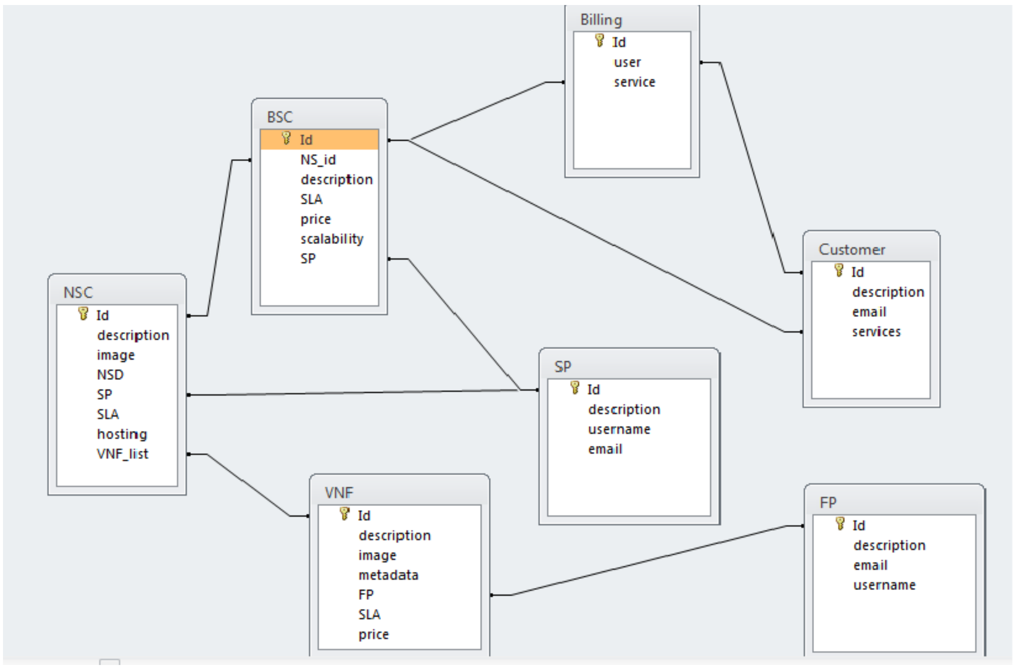


Figure 3-5 High-level information data model

When the Service Provider (SP) wants to create a new service to be available through T-NOVA marketplace, on one hand he will create the offering to be included in the business service catalog (marketplace), including a high level description definition, SLA offer and price. At orchestration level, this implies the creation of a Network Service Descriptor (NSD) with a service template to specify the service structure and management of behaviour of IT services. It is considered that, typically, services are provisioned in an IT infrastructure and their management behaviour (scaling, patching, monitoring, …) must be orchestrated in accordance with constraints or policies from there on, for example in order to achieve service level objectives related to SLAs.

Based on the T-NOVA orchestrator requirements, and in compliance with ETSI NSD, the first draft of the T-NOVA Network Service Description template is in depicted in Figure 3-6 [24].

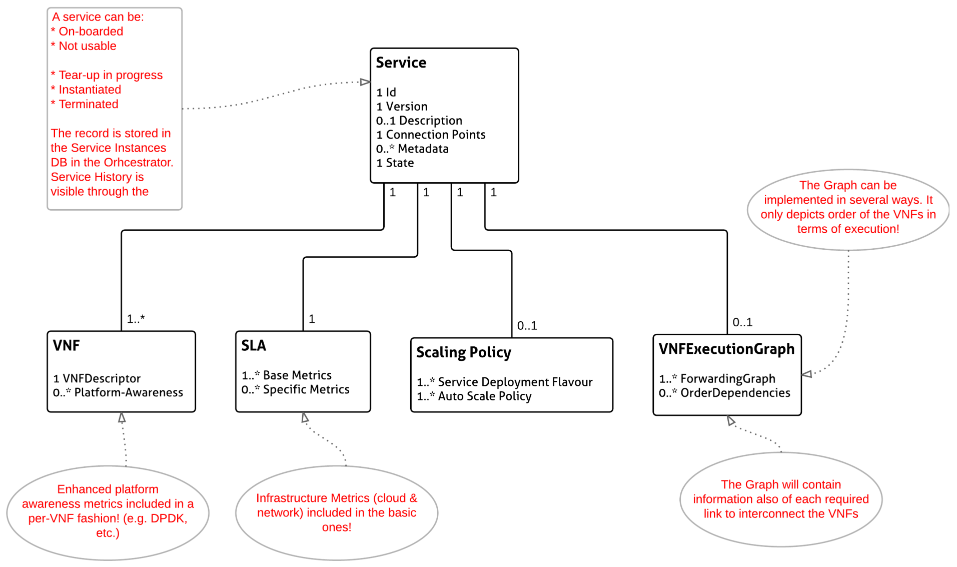

A service can be:
* On-boarded
* Not usable
* Tear-up in progress
* Instantiated
* Terminated
The record is stored in the Service Instances DB in the Orhcestrator. Service History is visible through the
Service
1 Id
1 Version
0..1 Description
1 Connection Points
0..* Metadata
1 State
1
1
1
1
The Graph can be implemented in several ways. It only depicts order of the VNFs in terms of execution!
1..*
1
0..1
0..1
VNF
1 VNFDescriptor
0..* Platform-Awareness
SLA
1..* Base Metrics
0..* Specific Metrics
Scaling Policy
1..* Service Deployment Flavour
1..* Auto Scale Policy
VNFExecutionGraph
1..* ForwardingGraph
0..* OrderDependencies
Enhanced platform awareness metrics included in a per-VNF fashion! (e.g. DPDK, etc.)
Infrastructure Metrics (cloud & network) included in the basic ones!
The Graph will contain information also of each required link to interconnect the VNFs

Figure 3-6 T-NOVA Network Service Descriptor [24]

3.5. Candidate Technologies and Rationale

In this section we justify the selection made for the service description language at marketplace level and the way to implement the BSC (Business Service Catalog).

Service description language

In [25] an exhaustive comparison work is done in relation to service description languages in virtualized environments applied in three different domains - general services, Web/SOA services and cloud services, and considering seven major aspects, which are: domain, coverage, purpose, representation, semantic expressivity, intended users and features. This is collected in Table 3-3. Semantic expressivity and coverage dimensions are regarded as the most essential factors for the evaluation of a service description model.

	Domain	**Coverage**	**Purpose**	**Representation**	**Semantics**	**Intended user**	**Feature**
O'Sullivan	General	Business / Non-functional	Service discovery / comparison / selection /substitution	Attribute taxonomy	No	Service requestor	Domain independent
USDL	General	Business & technical	Service matching & discovery	MOF-based meta-model	Low	Service requestor	Domain independent
WSDL / UDDI	SOA	Technical	Service communication / discovery	XML	Low	Service requestor	Web service
Rich service specification	SOA	Business & technical	Description, implementation, deployment	Document	No	Requestor, provider & developer	Timing, level of detail, role-related
SoaML	SOA & cloud	Software engineering	Service modeling &	UML	No	Software engineering	Model-driven

		g	design			practitioner	
OWL-S	Semantic Web service	Technical / Functional	Service modeling & design	OWL/RDF	High	Web service requestor	
WSDL-S	Semantic Web service	Technical / Functional	Service modeling & design	WSDL/XML	High	Web service requestor	Aligning with Web service standards, language interoperability
SAWSDL	Semantic Web service	Technical / Functional	Service modeling & design	WSDL/XML/RDF	High	Web service requestor	Lightweight SWS description language
Sun	Cloud	Technical / Software engineering	Cloud resource discovery & integration	Programming model	No	Application developers	Constraints-based
SMI	Cloud	Business	Service selection / comparison	Attributes taxonomy	No	Service requestor	Users satisfaction / preference
Blueprint	Cloud	Software engineering	Selection, customization & composition	Blueprint template	No	Application developers	Inter-changeability interoperability
Cloud#	Cloud	Technical	Transparency / enhance trust	Specification model	No	Service requestor	Internal organization

Table 3-3 Service Description Languages comparative [25]

Based on previous studies comparing service description languages in cloud/virtualized environments [25] and based on the state of the art we included in section - we have have concluded that USDL [13] is the language that best matches the T-NOVA marketplace requirements as USDL is the language with the widest coverage from business, technical and operational aspects.

USDL provides even friendly GUI editors allowing including SLA features and pricing. In contrast, while OWL-S is the one that has the highest semantic expressivity, it does not provide common vocabulary and taxonomies for defining some business aspects like the license model.

Business Service Catalog

There are several previous research projects implementing marketplaces, mainly in the context of Future Internet that rely on WStore [18]. Considering the modelling used in WStore (section 3.3.2)., T-NOVA will apply a similar usage scenario to describe the creation and publication of offerings but with the particularities of the services that will be offered in T-NOVA, in the NFV scheme, meanwhile WStore is oriented to Future Internet services.

In WStore, before creating the offerings the components that are part of the services of those offerings should be uploaded to the deployed WStore server (respository). In T-NOVA, the analog repository is part of the orchestrator [24]. To register those assets, in WStore the service provider can use the web interface provided by the WStore service.

Once the digital assets to be offered have been registered, the next step is the creation of the offering in WStore using the web interface provided. To create the offering, the service provider has to provide an USDL document and select the different downloadable assets previously registered in the WStore server. The provider can make use of the simple USDL creation form provided by WStore for creating simple USDL documents. During this process WStore uploads the USDL description of the offering to the repository.

The final step is the publication of the offering. Since the offering has been created in the WStore, its web interface can be used for the publication. When an offering has been published, it is included in the Marketplace to be available to potential customers.

In T-NOVA we will build our own services storage, using a similar approach of the one provided by WStore but applied to NFV services, and considering the service repositories that are needed in NFV at orchestration level. The offerings will be stored in the business service catalog, which will be implemented by means of a database developed with MySQL. This database will be called by the Dashboard using standarized communication by means of a REST API (developed in Python) and the SP will create a customized description using the USDL format.

3.6. Architecture

According to the decision explained in the previous section on how the service exposure and service discovery will be implemented in T-NOVA, the overall architecture for the interaction with the business service catalog is depicted in Figure 3-2.

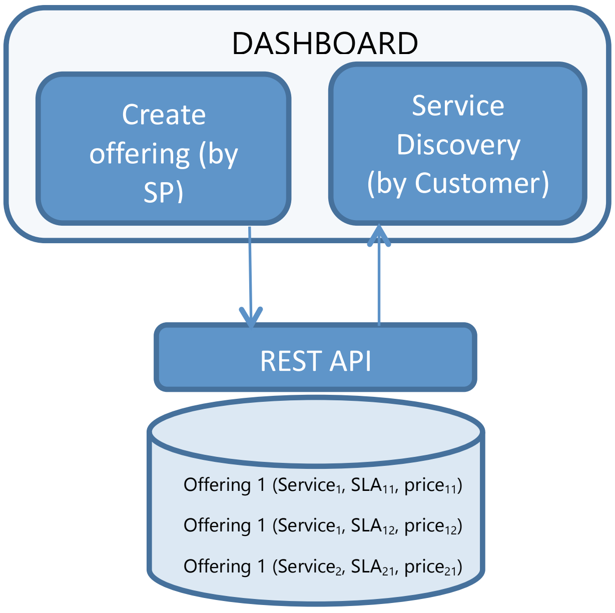

DASHBOARD
Create offering (by SP)
Service Discovery (by Customer)
REST API
Offering 1 (Service$_1$, SLA$_{11}$, price$_{11}$)
Offering 1 (Service$_1$, SLA$_{12}$, price$_{12}$)
Offering 1 (Service$_2$, SLA$_{21}$, price$_{21}$)

Figure 3-7 Dashboard-Business Service Catalog interface

3.6.1. Relation with other T-NOVA components

3.6.1.1. Dashboard (T-Da-BSC)

The business service catalog will be directly accessessible by the SP to publish their offerings, and browsable by the customer to select one of them.

3.6.1.2. Orchestration

Once the customer has selected a service, this will be translated in the orchestrator as an instantiation service ordering. This will be perform by means of the dashboard (T-Da-Or).

When the Service Provider creates a new service it will have to:

1. Provide the service parameters needed for the NSD in the orchestrator [24].
2. Describe offering in the Business Service Catalog (BSC) once the service is available.

3.6.1.3. Brokerage

The brokerage module so far was the only channel for the customer to receive the offering based on a service request; however at this stage it has been decided to consider also the option that the customer can directly browse the BSC ir order to provide flexibility to the system, simplicity and avoid limitations. The BSC will be directly accessed by the SP to publish their offerings, and browsable by the customer to select one of them. In next steps in the project, we may consider also the option that the customer will perform a request that the brokerage will use to provide the most suitable offering based on the customer demand.

3.6.2. API definition

According to the interfaces explained in the previous section (T-Da-BSC), the operations that the business service catalog API will support are the following:

- Operations that will be called by the "Service description" (SP dashboard):
  - new_service, new_USDL (POST)
    - Uploads the newly created service into the BSC database and its description.
  - services_list (GET)
    - Obtains the list of available services.
  - withdraw_service (DELETE)
    - Deletes a certain service from the BSC.
  - modify_service, modify_USDL, modify_SLA (PUT)
    - Updates the content of any service field, the service description or the SLA agreement.

- Operations that will be called by the "Service discovery" (Customer dashboard), alternatively they could be called by the brokerage module:
  - services_list (GET)

    - Obtains the list of available services.
  - service_description, service_SLA (GET)
    - Obtains the description of a service and its SLA agreement.

3.7. Technology selection summary

This section summaries in the following tables the two main decisions taken considering the implementation of service description in T-NOVA Marketplace, including alternative options available at this point in time, the requirements [2] that they satisfy, the trade-offs and the justification (e.g. the rationale behind the selection).

Final details of the technologies ultimately used to implement T-NOVA Service Framework Description will be provided in the next set of deliverables.

Topic	**Business Service Catalog**
Choice	▪ BD + REST API: module to be used by the dashboard that provides a customized description of the required services using standardized communication.
Alternatives	▪ WStore: store for selling services to both consumers and developers of Future Internet applications and services ▪ Murano. Application catalog over Openstack. Allows the developer to upload an application and manage its lifecycle.
Requirements Related	< SC.1 > The service catalog SHALL be able to store all the available NSs in the T-NOVA marketplace, specifying SLA level and price. < SC.2 > The service catalog SHALL be browsable by the SP and customer
Decision	The business service catalog will be implemented by means of a DB built from scratch but with a similar approach to WStore. It will not be used as it is because of its specifications for Future Internet applications, mainly in relation with the service repository, which in T-NOVA comes from the NFV orchestrator.

Topic	**Description language**
Choice	▪ USDL: reflects diverse non-functional aspects such as pricing and legal constraints, as well as, service interfaces for delivery and SLAs together with details for the service functionality.

Alternatives	▪ OWL-S, and WSMO. Ontology web language for services. Describes semantic web services. ▪ SA-WSDL. Defines a set of extension attributes for the Web Services Description Language (WSDL) and XMLS schema definition language. The specification defines how semantic annotation is accomplished using references to conceptual semantic models
Requirements Related	< SC.1 > The service catalog SHALL be able to store all the available NSs in the T-NOVA marketplace, specifying SLA level and price.
Trade-off	Not such a semantic expressivity as OWL-S.
Decision	The main reason for selecting USDL are the following features: - Business oriented: including price, SLA features. - Friendly GUI editor for the provider. - Simple format. OWL-S, WSMO and SA-WSDL describe services through semantic information, but not common vocabulary and taxonomies for defining business aspect like the license model.

3.8. Future work

In this section the first steps for the service description schema implementation have been presented. Once the business service catalog is finalised it will be integrated with the rest of T-NOVA modules, mainly the dashboard, by means of the "service description" and "service selection" GUIs. It is expected that this development work will be one of the first stages of the marketplace for demonstration purposes.

4. Brokerage Module

4.1. Introduction

In order to facilitate competition among Function Providers (FPs) a novel brokerage platform is designed. It will allow i) the T-NOVA customers to search for available offerings ii) trading between the third-party function developers (FPs) and the SP, in order to find the best price for the VNFs that will be part of each T-NOVA Network Service.

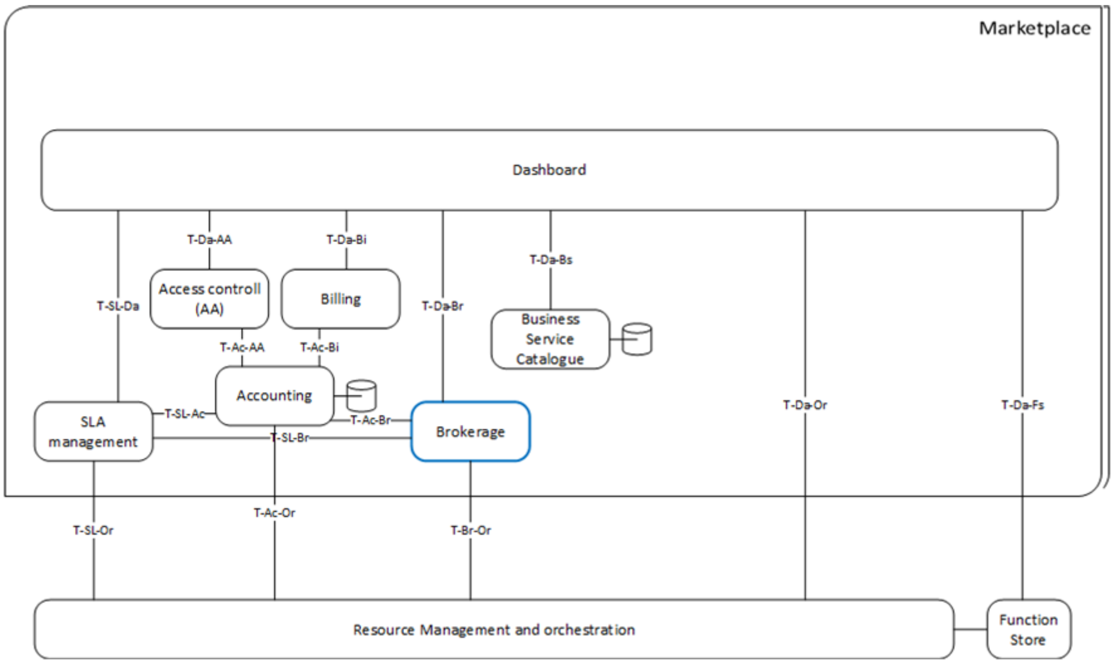


Figure 4-1 Brokerage module and its interfaces in the marketplace architecture

4.2. Requirements overview and rationale

The requirements for the T-NOVA Brokerage module gathered in the specification task are related to the need to provide a brokering mechanism that can support Services and Functions while on the same time to optimize the decision. [3] gathers the requirements for the Brokerage module.

Req. id	Use Case	Domain	Requirement Name	Requirement Description	Justification of Requirement
B.1	UC1.3	Management & Orchestration	Service Catalog	The Brokerage SHALL be able to communicate with Service /Function provider	The Brokerage must be able to negotiate with the Service/Function provider the necessary details
B.2	UC1.3	Operational	Trading/Bidding	The Brokerage SHALL be able to perform auctions among Service provider and Function providers	The Brokerage must be able to initiate auctions whenever it is required.
B.3	UC1.3	Management & Orchestration	Service Provision	The Brokerage SHALL be able to provide new service offerings to the dashboard	The Brokerage must be able to provide new services to in the Service catalog in order the Customer to be able to

					browse/select them in the dashboard.

Table 4-1 Brokerage requirements

4.3. State of the Art for T-NOVA brokerage

Trading is defined as the process of exchanging goods or services in a market. This exchange can be performed directly between goods and services (i.e., bartering) or using a medium of exchange (e.g., money). Similarly, VNF trading is the process of exchanging VNFs, which can be performed based on economic exchange.

The BonFIRE project which built a federation of cloud-based tesbeds for Future Internet Research and Experimentation, introduced a cloud brokerage system as a service for experimenters to hide the heterogeneity of the underlying cloud infrastructure. In particular, the BonFIRE enactor [26] mediates between the Resource Manager and different cloud testbeds, depending on the needs of experiments. It is the role of the enactor to hide the small differences between the various testbeds from the Resource Manager component. This broker concept differs with the broker in T-NOVA. In fact, it is more related to the orchestrator function than with the commercial broker of the T-NOVA marketplace.

There are also in the state of the art some solutions for brokering in radio spectrum, which differ from T-NOVA brokerage scenario, but which may be applicable in some economic aspects. One example is ICT-COGEU [27] defined new methodologies for equipment certification and compliance addressing coexistence with several European standards. It considered a centralized topology with a broker entity to trade with potential players. The resources broker controls the amount of bandwidth and power assigned to each user in order to keep the desired QoS (Quality of Service) and interference below the regulatory limits. In the ICT-COGEU reference model, the centralized broker is an intermediary between the geolocation database (resources information supplier) and players that negotiate the resources on behalf of the users. The ICT-COGEU broker is in charge of assigning the access to the resources under a real time secondary spectrum market regime. It incorporates a process of optimally allocating spectrum to secondary systems taking into account matching optimization methods, spectrum pricing and spectrum auction methods. A spectrum broker allocation process is designed and presented for an efficient radio resources allocation in the ICT-COGEU secondary spectrum market. The brokering of RF spectrum is not directly applicable to T-NOVA, since considering VNFs, one VNF can be use at the same time as part of different services.

CompatibleOne [28] is an open source project bringing together industry and academic leaders, innovative technology startups and open source community expertise providing a cloud brokerage. CompatibleOne is open to all, the results of which can be freely reused, modified and distributed. The availability of the entire code base and its documentation under open source license makes possible not only the adoption of the software but also aims at encouraging participation from new contributors in order to enrich and enhance the existing code base. CompatibleOne offers a simple and unique interface allowing for the description of user cloud computing needs, in terms of resources, and their subsequent provisioning on the most appropriate cloud provider. Resource descriptions may cover the complete cloud computing paradigm ranging from complete applications (SaaS), through development platforms (PaaS) down to low level compute and storage defined infrastructure (IaaS).

In addition, several project brokering websites bring together free agents, independent consultants and contractors with potential customers or clients. A project brokering website [29] acts as an intermediary between the contractor and the client. Generally, the project brokering website itself receives a fee for creating an opportunity for two entities to enter into an agreement to conduct business. Project brokers or project brokering websites offer exposure to contractors. In addition, some project brokering sites provide a platform so that the two business parties (contractor and client) can keep track of their communication, milestones, and schedule payments for services rendered. Some Project Brokering sites provide conflict resolution in the event that the business relationship between contractor and client is not a good match and the two parties want to end their business relationship.

Finally, ICT-Broker@Cloud [30]takes up the challenge of researching and developing solutions with respect to some of the most valuable and technically demanding types of brokerage capabilities foreseen: continuous quality assurance and optimisation. The starting point is to develop a through understanding of the functional and non-functional requirements that our brokerage framework should address, and implications with respect to integrating such a framework in enterprise cloud service delivery platforms. An additional aim of this project is to confirm and to update our present understanding of the relevant theoretical frameworks, techniques and open source tools that can be employed as the basis for the development of the framework components.

4.4. T-NOVA brokerage framework

While the provision of VNFs encompasses several system functionalities, VNF trading can be regarded as one part of the process that deals with the economic aspects. The trading process determines all the issues related with VNFs selling and buying (e.g., direct trading between service provider and function provider or via a brokerage module), while pricing is a major issue that determines the value (or worth) of the VNFs to the service provider and the function provider. Another issue is the competition/cooperation among function and service providers, as well as customers involved in VNF trading. Depending on the VNF trading model, the VNF access may require permission through the cooperation of service provider and function provider, through a payment process. To determine the optimal network function provision during the trading process, optimization and decision theory techniques can be used.

In T-NOVA the trading is performed by T-NOVA brokerage module typically between the SP and FPs when the SP is an external stakeholder issuing the T-NOVA system as a particular case of T-NOVA customer. In case the T-NOVA operator is acting as

Service Provider, the trading would be performed between FP and customer directly, acquiring VNFs to compose its T-NOVA Network Service.

VNF pricing plays an important role in trading since it indicates the value of VNF to both the function and service providers. For the service provider, the price paid to the function provider would depend on the satisfaction achieved through the usage of that VNF. For the function provider, the price determines its revenue (and hence profit). If VNF price is high, the satisfaction of the service provider is reduced while the revenue of the function provider is increased. The VNF price should be set based on the VNF demand of the service provider and the VNF supply of the function provider. Also, competition between service and function providers will affect price setting.

The SLA agreed between Function Provider (FP) and Service Provider (FP) plays a determinant role in the results that will be provided. More specifically a VNF can be offered by a FP with different SLA levels, according to different prices that will be negotiated through the trading mechanisms implemented by the brokerage module. After the Brokerage finalise the Brokering based on the SLA agreed it informs the respective stakeholders on the agreement. Finally, the brokerage after a request stemming from the SP can request a modification on the SLA of the VNF.

4.4.1. Modeling of the T-NOVA brokering approach

According to the trading process (i.e. auction-based algorithm in Table 4-2) T-NOVA brokerage module determines the optimal allocation solution, considering the maximization of Service Provider (SP) income. For this, the brokerage module undertakes the trading mechanism that collects bids from Function Providers (FPs), in order to lease the VNFs to the T-NOVA customers, through the SP. The brokerage module computes the assigning solution through this mechanism together with price and SLA per network service.

Furthermore, when the auction-based algorithm is followed, the sellers (i.e. FPs) that are denoted as N = {1,2,…,n} lease the VNFs that denoted as S = {1,2,…,s} to I = {1,2,…,i} customers, through the SP. Each buyer “i” is able to lease *xs* VNFs for a specific time period *ti*, by reporting a price *Pi(b) = {xs, ti}* (i.e. bid price of VNFs in a specific time), while the FPs lease yn VNFs for a specific time ti, by reporting a price*Pn(s) = {yn, ti}* (i.e. asking price of VNFs in a specific time). Finally, $x_{i,n}$ is the quantity, which is leased by “i” customer from the brokerage module. The pair (i,s) in the pseudo-code of Table 4-2 represents possible combinations of solutions, regarding “s” VNF to “i” customer. In case that SP benefit has to be maximized, an optimization problem is formulated as follows, based on linear programming, i.e.the following equation:

$$max\colon \sum_{i=1}^{i}\sum_{n=1}^{n} x_{i,n} t_i \left(P_i^{(b)} - P_n^{(s)}\right) \quad (1)$$

```
1: **Inputs: VNFs**, $Demand_{SP}$
2: Access service catalog store
3: Estimate the initial price per VNF
4: Create and advertise price-portfolio
5: Receive FPs bids $P^{(b)} = \{P_1^{(b)},\ldots, P_I^{(b)}\}$, where $P_i^{(b)} = \{x_s, t_i\}$
6: for all Bids do
7:     Sort $P_i^{(b)}$ in descending order based on price and create the auction-portfolio
8: end for
9: Calculate the highest valuation S[i,s] for all VNFs (i,s) ∍ {1, 2,..., s}
10: set $S_{optimal}$ = S[i,s] //Random solution for algorithm initiation
11: for each bid $P^{(b)}$do //Iteration process in order to find the best solution
12:     if (S[i,s]) ≤ (S[i+1, s+1]) // Check if the current solution is better or not to the neighbor solution
13:         then save the new allocation solution (S[i+1, s+1]) to the best found
14:     end if
15: end for
16: returnBest Solution
```

Table 4-2Algorithm Pseudo-Code

The Brokering T-NOVA system will support several cases:

1. The T-NOVA customer that visits the marketplace seek/request for the preferred VNFs. In this case, the trading algorithm is the fixed-price because more than one user can utilize a NF and there is no need for further negotiation (i.e. auctions).
2. In case that there is more than one FPs that perform the same function, the brokerage module may initiate an auction (or keep fixed price as the offer is higher to demand) in order the different FPs to have the same opportunities to sell the application. In this occasion, the SP has the possibility to select the either the cheapest or the most expensive function depending their requirements and desired SLA.

In this case, a FP may wish to sell the service/NF in lower price if he reach his payoff and he prefer to keep sell and increase his benefit in lower rate, despite not having any profit.

Finally yet importantly, is the case where the different FPs have to negotiate with the T-NOVA Service provider in order to add the NFs to Function store, either exploiting fixed-price or auctions. The trading method that will be followed depends on the incentives of each FP.

4.5. Architecture of the brokerage module

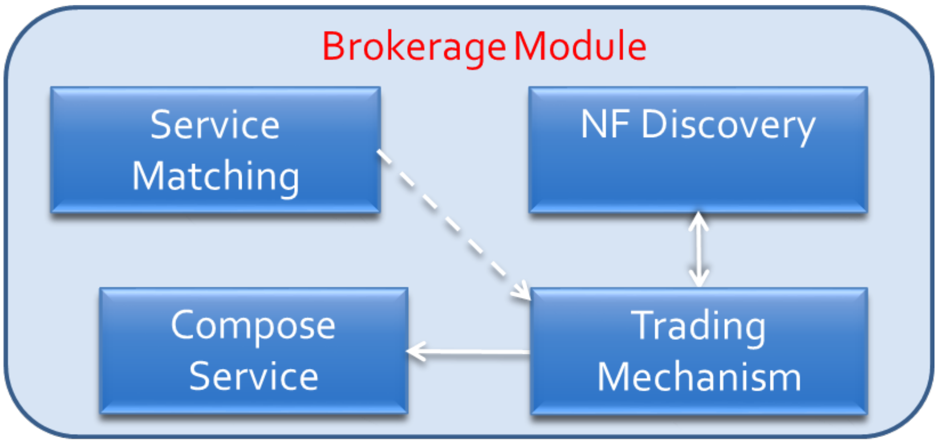


Figure 4-2 Brokerage module internal architecture

The overall architecture of the brokerage module is depicted in Figure 4-2. It consists of four main modules that are used in various interactions, as shown in the Figure above:

- The NF discovery module is exploited in the entire process to enable the brokerage module searching for the requested VNFs.
- The service composition is used in the case that there is not any ready available service and SP will ask for VNFs to create a new service.
- The service matching included the process of the brokerage module towards finding the required services in the business service catalog.
- The trading mechanism that is used in order to communicate with the brokering stakeholders and perform the Auctioning.

According to the proposed mechanism, the T-NOVA customers browse the offerings from the Service catalog that match his requirements. If the requested Service/Function supports Brokering the Service Matching internal modules will try to fulfill the criteria set by the Customer and if not, a new service composition will take place in the compose service module. Furthermore, the broker initiates the appropriate bid/trading policies according to the T-NOVA customer request inside the trading mechanism in collaboration with the NF Discovery.

4.5.1. Relation with other T-NOVA components

As it is represented in Figure 4-1, the brokerage interfaces with other T-NOVA components, are the following:

(T-Da-Br) The brokerage will interface the dashboard to to facilitalte the trading among prcoduere among SP and FPs.

(T-Br-Ac) With this interface the brokerage module provide to the accounting system the appropriate information related to service selections and price.

(T-Br-Sl) This interface is exploited in order for the brokerage module to provide information to the SLA management module regarding the SLA agreed between SP and FPs as a result of the trading process.

(T-Br-Or) This interface is required in order for the brokerage module to retrieve information about the available VNFs.

4.5.2. API definition

The basic operations that the brokerage API will support are the following:

- Operation invoked by the Dashboard (T-Da-Br):
  - new_ServiceRequest (POST)
    - Creates a new service request by the Customer in the Brokerage module.
  - new_VNF_request (POST)
    - Creates a new VNF request by the SP in the Brokerage module.
  - get_VNF_request (GET)
    - Obtains VNF request by the FPs.
  - Post_VNF_offer (POST)
    - Creates a price offer by the FP for a VNF request in the Brokerage module.
  - Put_VNF_offer (POST)
    - Updates a price offer by the FP for a VNF request in the Brokerage module.
  - Get_VNF_offer (GET)
    - Obtains price offers by the SP for a previous VNF request.

On the other hand the brokerage module will call the API of the following modules:

- The brokerage will perform POST/PUT operation over the accounting module API *to* send the selected service and price to accounting module.
- The brokerage will perform POST/PUT operation to send for a specific service id, the agreed SLA.
- The Brokerage module will make GET operation over the correspondent orchestrator API in order to retrieve available VNFs.
- (If way may consider the option that a service request can be done by means of the Brokerage, the Brokerage will have to retrieve information from the business service catalog (making GET operations to business catalog), to find if a service that fulfills customer's service request).

4.6. Candidate technologies and rationale

This section elaborates on candidate technologies that could be used to implement the T-NOVA brokerage module. More specifically, the externalization of the network functions requires a consistent platform, where the VNFs can be deployed and interoperate with other network functions and the network itself. Several approaches elaborate on implementation and deployment frameworks for enabling VNFs with open Network as a Service platform (NaaS) [31]. The management of the VNFs instances can be centralized to exploit most of the advantages and the flexibility provided by OpenNaaS. OpenNaaS is an open source framework that could be possible exploited for the T-NOVA case, offering tools to manage the resources of a network, enabling different stakeholders to contribute and benefit from a common

NaaS software stack-oriented applications and services. Its design allows the deployment of VNFs as a service and offers functionalities for managing their lifecycle. The elements loaded in OpenNaaS contain an abstraction model that stores all the required information of the network function or resource and a set of capabilities that allows accessing the data of the model. Besides, OpenNaaS allows the creation of specific Web Service interfaces for VNF management. OpenNaaS is proposed as an open-source enabler to implement and deploy VNFs. The main advantages of this approach are numerous. First, it is a robust, open-source, and consolidated platform for network management. It allows having a complete view of the entire network and interacts directly with the data plane through its Hardware Abstraction Layer (HAL) and southbound interfaces. Moreover, the data and the actions of each network function can be mapped in OpenNaaS, where the structure of the data managed represents the model, and the capabilities of each function give access to the model and to the data saved in it. Finally, these capabilities can be implemented with Web Services through a REST (Representational State Transfer) interfaces.
It should be mentioned that up to now, the mainstream of brokering studies has been done on an economic level and the results usually do not take into account technical constraints. Regarding the proof-of-concept, no real-time brokerage demonstration has been made or reported up to now in order to research the different challenges enumerated above in realistic conditions. Furthermore, dynamic brokerage platforms have been developed in several projects funded by EC during the last years, such as ICT-SENDORA, ICT-ARGORN and ICT-E3. However, none of them addressed specifically the technical specifications to trade the VNFs. Finally, the brokerage module proposed by ICT-COGEU project could be exploited as the basis for the development of the T-NOVA brokerage module. This brokerage module considered a centralized topology with a broker entity to trade with potential players. The ICT-COGEU broker is in charge of assigning the access to the resources under a real time secondary spectrum market regime. This approach is a very close relation with the T-NOVA rationale.

4.7. Technology selections summary

This section summaries in the following table for the implementation of brokerage module in T-NOVA Marketplace the alternative options available closer to T-NOVA, the requirements [2] that they may satisfy, the trade-offs and the justification of the decision taken at this point in time (e.g. the rationale behind the selection).

Most of the alternatives are still under evaluation so final details of the technologies ultimately used to implement T-NOVA Brokerage will be provided in the next set of deliverables.

Topic	Trading and Brokering Platforms
	▪ ICT-SENDORA, ICT-ARGORN, ICT-E3 do not address specifically the technical specifications to trade the services via a brokerage module through a marketplace

Alternatives	▪ ICT-COGEU proposed a centralized broker as an intermediary between resources information supplier and players that negotiate the resources on behalf of the users. ICT-COGEU approach could be adapated for the development of the T-NOVA brokerage module.
Requirements Related	B.2 The Brokerage SHALL be able to perform auctions of VNFs among Service provider and Function providers B.3The Brokerage SHALL be able to provide new service offerings to the dashboard
Trade-off	Building software from scratch.
Decision	A customized software development considering only the needs of T-NOVA will ease the integration with the rest of the modules. While none of the existing solutions is working in the T-NOVA Model

4.8. Future work

In this section several brokerage existng solutions have been presented, as well as a first approach to a pseudo-code algorithm for the T-NOVA trading scenario, in which the brokerage of VNFs will be mainly performed based of different SLA levels. Nevertheless further research work may be needed before implementation to consider some other particularities when performing auctioning among VNFs that are different for the existing solutions in the state of the art.

5. USERS DASHBOARD

5.1. Introduction

The dashboard constitutes the T-NOVA system front-end, offered to the customer, the Service Provider (SP) and the Function Providers (FPs) for service consumption, discovery, interaction, publication, etc. In order for the dashboard to be as up-to-date as possible and terminal-agnostic, a web-based implementation has been selected.

Furthermore, the dashboard shall be able to meet and, if necessary, to adapt to the specific stakeholder's needs/requirements as much as possible providing the best usage experience. T-NOVA Dashboard will allow personalization for a variety of settings such as interface, appearance and content according each dashboard user profile configuration.

This section describes the Graphical User-Interface (GUI) of the dashboard, the technical design of the dashboard and the implementation framework. An initial implementation of the GUI is presented in Annex 9.1, which has been implemented considering the requirements gathered in previous stages. It has to be remarked that it is only the front-end for the stakeholders (one dashboard view per stakeholder: SP, FP and customer) to interact with the whole system; the components below that it will interact with have not been developed yet at this phase of the task.

5.2. Requirements overview and consolidation

The main design decision gathered from the requirements in [3] has been to provide a common dashboard with different customized views based on different roles.

In Table 5-1 we collect the dashboard requirements that are common for the three views.

Req. id	Use Case	Domain	Requirement Name	Requirement Description	Justification of Requirement
D.1	UC1	Security - AA	Authentication and access control	The dashboard SHALL provide a "login in" page for the different stakeholders to be authenticated	Stakeholders interacting with the T-NOVA system should be authenticated and authorised in order to be able to browse the Business Service Catalog, issue SLA requests, or upload NFVs
D.2	UC1.1	Security - AA	Authentication and access control	The "login" page in the SHALL offer to the different stakeholders means to use (username, password, OpenID, Google API) for authentication	Stakeholders interacting with the T-NOVA system should be able to use different authentication techniques to access T-NOVA

Req. id	Use Case	Domain	Requirement Name	Requirement Description	Justification of Requirement
D.3	UC1.2	Security - AA	Authentication and access control	The "login" page in the SHALL offer to the different stakeholders means to remember credentials when logging on	Stakeholders interacting with the T-NOVA system should not be obliged to insert credentials when accessing again the system
D.4	UC1.1	Management & Orchestration	Web access	The Dashboard SHALL be accessible to authorized users via the Internet	The Dashboard will provide the necessary interface in order to be viewed over the Internet
D.5	UC1.1	Management & Orchestration	Parallel Access	The Dashboard SHOULD provide multiple users login and no less than 10	The Parallel access will provide the necessary tools for every user to be able to provide his content
D.6	UC1.1	Management & Orchestration	Availability	The Dashboard SHOULD be available 24/7 365 days per year	The Dashboard must be always on in order to control every part of the T-NOVA infrastructure.
D.7	UC1.1	Management & Orchestration	Operation	The Dashboard MUST be as light way as possible in the Server side	The Dashboard must be able to take into account the processing power of the stakeholder accessing it.
D.8	UC1.1-UC.1.2	Management & Orchestration	Open source	The Dashboard MUST be Open source	The Dashboard must be Open source

Table 5-1 Dashboard basic requirements

5.3. T-NOVA Dashboard

As explained before (section 2) the T-NOVA Marketplace consists of a Dashboard front-edge, and several microservices that are used in order to have independently implementation of the several Marketplace modules.

The main idea is to create a modern and powerful tool to support the different microservices that will be hosted inside the Marketplace, as well as providing a user-friendly environment for the different stakeholders' views. In order to achieve this, state-of-the-art technologies are required in order to meet the following targets:

- Support various devices and not just Personal Computers. In order to be able to use mobile devices or tablets, etc., we need the dashboard GUI to be as lightweight as possible when choosing the technologies for implementation.
- Minimization of the creation of overhead traffic created from the dashboard client minimizing as much as possible the data for calculations and processing that the dashboard client will react.

To overcome these requirements, the T-NOVA Dashboard is decided to be implemented as a web-based application that will be accessible on devices using a web browser such as computers, tablets and mobile phones. The advantage of creating an application within a browser environment is that the application is platform independent, and the only rule that exists is that it must follow the web standard specifications from W3C.

During the authentication stage, all stakeholders share a common layout provided by the dashboard GUI that displays the generic graphical interface composed by the basic controls that enable stakeholder specific authentication. Once authenticated to the access control module, every stakeholder will be able to customize the overall experience according to a set of preferences and his profile.

5.4. Architecture

The overall Dashboard GUI conforms a number of RestAPIs that will be used in the Inter-Marketplace communication. More specifically every module implemented inside the Marketplace that wants to expose a functionality to the Dashboard GUI needs to have a RestAPI that provides the respective functionality (e.g. Accounting modules Providing Available Users). In this way, the Dashboard implemented inside the T-NOVA is just an "GUI enabler" for the various modules of the Marketplace while on the same time the implementation technique selected can be used in modular FP's or SP's dashboard implementation.

Figure 5-1 presents how the various Stakeholders will interact with the T-NOVA marketplace. In the schema, the NGINX/Reverse Proxy is a Web server that exposes the Marketplace and is able to redirect all requests to the respective Module. Furthermore, every module implemented inside the Marketplace (e.g.Module X) can communicate through the NGINX to the respective micro service module inside the Marketplace.

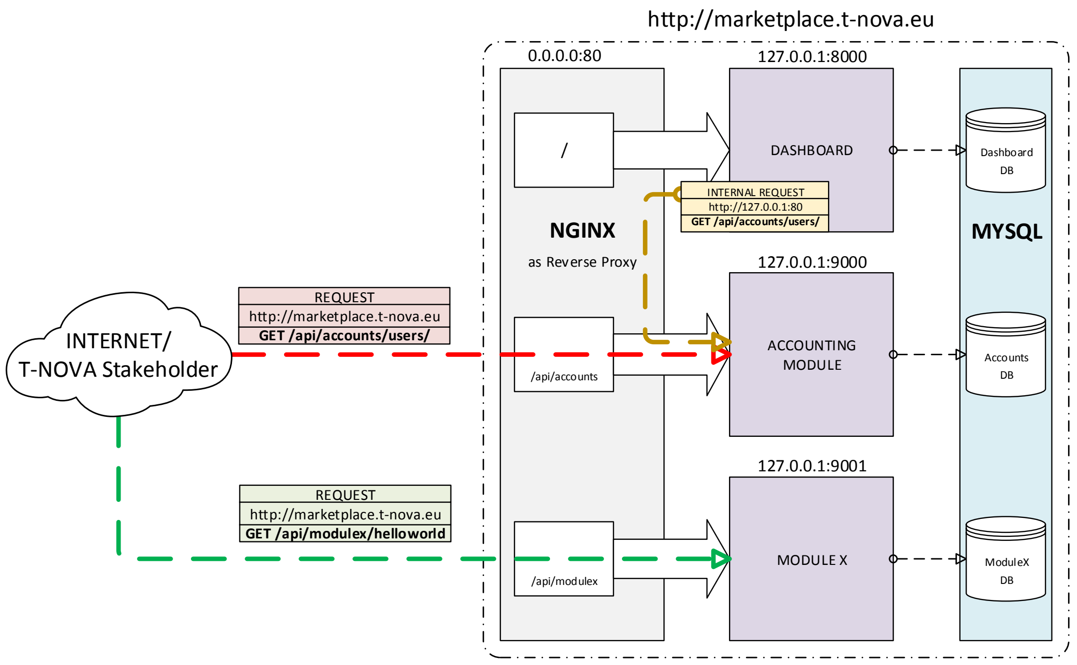


Figure 5-1 Interaction of the T-NOVA stakeholders through the Dashboard

Figure 5-2 provides the view of the different micro services interfacing the rest of T-NOVA modules. The details about them are explained in the corresponding sections devoted to the other modules.

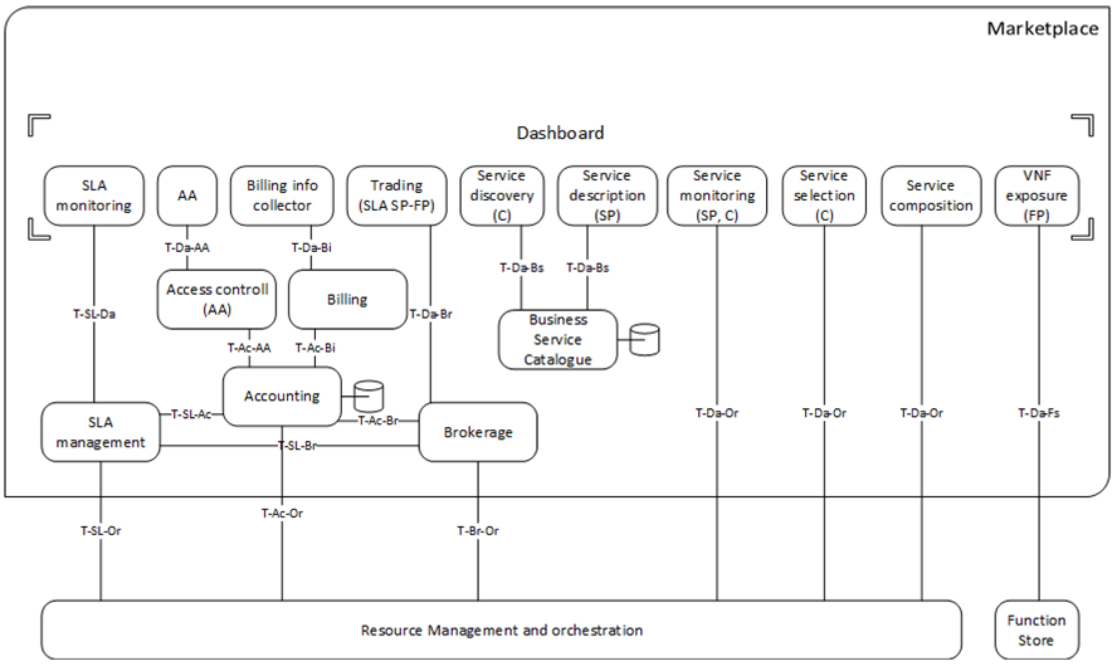
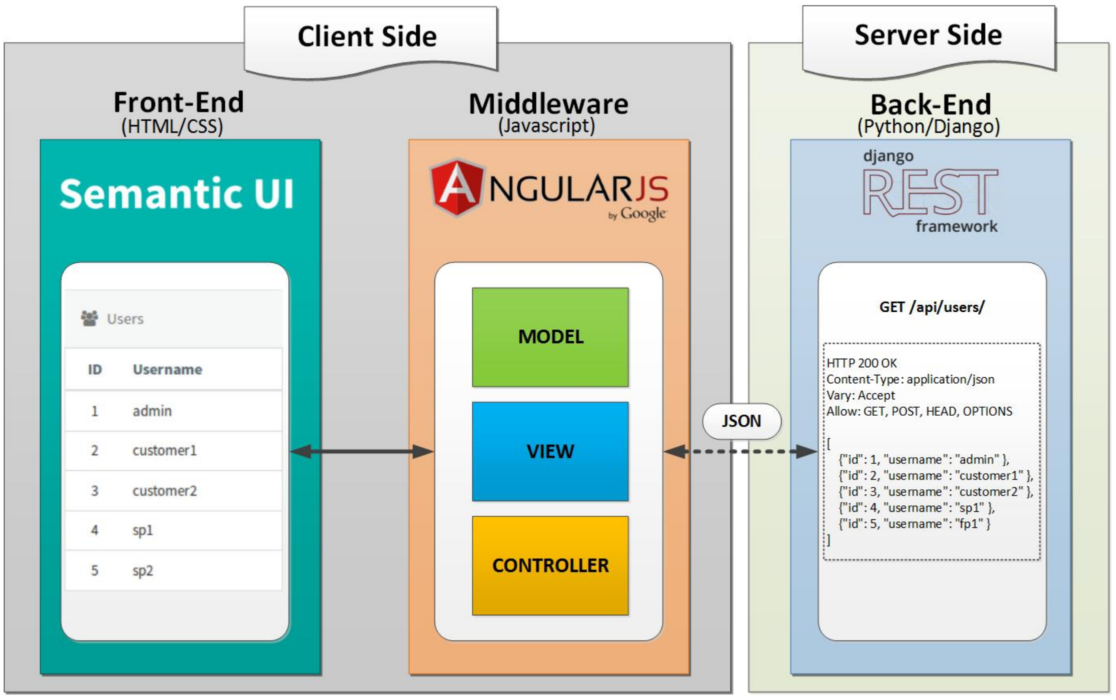


Figure 5-2 Dashboard High-level components

5.5. Candidate technologies and rationale

The first initial architecture is present in the following schema presenting the main technologies selected in order to enable the microservices implementation that has been selected for the T-NOVA marketplace environment.



ID	Username
1	admin
2	customer1
3	customer2
4	sp1
5	sp2

GET /api/users/

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, POST, HEAD, OPTIONS

[
   {"id": 1, "username": "admin" },
   {"id": 2, "username": "customer1" },
   {"id": 3, "username": "customer2" },
   {"id": 4, "username": "sp1" },
   {"id": 5, "username": "fp1" }
]
```

Figure 5-3 Dashboard GUI implementation technologies

5.5.1. Angularjs

AngularJS [32] is an open-source MVC (model-view-controller) framework that assists with creating dynamic web applications, which consist of HTML pages, CSS, and JavaScript on the client side, and its deal partner with any server technology. It lets you extend HTML's syntax to express your application's components clearly and succinctly, it also declaring dynamic views in web-applications and encapsulates the behaviour of your application in controllers. It focuses on strong separation of concerns (MVC) and dependency injection to encourage creating maintainable (and testable) modules that can be integrated to develop rich client side functionality.

5.5.2. Django Rest Framework

Django REST [33] framework is a powerful and flexible toolkit for Django framework that makes it easy to build Web APIs. Provides powerful model serialization, display data using standard function based views, or get granular with powerful class based views for functionality that is more complex.

5.5.3. JSON Web Token (JWT)

JSON Web Token [34] is a relatively new token format used in space-constrained environments such as HTTP Authorization headers. JWT is architected as a method for transferring security claims based between parties.

There are two different ways of implementing server side authentication for apps with a frontend and an API:

- The most adopted one is Cookie-Based Authentication that uses server side cookies to authenticate the user on every request.
- A newer approach, Token-Based Authentication, relies on a signed token that is sent to the server on each request.

5.5.3.1. Token based vs. Cookiebased

While the cookie based authorization's is used as the de-facto standard for communication between client server, this technology cannot be used when we have a multi-domain networks, taking this into account a new token-based technology [34] was proposed that allows to make AJAX/REST calls to other domains, by including the user information in HTTP header of the http request. The main benefits of the new approach are presented below:

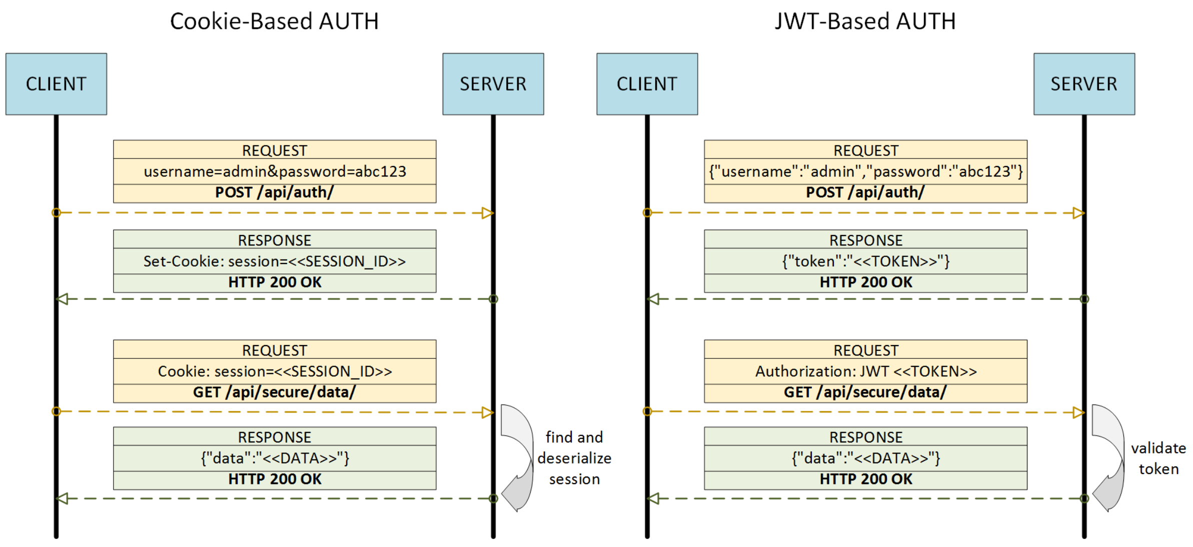


Figure 5-4 Cookie-Based vs JWT-Based Authentication

- **Stateless**: there is no need to keep a session store; the token is a self-contained entity that conveys all the user information. The rest of the state lives in cookies or local storage on the client side.
- **Decoupling**: The application is not tied to a particular authentication scheme. The token might be generated anywhere, hence the created API can be called from anywhere with a single way of authenticating those calls.
- **Mobile ready**: Working with cookies on native platform (iOS, Android, Windows 8, etc.) is not ideal when consuming a secure API (We need to deal with cookie containers) by adopting a token-based approach simplifies this a lot.
- **CSRF**: since you are not relying on cookies, you do not need to protect against cross-site requests (e.g. it would not be possible to <iframe> your site, generate a POST request and re-use the existing authentication cookie because there will be none).
- **Performance**: we are not presenting any hard performance benchmarks here, but a network roundtrip (e.g. finding a session on database) is likely to take more time than calculating an HMACSHA256 to validate a token and parsing its contents.
- **Standard-based**: your API could accept a standard JSON Web Token (JWT). This is a standard and there are multiple backend libraries (.NET, Ruby, Java, Python, and PHP) and companies backing their infrastructure (e.g. Firebase, Google, Microsoft). As an example, Firebase allows their customers to use any authentication mechanism, as long as you generate a JWT with certain pre-defined properties, and signed with the shared secret to call their API.

5.6. Access Control Module

In this section we define the groups and permissions to be implemented in the Access Control Module. In order the implementation to be as modular as possible, we have enabled multiple groups that can be assigned to a stakeholder.

5.6.1.1. Stakeholders Permissions

The following table gathers the various T-NOVA roles and their perimisionw.

Group	view	add/edit	delete
Administrator (T-NOVA operator)	yes	yes	yes
Customer	own	own	no
Service Provider	own	own	no
Function Provider	own	own	no

Table 5-2 Stakeholders Permissions

Only “Administrator” group can view/add/edit/delete all various stakeholders. Other groups can only view and manage their own profile.

5.6.1.2. Services Permissions

The following table gathers the various T-NOVA Service Permisions.

Group	view	add/edit	delete
Administrator	yes	yes	yes
Customer	yes	no	no
Service Provider	own	own	own
Function Provider	no	no	no

Table 5-3 Services permissions

Administrator group can view/add/edit/delete all services. “Service Provider” group can only view/add/edit their own services. `Customer` groups can only view the services. “Function Provider” group have no permission on services.

5.6.1.3. Functions Permissions

The following table gathers the various T-NOVA Functions Permisions.

Group	view	add/edit	delete
Administrator	yes	yes	yes
Customer	yes	no	no
Service Provider	no	no	no
Function Provider	own	own	own

Table 5-4 Functions Permissions

Administrator group can view/add/edit/delete all functions. "Function Provide" group can only view/add/edit their own functions. `Customer` groups can only view the functions. "Service Provider" group have no permissions on functions.

5.7. Technology selection summary

This section summaries in the following table for the implementation of T-NOVA Dashboard, the alternative options available at this point in time, the requirements [2] that they may satisfy, the trade-offs and the justification of the decision taken (e.g. the rationale behind the selection).

Final details of the technologies ultimately used to implement T-NOVA Dashboard will be provided in the next set of deliverables.

Topic	**Software Architectural Pattern for the User Interface**
Choice	• Browser-Centric Web Application (BSWA) approach embeds all the functional parts (e.g. Scripts) on the client's side and processes them on the client's Internet browser. The advantage of BCWA architecture is the faster response time and less overhead (e.g. Data, Processing Power) on the web server. Addionally is ideal when the page elements need to be changed without the need to contact the database.
Alternatives	• Server-Centric Web Application (SCWA) use a Server in order to collect/manage data and serve the HTML to the client. The disadvantage of this approach is the page post back introduce processing overhead that can decrease the performance and force the user to wait for the page to be processed and recreated. More specifically, once the page is posted back to the server, the client must wait for the server to process the request and send the page back to

	the client.
Requirements Related	D.8 The Dashboard MUST be Open source D.7The Dashboard MUST be as light way as possible in the Server side
Trade-off	Browser-Centric Web Application
Decision	The Browser-Centric Web Application architecture provides the necessary interaction for users to interact with the dashboard and concurrently offers the necessary tools for third party applications (e.g. with RESTful API) to communicate this the Dashboard. Additionally this architecture is much more light-wave for the server due to the fact that everything running on client side.

Topic	**Dashboard Front-end**
Choices **Alternatives**	▪ Semantic-UI a newly created User interface with rising community ▪ Bootstrap is the most popular CSS Framework
Requirements Related	D.1 The dashboard SHALL provide a “login in” page for the different stakeholders to be authenticated D.8 The Dashboard MUST be open source
Trade-off	Semantic-UI
Decision	Semantic-UIdesign is better and cleaner. It’s easy to use, strict coding, useful components, lightweight and very well documented

Topic	**Dashboard Middleware**
Choices	• AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you would otherwise have to write. In addition, it all happens within the browser, making it an ideal partner with any

Alternatives	server technology. • jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript. • Ember.js is a framework for creating ambitious web applications. It main focus is to give you tools, powerful enough to reduce the amount of code you write. Ember incorporates many common idioms and frees you from reinventing the wheel. Similar to other opinionated frameworks, Ember values convention over configuration.
Requirements Related	D.8 The Dashboard MUST be Open source
Trade-off	AngularJS
Decision	AngularJS is a Web framework that provides all the necessary tools for using it in an interconnected way. The major benefits of AngularJS are: • Integration with Existing Apps • Simplicity • Extensibility

Topic	Dashboard Back-end
Choices Alternatives	• Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. • Flask [ref] is a micro framework for Python depends on two libraries. It also binds to a few common standard library packages such as logging. • Symfony is a set of reusable PHP components and a PHP framework for web projects. Speeds up the creation and maintenance of PHP web applications. • YII is a framework developed in PHP and widely used for building highly secured Web applications
Requirements Related	D.8 The Dashboard MUST be Open source

Trade-off	Django
Decision	Django is a powerful and flexible framework is that makes it easy to build Web Applications. Django have plenty of plugins, speeds up the creation of the web applications and easy support REST API technology.

5.8. Future work

In this section we have presented an initial view of the Dashboard in the context of the T-NOVA. The initial version of Dashboard provides all the necessary expansion modules in order every module inside the Marketplace to be able to provide a RESTful API with the exposed function that will be visualised inside the T-NOVA Dashboard. The Dashboard as seen in the Annex is providing all the necessary parts that can be used in an initial internal evaluation providing all the future demo functionalities.

6. SLA MANAGEMENT

6.1. Introduction

This section summarizes the progress so far on the implementation of the T-NOVA SLA ecosystem, with the later objective of developing the most suitable compoments for SLA management, including the study of:

- Mechanisms for SLAs that can support the formal definition and management of the relationships between the T-NOVA stakeholders – SLA template, negotiation, agreement.
- SLA management information for later billing and conciliation, depending on the terms and conditions gathered in the SLA and on whether this SLA has been met by all parties or not.
- Management of the monitoring information from the orchestrator whether the committed SLA has been met or not, taking the necessary actions, which can lead to simple reports or additional credit/charges in the billing account for this customer.

Figure 6-1 remarks the SLA management module and its interfaces in the marketplace acchitecture diagram.

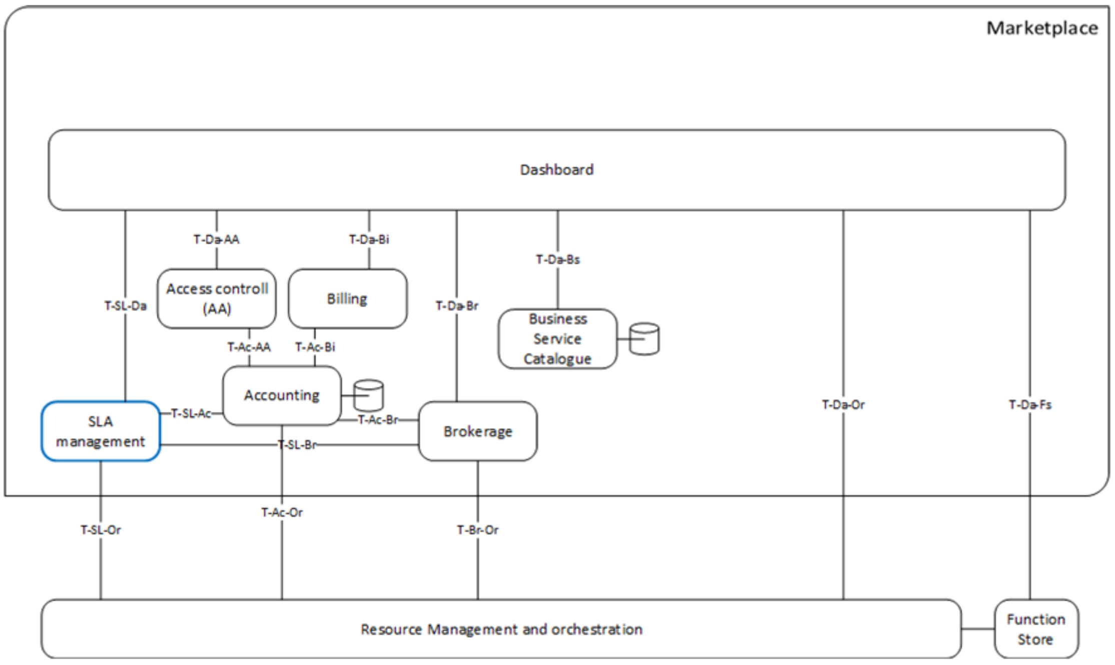


Figure 6-1 SLA management module in T-NOVA Marketplace

6.2. Requirements overview and consolidation

The requirements for the T-NOVA SLA management module gathered in the specification task are related to the need to provide mechanisms to get an agreement presented and agreed, store all the SLA agreements, to inform the orchestrator about the SLA thresholds that need to be achieve for each service, and to know all the SLA fulfilment to inform the billing system for possible penalties. Table 6-1 gathers the requirements for the SLA management module.

Req. id	Domain	Requirement Name	Requirement Description
SLA.1	Service continuity	SLA information customer-SP storage	The SLA management module SHALL store all the SLA agreements between a customer and the SP for each service.
SLA.2	Service continuity	SLA information SP-FPs storage	The SLA management module SHALL store all the SLA agreements between the SP and the FPs for each VNF.
SLA.3	Management & Orchestration, Operations, Service Continuity	SLA – orchestrator interface	The SLA management module SHALL be connected to the orchestrator to let it know about the agreed SLA for each service. (When the SLA is not fulfilled the orchestrator will have to initiate the applicable action, e.g. rescaling)
SLA.4	Market / commercial operability	SLA fulfilment information storage (from the orchestrator)	The SLA management module SHALL store all the information about SLA fulfilment for eventual compensations or penalties for later billing.
SLA.5	Market / commercial operability	SLA – accounting/billing interface	The SLA management module SHALL be connected to the accounting/billing system to let it know about eventual compensations or penalties when the SLA has not been fulfilled for a specific service.
SLA.6	Service continuity	SLA visualisation by customer and SP	The SLA management module SHALL be connected to the Dasboard to allow a customer and SP to visualize SLA fulfilment information when requested.
SLA.7	Service Continuity	SLA procedure mechanisms	The SLA management module SHALL provide mechanisms to get an agreement presented and agreed between the parties

Table 6-1 T-NOVA SLA basic requirements [3]

Based on these requirements, the following sections present the further work done based on them, the study of previous solutions in the state of the art, and the progress to find the best suitable SLA framework to be implemented in T-NOVA.

6.3. State of the Art analysis for T-NOVA SLA

6.3.1. Other research projects

We have identified several recent research projects implementing SLA management framework:

- Cloud4SOA (FP7) [35] focused on resolving the semantic interoperability issues that exist in current clouds infrastructures and on introducing a user-centric approach for applications, which are built upon and deployed using cloud resources. To this end, Cloud4SOA aims to combine three fundamental and complementary computing paradigms, namely cloud computing, service Oriented Architectures (SOA) and lightweight semantics. The system developed in Cloud4SOA supports cloud-based application developers with multiplatform matchmaking, management, unified application and cloud monitoring and migration. It interconnects heterogeneous cloud offerings across different providers that share the same technology through the concept of adapter that provides a REST-based API for any-cloud access. Cloud4SOA SLA module allows SLA negotiation and enforcement to develop a framework enabling dynamic SLA negotiation and tools that enable cloud providers to analyse their offerings and performance and adapt the SLAs accordingly. The framework allows providers and customers to negotiate flexibly between standard and customized SLAs, while supporting business dynamics through business-performance related SLA metrics being monitored and analysed. To deal with an SLA in an automatic way, the SLA itself has to be expressed in a formalized way using an SLA specification language. In terms of technology baseline, Cloud4SOA provides a RESTful implementation of the WSAgreement standard. In the context of the project several languages for SLA specification have been reviewed (e.g. WSOL, WSLA, WSML etc.) but the WS-Agreement specification was selected as the most appropriate.
- Fed4FIRE Project (FP7) [36] aims at establishing a common federation framework by developing, adapting or adopting tools that support experiment lifecycle management, monitoring and trustworthiness. In relation to the SLA management, in Fed4FIRE there are several providers involved in an experiment and an SLA must be agreed between the experimenter and all those providers who adopt SLAs. Besides, facilities are heterogeneous and different resources and services can be offered within one facility. In order to avoid manual procedures as much as possible, an SLA can be based on a template describing the terms and conditions based on attributes. An SLA is agreed between the experimenter and every single testbed involved in an experiment offering SLAs. The federation does not ensure there is an SLA established with every testbed before the experiment begins. Every testbed having adopted SLAs will expose its SLAs and will manage its SLA commitments individually. With this approach, the federation only provides the set of tools for a testbed to implement SLA management. It is up to the testbed to use it or not.
- XIFI Project (FI-PPP) [17] establishes a unique marketplace for European players through the creation of a sustainable pan-European open federation of test infrastructures to overcome the current fragmentation and enable widespread and replicable commercial launch of Future Internet (FI) services and applications. The XIFI SLA manager provides mechanisms to support service level agreements management in the federated environment, based on WS-Agreement specification. The component allows the direct interaction among the different actors through the graphical user interface and is

designed to be part of XIFI portal. Based on SLA management standards, it covers the SLA lifecycle and allows the evolution of the lifecycle according to the project needs. The component relies on the knowledge of the federated environment and of the infrastructures in order to define service characteristics and QoS for the offered service. The end users, through this component, define and monitor SLAs. In this first release, the SLA Manager, as simple SLA lifecycle, supports these actions: i) the providers indicate the QoS metrics that can be monitored on their infrastructures ii) the users view the available metrics for the service and decide the boundaries which should monitor.

These projects address the SLA management in different envirorments so it will be a good input to consider them when designing the T-NOVA SLA ecosystem. However, they do not address the the specific particularities for the NFV business ecosystem that we are studying in T-NOVA which is explained in section 6.4.

Furthermore, automatic SLA SOTA is mainly in the scope of cloud but T-NOVA requires a combination of network functions and cloud. Cloud is not enough. Usually, in telecommunications, SLAs can be seen as the minimum service acceptance level a customer would agree to be delivered by a communication service provider, though they are usually vague, not end-to-end and unknown to the network [37] [38]. Moreover, they are not as dynamic or as automatically managed as T-NOVA requires.

6.3.2. Standarization bodies

6.3.2.1. ETSI

ETSI does not define at this stage a business perspective to manage the SLA relationships among the posible stakeholders in the NFV scheme, but identifies some requirementes for the final network service SLA [39]:

[Res. 5] The SLA shall specify the "metrics" to define the value and variability of "stability".

[Res. 6] The NFV shall support mechanisms to measure the following metrics and ensure that they are met per SLA:

- Maximum non-intentional packet loss rate.
- Maximum rate of non-intentional drops of stable calls or sessions (depending on the service).
- Maximum latency and delay parathion on a per-flow basis.
- Maximum time to detect and recover from faults aligned with the service continuity requirements
- Maximum failure rate of transactions that are valid and not made invalid by other transactions.

[Cont.1] The SLA shall discribe the level of service continuity required.

Moreover, in [40] ETSI gives a first approach of the different service quality metrics that will influence the final service quality level that the end-user will experiment. ETSI classifies these quality metrics in four groups:

- Virtual machine service quality metrics.
- Virtual network service quality metrics.
- Technology components offered as a Service (standalone VNFs).
- Orchestration service quality metrics.

Service Metric Category	**Speed**	**Accuracy**	**Reliability**
Orchestration Step 1 (e.g., Resource Allocation, Configuration and Setup)	VM Provisioning Latency	VM Placement Policy Compliance	VM Provisioning Reliability VM Dead-on-Arrival (DOA) Ratio
VirtualMachine operation	VM Stall (event duration and frequency) VM Scheduling Latency	VM Clock Error	VM Premature Release Ratio
Virtual Network Establishment	VN Provisioning Latency	VN Diversity Compliance	VN Provisioning Reliability
Virtual Network operation	Packet Delay Packet Delay Variation (Jitter) Delivered Throughput	Packet Loss Ratio	Network Outage
Orchestration Step 2 (e.g., Resource Release)			Failed VM Release Ratio
Technology Component asa- Service -	TcaaS Service Latency		TcaaS Reliability (e.g.,defective transaction ratio) TcaaS Outage

Table 6-2 Summary of ETSI NFV service quality metrics [40]

6.3.2.2. TMForum

Currently the TMForum has not provided any study of the possible SLA relationships that can arise in NFV ecosystem specifically. However, in order to implement the T-NOVA SLA management system we can look at SLA management in cloud environment to be adapted to the business NFV scenarios identified in T-NOVA.

TMForum released in October 2014 a new version of its technical document: Enabling End-to-end Cloud SLA Management [41] which refers to concepts and considerations in multi-provider cloud environment that in T-NOVA may be applied to the study of NFV SLA management, for instance:

- Cloud metrics, fall into two major categories: business metrics (often defined within the SLA), and Technical metrics (monitoring metrics) that allow the business SLA to be met. This can be applied also for SLA NFV metrics. For instance, “response time” may be specified in the SLA, meanwhile other technical measures such as “hops” and “bandwidth” may be used to dynamically allocate resources, enabling “response time” SLAs to be met.
- Usage-based costing metrics are generally a sub-category of the business metrics and will be a major component of a Service Agreement they may or may not be part of Service Level Agreement. Some examples of usage-based

metrics are: number of users, instance minutes, storage resource capacity used bytes, CPU minutes and RAM in megabytes, etc. Cost metrics are established based on money currency per unit ("€/instance minute" for example). SLA is primarily dealing with service assurance, but usage metrics that contribute to bill calculation will not be in scope for SLA management.

TM Forum's SID (Information Framework) has a metrics interest group, which delivered in September 2013 a modeling framework for metrics, in SID release 13.5 (definition of metrics, as well as hierarchy/relationships between metrics).

Requirements	Recommendation
SLA Modeling Methodology: use a common notation that is easily understood at the business level to model the SLA attachment point	Use the notation developed in [42] as the standards for SLA roles and responsibility analysis. Use TM Forum eTOM process fragments documented in [TMF GB917] for E2E SLA process analysis and design. See examples in (intermediary role & processes) Use TM Forum information model (SID) and related entities documented in [42] for E2E SLA information model analysis and design.
Metric Model: There are various types of metrics/measurements that contribute to the overall calculation of the SLA, such as Business Metrics, Performance Metrics, and Storage Metrics etc.). A meta model is required that provides a consistent description of these metrics that likely to be developed by different organizations and SMEs.	A Metric ABE (Aggregated Business Entity) is being defined by the TM Forum Shared Information/Data Model team, this work is to define a standardized definition and entity relationships so that metrics developed by various SDO/consortia can be joined up for the end-to-end management purpose. The intend of the SID metric ABE is to support all related work in this area, such as the work done in NIST Cloud Computing Metric group and the CSMIC work.
Service Level Specification(SLS) Model: the schema for service level specification that contains all measurements that need to be monitored for a given service.	Recommend to use SID ServiceLevelSpecification [23] as the standardized model for SLS schema development Recommend to use SID to construct Service Level Specification (SLS)
APIs: APIs to facilitate the automation and interoperability of SLA lifecycle management: SLA negotiation, activation, configuration and re-negotiation etc.	Candidates: WS-agreement, WS-agreement negotiation TM Forum SMI: for data collection SLA APIs, Catalog management APIs (under development)

Table 6-3 Standards for E2E Cloud SLA Management [41]

6.3.3. Protocols Overview

In the state of the art, we identified two high-profile options to implement an SLA protocol:

- **WS-Agreement** [43] is a web services protocol for establishing an agreement between two parties, such as a service provider and a consumer, using an extensible XML language for specifying the nature of the agreement, and agreement templates to facilitate discovery of compatible agreement parties. The specification consists of three parts which may be used in a composable manner: a schema for specifying an agreement, a schema for specifying an agreement template, and a set of port types and operations for managing agreement life-cycle, including creation, expiration, and monitoring of agreement states.

- **WS-Policy** [44] is a specification that allows web services to use XML to advertise their policies (on security, quality of service, etc.) and for web service consumers to specify their policy requirements. WS-Policy represents a set of specifications that describe the capabilities and constraints of the security (and other business) policies on intermediaries and end points (for example, required security tokens, supported encryption algorithms, and privacy rules) and how to associate policies with services and end points.

6.4. T-NOVA SLA framework

In T-NOVA there is a hierarchical SLA ecosystem, since there are two different SLAs:

1. SLA agreed between Function Provider (FP) and Service Provider (FP): a VNF can be offered by a FP with different SLA levels, according to different prices that will be negotiated through the trading mechanisms implemented by the brokerage module;
2. SLA between the Service Provider and its customers. One Network Service (NS) could have also different SLA levels with different prices, being part of different offerings as it is represented in Table 6-4:

SLA per service
service1, SLA11, price11
service1, SLA12, price12
service2, SLA21, price21
service2, SLA22, price22

Table 6-4 SLA per service

The T-NOVA SLA lifecycle will be implemented in the following steps:

1. *SLA Template Specification:* the SP and FPs follow a clear step-by-step procedure describing how to write an SLA template to provide a correct service description.

2. *Publication and Discovery*:
    - The SP publishes the different SLA offers for the NSs through the business service catalog for the customer to browse/compare offers.
    - The FP publishes the different SLA offers as part of the metadata that will be stored in the T-NOVA Function Store for each NF when uploading a VNF packaged [45]. The SP will discover the different SLA options by means of the brokerage procedure.

3. *Negotiation*: agreement on SLA conditions between the customer and the SP and between the SP and the FPs.

- For the SLA between customer and SP this will be done by the customer selecting one of the predefined choices of specific offerings from the business service catalog.
- For the SLA between SP and customer this will take place as the result of the trading process of VNFs with specific SLA levels.

4. *Resource Selection*: depending on the chosen SLA for every service, the orchestrator will allocate the resources that need to be assigned to the service in order to meet that SLA [24].

5. *Monitoring and Evaluation of the SLA*: this step will take place by comparing all the terms of the agreed SLA with the metrics provided by the orchestrator monitoring system. (These results will be available to be shown through the dashboard when the SP or customer requires it).

6. *Accounting*: this will be done invoking the charging/billing system to inform about billable items as penalties or rewards based to the result of step 5.

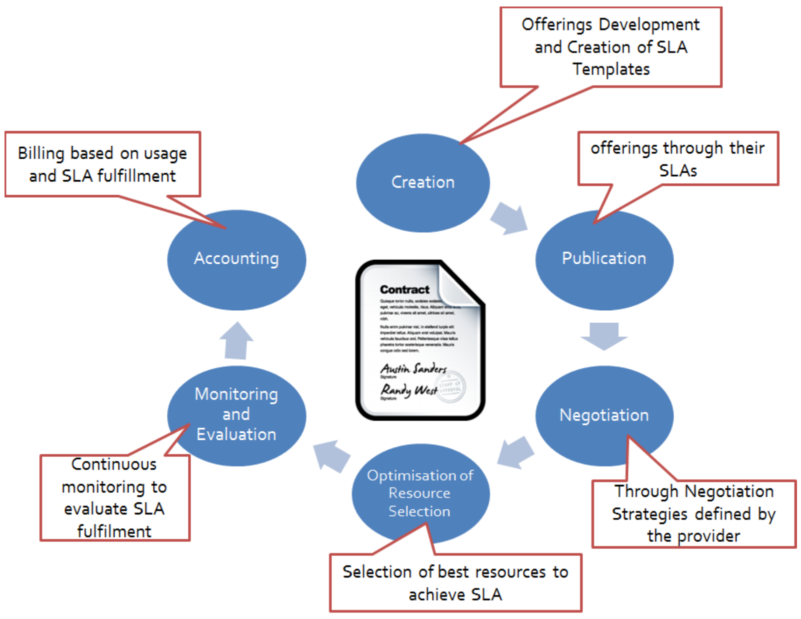

Offerings Development and Creation of SLA Templates
Billing based on usage and SLA fulfillment
Creation
offerings through their SLAs
Accounting
Publication
Contract
Monitoring and Evaluation
Negotiation
Optimisation of Resource Selection
Continuous monitoring to evaluate SLA fulfilment
Through Negotiation Strategies defined by the provider
Selection of best resources to achieve SLA

Figure 6-2 SLA lifecycle

6.4.2. SLA between SP and FP: VNF SLA

This will be the SLA agreed between the SP and the different FPs that sell the VNF as part of a network service.

We could have here two different approaches for the SLA associated to a VNF adquisition by the SP. On one hand each network function is a software product that the SP acquires to be deployed in his own infrastructure therefore we could think on one hand of a SLA associated to the software itself, to which we refer in 6.4.2.1. , and on the other hand having different levels for parameters associated for each VNF, such as VNF downtime, number-of-subscribers, etc, to which refer in 6.4.2.2..

6.4.2.1. SLA software

What the SP purchases to the FPs are software applications with accompanied metadata and images, this is, the deployment view of the software architecture.

In software development, specific SLAs can apply to application outsourcing contracts in line with standards in software quality, and recommendations provided by neutral organizations like CISQ, which has published numerous papers on the topic (such as Using Software Measurement in SLAs) [46].

6.4.2.2. SLA VNF monitoring parameters

The SLA agreed between SP and FP could include different levels of several parameters depending on the VNF itself, according to the monitoring parameters that will be part of the VNFD in accordance with the definition for the monitoring parameters that ETSI gives for the VNFD:

Monitoring parameters, which can be tracked for a VNF can be used for specifying different deployment flavours for the VNF in a VNFD, and/or to indicate ***different levels of VNF service availability.*** *These parameters can be an aggregation of the parameters at* ***VDU level*** *e.g.,* ***memory-consumption****,* ***CPU-utilisation****,* ***bandwidth-consumption*** *etc.* ***They can be VNF specific*** *as well such as* ***calls-persecond (cps), number-of-subscribers, no-of-rules, flows-per-second, VNF downtime,*** *etc. One or more of these parameters could be influential in determining the need to scale* [22].

For the VNFs that are going to be developed within T-NOVA to be used in the T-NOVA applications scenarios we are identifying the specific data that should be monitored to be relevant for the SLA.

Table 6-5 provides an initial list of monitored data generated by VNFs that could be used for SLA analysis.

VNF	Functionality Description	Monitored item name	Monitored item description	Metric unit	Relevant for SLA (y/n)
Traffic Classification	Inspect the packets using DPI methods in order to support (i) traffic classification for QoS, prioritisation etc (ii) application level monitoring and analytics	Packets in/out	Incoming and outgoing packets per second	packets per sec	Y
		Errors	Errors in the link	errors per sec	
		Drops	Dropped packets by the vNICs	dropped packets per sec	
		Application	Identified Application	String	
		QoS class	Diffserv code point (DSCP)	hex value	
		CPU Utilization	CPU Utlization	Percent	
		Memory Util	Memory Util	Percent	

Table 6-5 Parameters to be monitored for the Traffic Classifier (DPI) VNF relevant for the SLA

An example of SLA description for the the Traffice Classifier VNF is described in Table 6-6, including:

- SLA threshold quantity representing values that would be reasonable to ensure.
- SLA evaluation period that could be for example service subscription time, bill cycle, etc. SLA management module needs to know this information to stop monitoring when necessary and to know when the evaluation can be done.
- Actions to be taken when the SLA is unmet. Examples: if met, the FP could be recommended by the broker during X time, if not met, a discount can be applied on the SP bill, or more service could be granted during the next cycle (e.g more throughput, more service time, more users, etc.).

VNF	SLA thresholds	SLA evaluation period	Actions when SLA unmet
Traffic Classification	99% availability 90% identification accuracy Up to 1 sec identification latency	Bill Cycle	discount free usage of advance features

Table 6-6 SLA description for the Traffic Classifier (DPI) VNF

The definition of the SLA for the other three T-NOVA VNFs is in progress (Session Border Controller, Security Appliance, Home Gateway).

6.4.3. SLA between SP and customer

As it happens with cloud services, metrics that will be part of the SLA between the SP and T-NOVA customer can belong to two main categories: business metrics and technical metrics (monitoring metrics) that allow the business SLA to be met [42].

We can consider usage-based metrics as a sub-category of the business metrics, such as number of users, storage resource capacity, CPU utilization (%), RAM allocated (megabytes), simultaneously active sessions, etc. SLA is primarily dealing with service assurance, but usage metrics will contribute to the bill calculation being out of scope of SLA management.

Given that there are various types of metrics/measurements that can contribute to the overall calculation of the SLA a metamodel is required to provide a consistent description of these metrics that are likely to be developed. In T-NOVA the final SLA between SP and customer will be described and agreed in T-NOVA depending on the metrics that the monitoring system in T-NOVA will measure, which will depend on:

- Orchestration operation
- Virtual machine operation
- Network operation
- And VNF level of quality which will be determined by the SLA agreed by the SP and FPs for all the VNFs that are part of the NS.

Therefore, in a general case, the SLA between SP and customer will be an aggregation of the SLAs agreed between the SP and FPs for the VNFs that compose the service.

A first collection of metrics to be monitored has been identified in T-NOVA in [47] at different domains: VN/VNF, compute node, storage and network.

Domain	Metric	Units
VM/VNF	CPU utilisation	%
VM/VNF	No. of VCPUs	#
VM/VNF	RAM allocated	MB
VM/VNF	RAM available	MB
VM/VNF	Disk read/write rate	MB/s
VM/VNF	Network Interface in/out bitrate	Mbps
VM/VNF	Network Interface in/out packet rate	pps
VM/VNF	No. of processes	#
Compute Node	CPU utilisation	%
Compute Node	RAM available	MB
Compute Node	Disk read/write rate	MB/s
Compute Node	Network i/f in/out rate	Mbps
Storage (Volume)	Read/write rate	MB/s
Storage (Volume)	Free space	GB
Network (virtual/physical switch)	Port in/out bit rate	Mbps
Network (virtual/physical switch)	Port in/out packet rate	pps
Network (virtual/physical switch)	Port in/out drops	#

Table 6-7 Metrics collected by the VIM monitoring manager [47]

The final inputs to evaluate if the network service SLA has been fulfilled or not, will be received by the T-NOVA SLA management module from the orchestrator **Service Monitoring Component** which is responsible for monitoring all the service-related metrics that will be specified inside the NSD. The service monitoring module will be responsible for the coordination of the different VNF monitoring components, which are the components responsible for receiving information from the specific VNF monitoring components (further details in [24]– section 2.3.1).

According to the Monitoring Parameters part of the NSD that ETSI has defined [22]:

The NS monitoring parameters represent those which can be tracked for this NS. These can be network service metrics that are tracked for the purpose of meeting the network service availability contributing to SLAs (e.g. NS downtime). These can also be used for specifying ***different deployment flavours for the Network Service*** *in Network Service Descriptor, and/or to indicate* ***different levels of network service availability****. Examples include specific parameters such as* ***calls-persecond (cps), number-of-subscribers, no-of-rules, flows-persecond****, etc. 1 or more of these parameters could be influential in determining the need to scale-out.*

Based on the collection of parameters that can be collected at different levels in the system, we will may have also as part of the SLA, SLA telco service paramenters such as: delay, jitter, packet loss, etc. that are related directly with Quality of Service that the final customer will perceived when using end-to-end services.

6.5. Architecture

According to the T-NOVA SLA framework explained above, the SLA management module has to provide the mechanisms to get an agreement presented and agreed, informing the involved parties (Customer, SP, and FPs) and storing the SLAs, it will later receive and will process all measurements related to the SLA from the monitoring system (in the orchestrator) and, checking if the SLAs have been fulfilled or not, will inform the accounting system for the pertinent billable items (penalties or rewardings).

The internal architecture of the SLA management module is depicted in Figure 6-3.

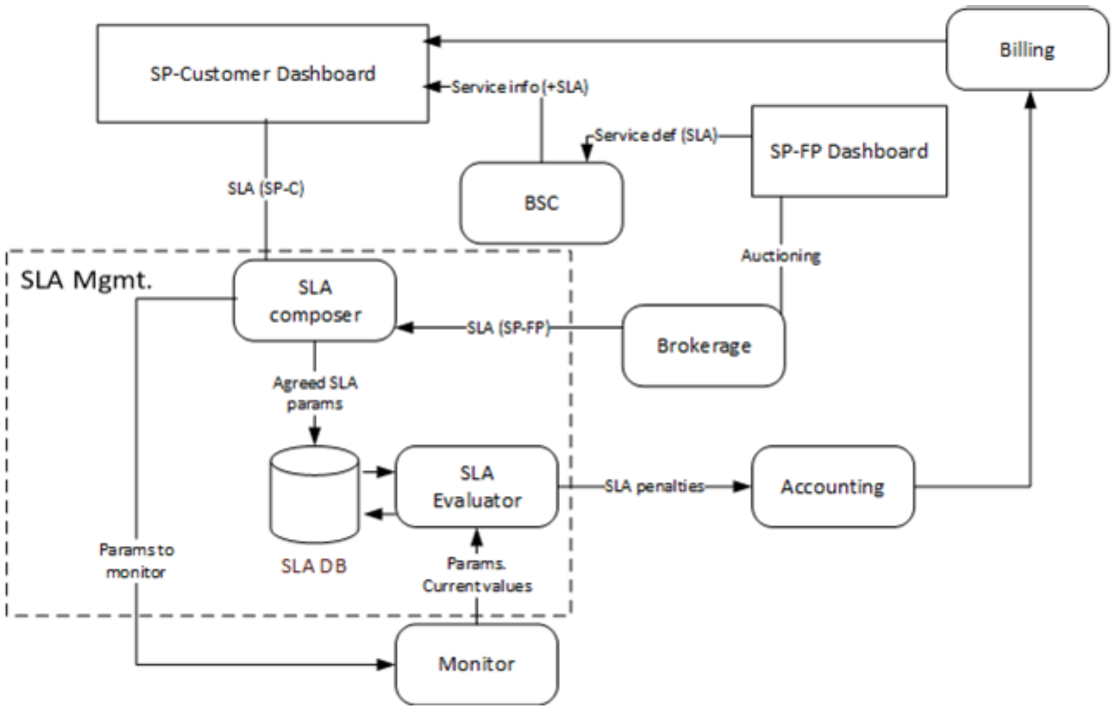


Figure 6-3 SLA management module architecture

The T-NOVA SLA management system will be composed of:

- SLA composer: it is the negotiation component in charge of creating the SLA template and the SLA agreement after the negotiation.
- SLA evaluator: it is the component in charge of checking if the SLA has been fulfilled or not and inform the accounting system about the pertinen billable items in terms of penalties of rewardings.
- SLA Database will store the information showed in Table 6-8:

Item	**SLA template specification**	**SLA contract** *Parties involved* *Parameters* *Penalties*	**SLA fulfiment**	**Billable items**
VNF ID	Input from FP dashboard	Input from the brokerage module as the output of the SLA negotiation	Input from the the monitoring system in the orchestrator	Output of the SLA evaluator (to be sent to the accounting module)
Service ID	Input from the SP dashboard	Input as the result of the service selection performed by the customer	Input from the the monitoring system in the orchestrator	Output of the SLA evaluator (to be sent to the accounting module)

Table 6-8 SLA management module information

6.5.1. API definition

Considering the interfaces of the T-NOVA SLA management module in Figure 6-4, the operations that its API will support are the following:

- Operation invoked by the Dashboard (T-Da-SL):
  - new_SLA (POST)
    - Publishes a new SLA agreement for a to-be-deployed service.
  - View_SLA (GET)
    - Obtains the content of an SLA agreement for visualizing purposes.
  - Change_SLA (PUT)
    - Modifies an SLA agreement.

- Operation invoked by Brokerage (T-Br-SL)
  - Set_SLA (POST)
    - Brokerage informs about the SLA agreed by FP-SP.
  - Change_SLA (PUT)
    - Modifies an SLA by the Brokerage

On the other hand the SLA management module will call the API of the following modules:

- Interface to Monitoring system (T-Sl-Or): SLA module accesses the Monitoring System (orchestrator) API to post the metrics to be monitored and to get the current values of each metric.

- Interface to Accounting (T-Sl-Ac): SLA module accesses Accounting to introduce SLA violations for penalties to be applied.

6.6. Candidate Technologies Selection and Rationale

The T-NOVA SLA Management module will be built considering the results of Cloud4SOA project [35]. In terms of technology baseline, Cloud4SOA provides a RESTful implementation of the WSAgreement standard. It offers a dynamic SLA negotiation and enforcement framework for multi-cloud environment based on a unified monitoring interface and metrics. It leverages on WS-Agreement specification to offer three main functionalities that enable users negotiate and enforce SLA, as well as recover from SLA violations: Agreement Negotiation, Agreement Enforcement and Violation recovery.

It allows the automatic negotiations on behalf of Cloud providers, based on the semantic description of offerings and the QoS requirements specified by application developers.

WS-Agreement is a novel but well-accepted standard offering a protocol to be followed for the negotiation process and a common understanding (i.e. language) of the objects the negotiation is about (see section 6.3.3). If we compare WS-Agreement vs WS-Policy the former its more expressive and more SLA focus, providing:

- Customizable metrics
- Service templates (helps in setting up a first SLA)
- Negotiation strategies (can be based on business rules)
- Accounting strategies (can be based on business rules)
- Adaptation to the monitoring system for each case.

Therefore, in T-NOVA WS-agreement will be used to be adapted to NFV in T-NOVA.

6.7. Technology selection summary

This section explained the justification of the main technological decisions taken for the implementation of SLA in T-NOVA Marketplace, including alternative option available at this point in time, the requirements [2] that they satisfy and the trade-off.

Topic	**SLA protocol**
Choices	▪ WS-agreement: web services protocol for establishing agreement between two parties, such as between a service provider and consumer, using an extensible XML language for specifying the nature of the agreement, and agreement templates to facilitate discovery of compatible agreement parties.
Alternatives	▪ WS-Policy is a specification that allows web services to use XML to advertise their policies (on security, quality of service, etc.) and for web service consumers to specify their policy

	requirements.
Requirements Related	< SLA.7 > The SLA management module SHALL provide mechanisms to get an agreement presened and agreed between the parties.
Trade-off	WS-Policy maybe more complete in terms on features.
Decision	WS-Agreement over WS-Policy since its more expressive and focused in SLAs, providing: o Customizable metrics o Service templates (helps in setting up a first SLA) o Negotiation strategies (can be based on business rules) o Accounting strategies (can be based on business rules) o Adaptation to the monitoring system

6.8. Future work

In this section we have presented the research work done so far in relation to the SLA scenarios that we are considering in T-NOVA, as well as the architecture that the SLA management module will have to implement the required functionalities. Also the justification of the protocol chosen has been explained. Further inputs will be needed from the VNF developers to identify the typical monitoring parameters that can be considered for each T-NOVA scenario, to make sure that the SLA management module will manage all of them.

7. Billing and Accounting

This section summarizes the progress so far on the implementation of the T-NOVA billing ecosystem, studing the most suitable billing mechanisms for network services and VNFs in T-NOVA and to implement the whole billing system (fed by SLA, price, resources usage, etc.) which includes the T-NOVA accounting module that store all the information that will be needed in T-NOVA for billing purposes.

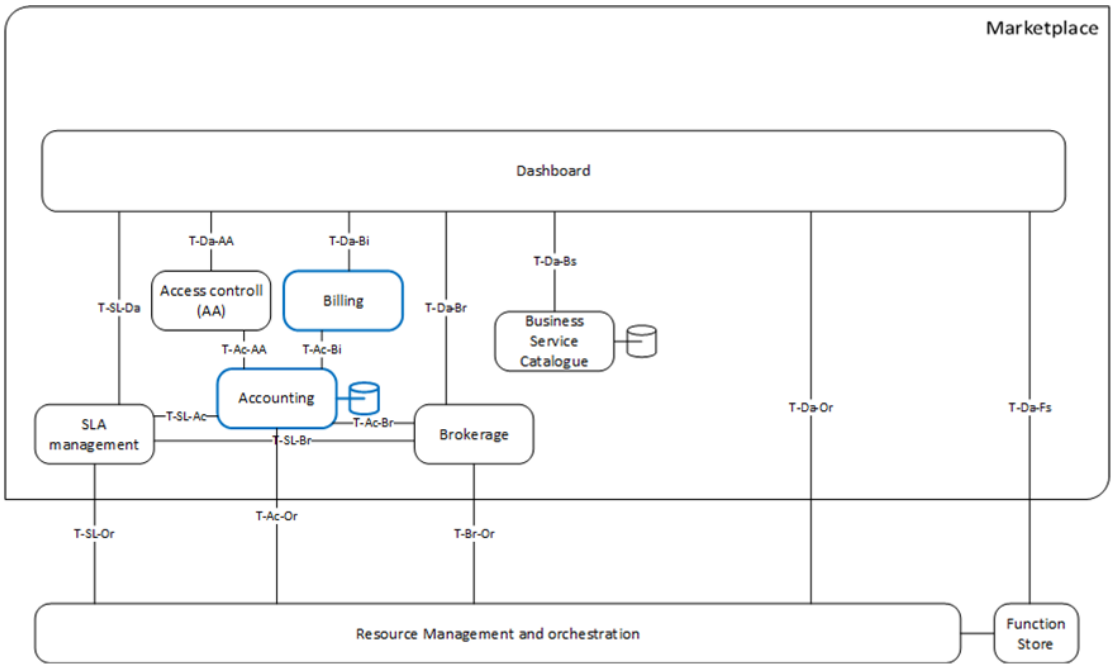


Figure 7-1 Billing + accounting in T-NOVA Marketplace

7.1. Requirements overview and consolidation

The requirements for the billing functionalities in T-NOVA come from the information that it will be needed and it will be store in the accounting module (Table 7-1) and the requirements for the billing module itself (Table 7-2).

Req. id	Domain	Requirement Name	Requirement Description
Ac.1	Management & Orchestration	Accounting notification - VNF starts	The accounting system SHALL know if a VNF starts correctly.
Ac.2	Market / commercial operability	Resources usage for billing	The accounting system SHALL store all the information about resources usage by each service for later billing purposes.
Ac.3	Market / commercial operability	Price information for billing	The accounting system SHALL store the information about prices agreed by each customer for later billing purposes.
Ac.4	Market / commercial operability	SLA billable items	The accounting system SHALL receive and store the information about SLA fulfilment for billing compensations or penalties.

Ac.5	Market / commercial operability	Bill cycle	The accounting module SHALL store the billing cycle information for each customer, and SP.

Table 7-1 Accounting module requirements

Req. id	Domain	Requirement Name	Requirement Description
Bil.1	Market / commercial operability	Price information for customer billing	The billing module SHALL receive the information about prices agreed by each customer for each service.
Bil.2	Market / commercial operability	Price information for SP billing	The billing module SHALL receive the information about prices agreed by each SP for each VNF.
Bil.3	Market / commercial operability	Bill issuing	The billing module SHALL issue bills when the customer's bill cycle finishes or service pay-as-you-go finishes and stores them within the customer profile.
Bil.4		Billing-accounting interface	The billing SHOULD receive all the information needed for billing from the accounting module.

Table 7-2 Billing module requirements

7.2. State of the Art analysis for Billing in T-NOVA

In the general case the T-NOVA marketplace requires the ability to allow the SP to combine VNFs from different FPs to be part of a same network service, in relation to which we could think in innovative **revenue sharing models** across partners.

Amazon [48], Google [49] and other internet players are already using innovative combination of business models, Internet-based platforms, and marketplaces, which have led to a successful monetization of offerings using marketplaces and business models based on proprietary domain-specific platforms operated by a single provider.

There are several current examples that require revenue distribution. The best known examples are Apple Application Store [50] and Google Play [49], which pay a percentage of the incomes from an application download to its developer. Another example relates to Telco API usage, as in Telefónica’s BlueVia [51] or Orange’s Partner [52], in which application developers receive a revenue share for the usage of Telco APIs by the final users.

Moreover, some of these players are offering their business capacities to third parties. For instance, Amazon DevPay [49] takes care of billing and accounting, on behalf of service providers, applications developed on top of Amazon Web Services [53]. Amazon DevPay facilitates the trading of digital goods to smaller players and, in exchange, Amazon retains a share of the revenues generated by selling these applications. This is also the case of T-NOVA, in which the platform operator does the invoicing for all the stakeholders involved (Customer, SP, NFP).

Another solution is named operator Billing, which allows customers to pay application downloads using their phone bill. For instance, Telefonica is offering this capability, by means of its BlueVia platform, which is already integrated with widely used marketplaces like Google Play and Windows Phone Marketplace [54].

Moreover, while a store like Apple, Google, and Amazon is owned by a store owner who has full control over the specific (limited) service portfolio, a marketplace is a platform for many stores to place their offerings to a broader audience and consumers to search and compare services and find the store, where to buy. In the case of T-NOVA marketplace there is only one store for standalone VNFs, and one store for VNF services, belonging to one owner, the T-NOVA operator, but in any case the customers will be able to search and compare services within that store.

In the app stores and marketplaces models environment **pay-per-license model** can be applied. The ´right to use´ model ensures that the ownership and control of software usage remains in hands of the NFP. This right to use is granted through a license agreement. This license agreement, with the help of a license enforcement mechanism, should ensure the protection of intellectual property and result in license compliance by tracking and managing the licenses in use. As of license pricing, users usually pay for an annual one-time license and a maintenance fee (although there is an increasing demand for the ´pay per use´ option for overloads). The most widely used pricing structures are per-seat (system, server), per-CPU and per-concurrent-user models. Conventional per-CPU, per-seat and per-job license management models and pricing structures are problematic and quite expensive for the use of commercial applications on top of cloud (which is the case of T-NOVA), web services or similar technologies, so custom-contract based models would be more suitable.

In an environment in which several cloud providers can provide the infrastructure to run software application, the Optimis project [20] proposes that:

- Software Licences are not bound to any hardware component or physical instance: the license is a mobile object that can be validated independently of the Cloud and physical machine in which the application is being executed.
    - In T-NOVA, this applies to the NFP charging the SP for using his function inside an offering.
- Software licenses are elastic objects: As analogy to Cloud’s IT infrastructure elasticity, in Cloud environments software licences would be able to being managed elastically, on-demand and with a pay-per-use approach.
    - In T-NOVA, if the SP needs to scale in (or out) the use of cloud resources on which the NF runs, there could be a price adjustment also between the NFP and the SP. In the simplest scenario, there would be no change in the price seen by the SP due to this (once the licence paid, the SP can use the NF “a volonté”).
- Software licences are charged when used: For any kind of virtualised environments, licenses would not have to be charged while the VM is not running, so that it is not providing service.
    - In T-NOVA, since the SP is also the owner of the infrastructure, this does not apply.
- Software Licences not bound to any specific virtualization technology: Currently there are License Management systems that are bound to the execution of a determined virtualisation technology. In Cloud environments, with the aim of getting full VM portability this restriction has to be eliminated.
    - In T-NOVA, since the SP is also the owner of the infrastructure, this does not apply.

7.3. T-NOVA Billing framework

As mentioned previously (section 1.2) the most generic T-NOVA business scenario comprises two different billing scenarios corresponding to the two different commercial relationships that we have in the T-NOVA ecosystem:

- The customer acquiring Network Services (NSs) from the Service Provider (SP), so the customer will be billed from the SP side.
- The SP acquiring VNFs from the Function Providers (FPs), so the SP will be billed by the different FPs.

7.3.1. Billing for VNFs

Based on the state of art analysis in the previous section in T-NOVA we could consider two main options for billing the SP for standalone VNFs:

7.3.1.1. Shared revenue billing model

Both composite and atomic NFs must be accounted, rated and charged according to their business model, and each of the players (SP, NFP) must receive their corresponding share. When a customer purchases an offering from a SP, he pays for it. This charge must then be distributed and split among different actors involved (even the NF store or marketplace owner could retain part of the revenue and FPs are paid out the corresponding revenue share). The T-NOVA marketplace is thus similar in some ways to the FI-WARE [16] one: it must allow sharing revenues not only between the platform and the service provider, but also among any provider in the value chain.

As far as price sharing is concerned, there could be different possibilities in T-NOVA:

- The SP pays a periodic fee to the T-NOVA operator for accessing the NF Store.
- The FPs pay a periodic fee to the T-NOVA operator for uploading their NFs to the NF Store.
- Then, each time a transaction occurs (i.e. the SP includes a NF in one of its offerings) and a customer purchases it, there is a revenue to be shared among the SP and the NFP. This can be based on a business model in an App Store paradigm-like fashion also contemplating a traditional pay-per-license business model.

Widely used marketplaces like Google Play rely on simple revenue sharing rules like a fixed percentage of the incomes generated by an application. This could be used in T-NOVA, but we can also explore the use of more complex revenue sharing models which also take into account additional business parameters, like the VNF type. An example could be to be able to share part of the monitoring information concerning the flow identification with the SP in order to have better management decisions in the future or for SP own usage. When agreed, the SP would be willing to charge the customer less for using this particular VNF.

7.3.1.2. License billing model

In T-NOVA, the FP could charge the SP once for the perpetual use of the software. Additionally, for support and updates, the FPs may ask for a yearly maintenance fee. Another case is when an annual fee is agreed on, that includes customer support and software updates (maintenance). The FP can also offer licenses for a short period of time (less than a year), including maintenance, or charge the SP on a ‘pay per use’ basis. As a combination of annual licenses and ‘pay per use’ model, the NFP can provide additional licenses on top of the regularly purchased annual licenses. The fee for these additional licenses is based on the utilization ratio during the license period.

The license may be paid by the SP directly or included in the price for the final customer. This depends on the scenario: the license might be reselled by the SP to the customer (who becomes the license user) or the SP might be the license user itself. If the SP is the infrastructure provider, as T-NOVA is proposing, then the scenario is less complicated than in the case of an external infrastructure provider. However, licensing models do not really fit the cloud business (pay-per-use is more suitable) and they even fit less when it is the customer who pays for the license (the user wants to benefit from cloud elasticity in a pay-per-use mode), so reselling the license is probably not the best approach for T-NOVA. Ideally, the SP holds the license and exploits it.

Based on the aboved and study in section 7.2 it has been decided that in T-NOVA the more suitable way would be the **FP providing a license to the SP,** e.g. number of features available, number of machines, which can be used to execute the application in parallel, number of users that might use the application in parallel, etc. The license server at the SP manages the license and in particular may create a license token for authorising the execution of the application. The license token contains all the information necessary to execute the application according to the user’s request (and with the constraints defined in the NFP license). This token is passed to the SP (together with the application if the service provider does not have a copy of it). In the last step the service provider deploys the service (the application) into the Virtual Infrastructure together with the license token and starts the service or informs the user that the service is ready for execution.

Licenses purchased from the FP -by the SP- should support pricing models such as pay per use. This would require a modification of the general licenses pricing models, which are now purchased at a fixed price independent from the effective usage and with annual maintenance fees. Also, as mentioned above, a SP would be able to buy licenses from an NF and host them on a license server inside the Cloud. The Cloud provider could then provide access to the license protected software on a pay per use basis (although this would mean reselling the license, which is not suitable in the T-NOVA scenario as described above).

The license server should be able to control the license-protected applications. For this it is required that:

- at runtime of the application a bi-directional network connection is available, which allows the application to connect to the license server to authorize the request to execute it (case 1),
- or, at runtime a local authorization is available, either based on a license server operated in the Cloud (SP) with local access to the required license or in form of a location independent authorization, e.g. a software token (case 2).

In T-NOVA, this means that the SP and NFP have both access to the license server for controlling the use of the software (NF).

Depending on the type of authorization:

- no changes are required if the license server provided at runtime can be accessed, either remote or local,
- in case 2, the applications need an API that allows the use of software tokens for authorisation.

The case of T-NOVA is simpler as far as the commercial relationship between the SP the and the NFP is concerned, since the NFP is only providing a function that can run on a virtual infrastructure, but not a SaaS product, since the FP is not a cloud provider scaling in resources, he just authorises the SP to use the NF inside an offering. On the contrary, the SP does offer SaaS (VNFaaS, PaaS, etc) to the final customer, instantiating more resources on the fly, scaling in and out when convenient, etc. This implies a SaaS pricing model, in which the final customer **pays per use of VNF and cloud infrastructure**.

The software license the SP purchases from the NFP may include, e.g.:

- Authorisation for N customers to use the NF in parallel.
- Recommendations concerning efficient deployments.
- Training for SP and/or end customers.
- Rights to new software releases during the term of agreement at no additional cost.

7.3.2. Billing for Network Services

Network services provide the backbone for all VNFs to communicate to one another and to the outside world, the network services not only comprise of inter-POP connectivity, but also intra-POP datacenter interconnection services. One could envision broad network services being provisioned to setup NF connections for inter and intra component connections at these levels:

- Hypervisor
- LAN (intra DC)
- LAN (inter DC – same provider)
- WAN (for communications to external domains)

Additionally, network service provisioning could involve legacy network equipment and also SDN capable equipment. So provisioning of network services could be differentiated along the lines of:

- Number of flow management commands used /
- Runtime flow management overhead

- QoS parameters to be enforced

And finally, the usage metering can be used to differentiate the data volumes crossing the hypervisor boundary, LAN, WAN, etc.

The T-NOVA billing system must be capable of supporting Network Services billing by enabling the provider to setup the accounting and billing systems utilizing any combination of service differentiating elements explored above. Any network flow associated to the customer's NF has to be metered and the usage data sent to the T-NOVA marketplace accounting module. Various billing models can be adopted including:

- Pay-as-you-go metered usage based
- Tiered rate peak/off-peak traffic
- Flat rate billing

7.4. Architecture

The T-NOVA billing system will be formed by an existing free-licensed billing module fed by the accounting module, which will be implemented by a database storing all the business relationships that will be needed for billing: subscriptions, prices agreed, services usage accounting, SLA billable items for penalties or rewardings and identification of customer and providers for each service or VNF.

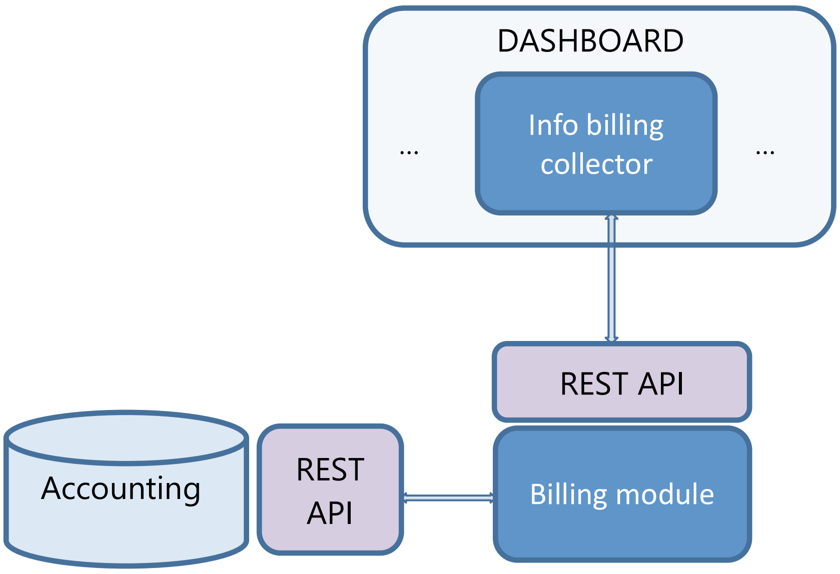


Figure 7-2 Billin + accounting architecture

The preliminary information that will be stored in the accounting module is described in Table 7-3.

User ID	User Role-permissions	Service ID/VNF ID	Service Usage	Pricing information	Bill cycle	SLA billable items
	Input from AA module	Identification in the business service catalog (service) or in the Function Store (VNF)	Input from monitoring system in the orchestrator	Input from the brokerage module about VNF price, and input from service selection by the customer in the dashboard	Agreed for each service/VNF in the trading process	Input from the SLA management module

Table 7-3 Accounting module information

7.4.1.1. Relation with other T-NOVA components

The SLA management module will provide the accounting module the billable items as penalties or rewards when the SLA has not been achieved.

The brokerage module will provide the accounting system with the appropriate information related to prices the VNFs as the result of the auctioning process.

The access control system will provide an API to access the "user profiles" database as some features are needed to handle the accounting.

The monitoring system in the orchestrator will use this interface to register in the accounting system the usage performed by the different services.

7.4.1.2. APIs definition

According to the interfaces that are explained in the previous section, the operations that the Accounting API will support are the following:

- Operations invoked by the the SLA Management (T-Sl-Ac)
  - SLA_violation (POST)
    - Billable items as result of violations of the SLA agreement.
- Operations invoked by the the billing module (T-Bil-Ac)
  - acc_info (POST)
    - Billing information to store it in the accounting DataBase.
- Operations invoked by the the brokerage (T-Br-Ac)
  - acc_price (POST)
    - VNF price information as a result of auctioning process from the brokerage.
- Operations invoked by the the orchestrator (T-Ac -Or)
  - service_run (POST) (PUT)
    - Status of a currently deployed service

On the other hand the accounting will call the API of the following modules:

- Interface to the access control (T-AA-Ac): accounting module accesses Access Control API to extract information to a certain user for validation.
- Interface to the billing (T-Bil-Ac): accounting module accesses Billing API to post accounting information to issue the bills.

7.5. Candidate Technology Selection and Rationale

In alignment with the marketplace architecture that we aim to implement, several opensource billing systems ant their feactures have been studied [see Annex 9.2] to find that one suitable based on T-NOVA billing requirements. Among them we have started to test *BoxBilling Free* [55] since it is offering more features, it has web access, and REST APIs for the integration of external elements, in compliance with the whole marketplace architecture that we aim to implement.

BoxBilling Free supports automated billing, invoicing and product provisioning. It's highly customizable, it will allow us to adapt it to the T-NOVA needs:

- Different payment methods for the products accepting one-time and recurring payments combined with the possibility to add a set up price. In case of T-NOVA we will have to customize this considering the two kinds of products that we are going to bill for: T-NOVA Network Services, and VNFs.
- *BoxBilling Free* supports introducing discounts using promotion codes: which in the case of T-NOVA will come from the SLA penalties.
- Payment reminders can be configured to be sent until the order will be terminated or late payment will be collected.
- A custom event hook script can be executed on order activation / suspension / reactivation / cancellation so it is posible to setup custom products easily and interrupt workflow as needed.
- It is posible to integrate any unsupported payment gateway by inserting own own HTML or Javascript code at the payment page.
- It has a RESTful API for T-NOVA to interact with the functions of the BoxBilling application.

Once development/integration work starts, in case there is any missing important feature for T-NOVA, we may consider extending the billing application selected.

7.6. Technology selection summary

This section summaries in the following tables the two main decisions taken considering the implementation of billing system in T-NOVA Marketplace, including alternative options available at this point in time, the requirements [2] that they satisfy, the trade-offs and the justification (e.g. the rationale behind the selection).

Some of these selections however are still under evaluation so they may change during the course of the activities as technical issues arise. Final details of the technologies ultimately used to implement T-NOVA billing framework will be provided in the next set of deliverables.

Topic	**Billing module**
Choices	○ **BoxBilling Free**
Alternatives	○ **Jbilling** Specific for telecom, fully customisable. Integration API available. ○ **Freeside** CRM and payments module included, self-management CRM. ○ **Fusioninvoice** Bill items based on free text (no pre-established catalog). ○ **Citrus DB** Web-based customer self-care. Reseller functionality.
Requirements Related	< Bil.3 > The billing module SHALL issue bills when the customer's bill cycle finishes or service pay-as-you-go finishes and stores them within the customer profile.
Trade-off	Limitation in number of users and products.
Decision	Offering more features, web access and REST API

Topic	**Accounting module**
Choices	▪ BD + REST API
Alternatives	▪ WStore (projects as XIFI and FI-WARE): wstore: It includes support for pricing (including pay-per-use modalities), accounting, charging, billing and revenue sharing.
Requirements Related	< Ac.2 > The accounting system SHALL store all the information about resources usage by each service for later billing purposes. < Ac.3 > The accounting system SHALL store the information about prices agreed by each customer for later billing purposes. < Ac.4 > The accounting system SHALL receive and store the information about SLA fulfilment for billing compensations or penalties. < Ac.5 > The accounting module SHALL store the billing cycle information for each customer, and SP

Trade-off	Building software from scratch.
Decision	Though following the same approach of billing that in WStore, we are not going to use it as it is, because it is an integrated solution specific for Future Internet applications, mainly in relation to the service repository, which in T-NOVA comes from the NFV orchestrator and the particularities related to NFV services. A customized software development will ease the integration with the rest of the T-NOVA modules.

7.7. Future work

In this section we have presented the study done so far in relation to the billing ecosystem, as well as the first technological decisions taken for its implementation. Further testing work will be done before implementation about the billing application chosen and its features to match them with T-NOVA, which in turn will have impact on the final design of the accounting database.

8. CONCLUSIONS

This document has introduced the work done in the first stages of the implementation of T-NOVA Marketplace and its components: business service catalog, brokerage module, user dashboard and SLA and billing management components. For each of these modules, the requirements gathered in previous work have been reviewed, amended if necessary, and consolidated. Then, a deep study of the state of the art has been elaborated focusing specifically to that applicable to T-NOVA approach to find the most suitable background to build on when implementing the T-NOVA marketplace: commercial solutions, standardization bodies and prvious research projects. Then the T-NOVA framework has been explained highliting the particularities for its implementation and justificating the first technical decisions made to achieve our objectives.

In order to provide the required modularity to the T-NOVA marketplace, it will be developed with a Software Oriented Architecture based on microservices that communicate each other by means of RESTful APIs. In this document, the definition of the operations that each REST API will have to support has been also included.

9. ANNEXES

9.1. Annex A – Dashboard front-end screenshots

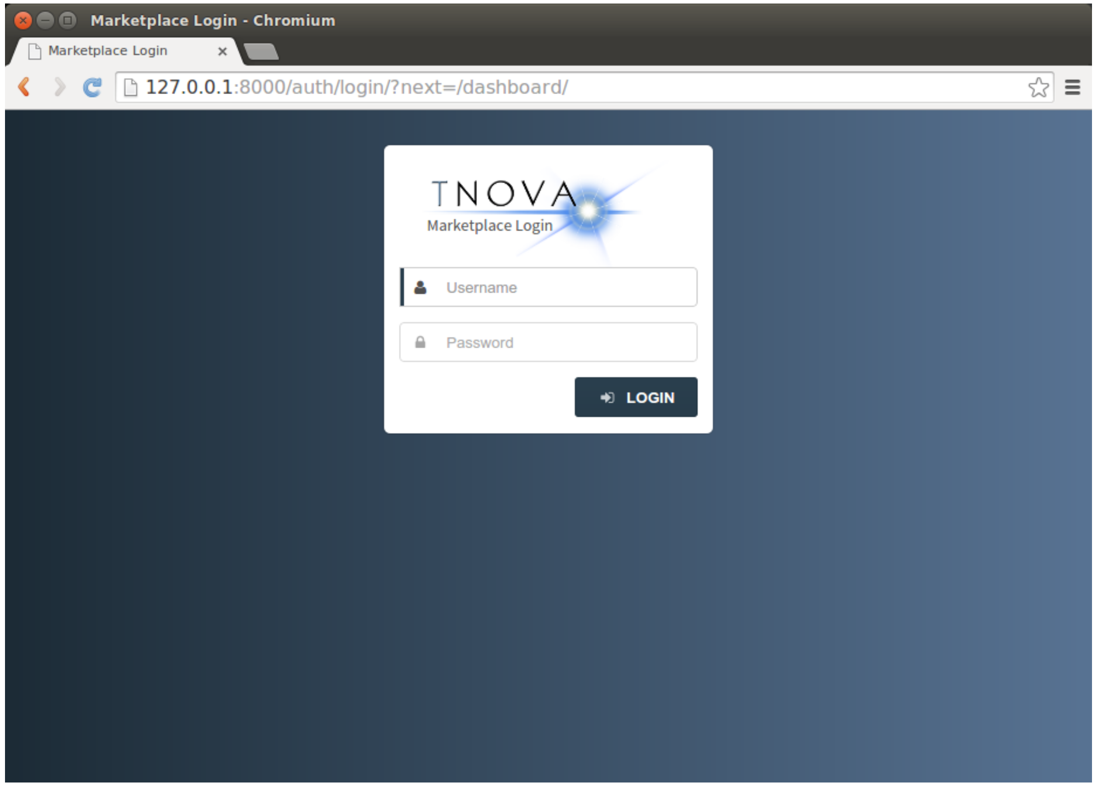
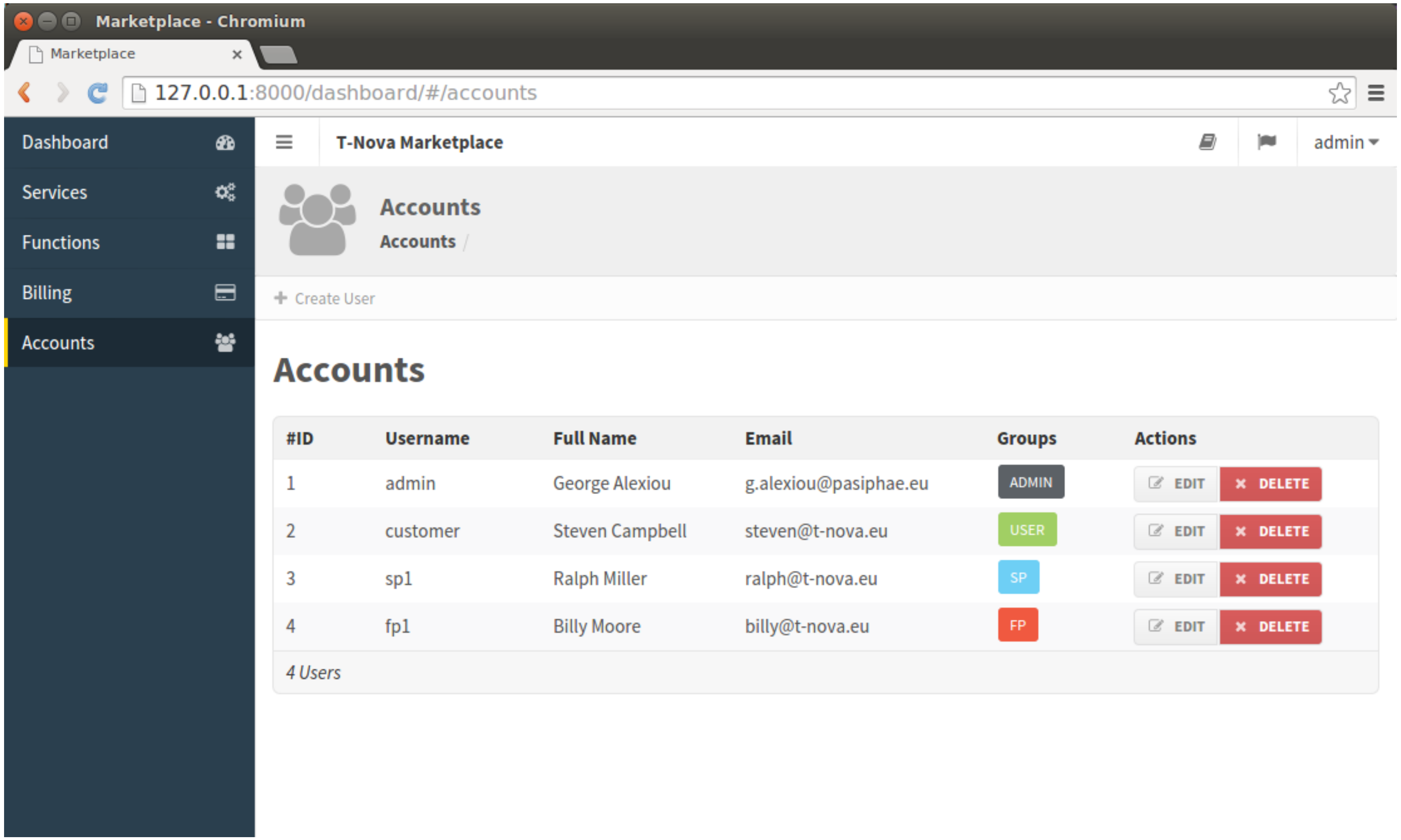

Marketplace Login - Chromium
Marketplace Login
127.0.0.1:8000/auth/login/?next=/dashboard/
TNOVA
Marketplace Login
Username
Password
LOGIN

Figure 9-1 Initial Dashboard Screen

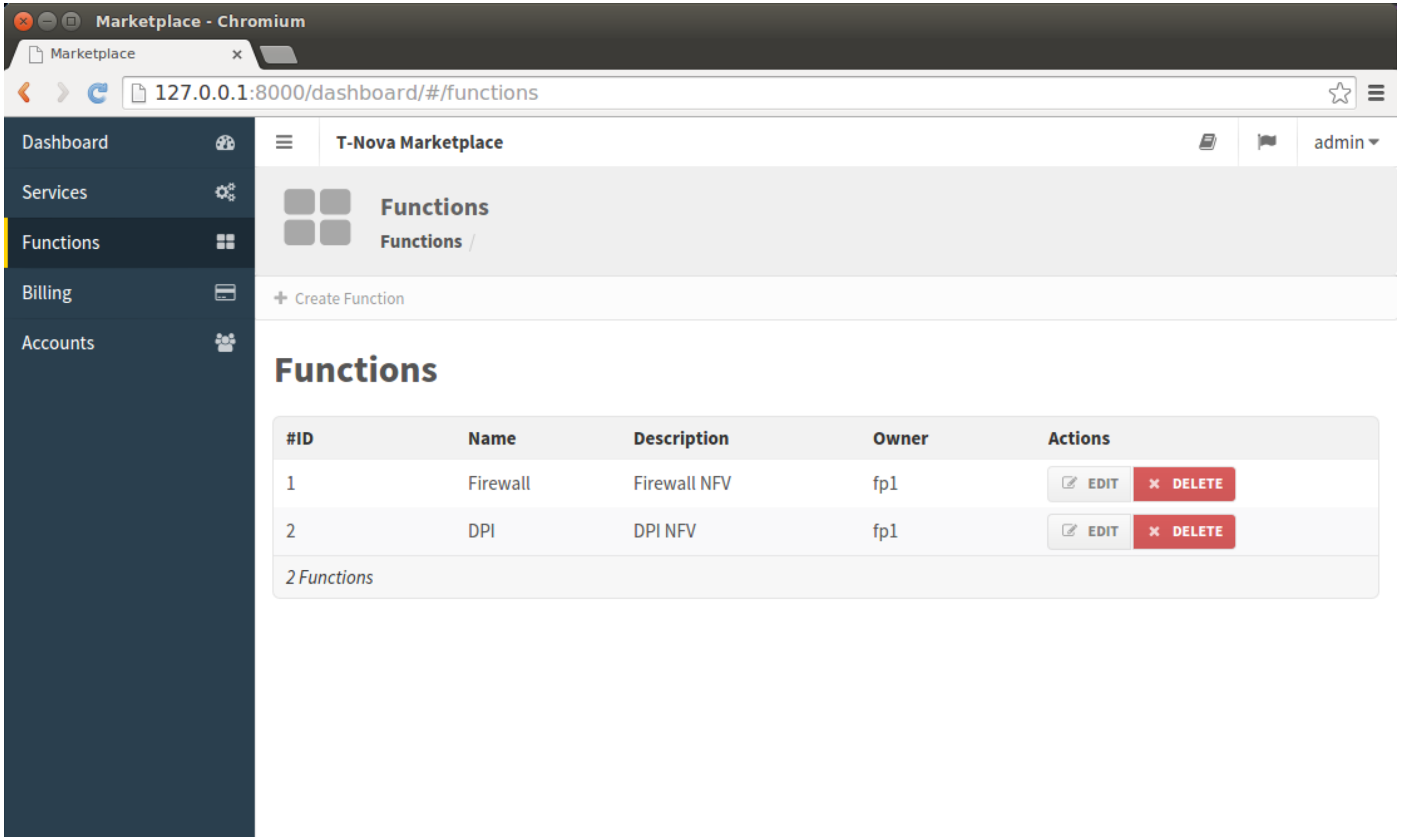


Accounts

#ID	Username	Full Name	Email	Groups	Actions
1	admin	George Alexiou	g.alexiou@pasiphae.eu	ADMIN	EDIT DELETE
2	customer	Steven Campbell	steven@t-nova.eu	USER	EDIT DELETE
3	sp1	Ralph Miller	ralph@t-nova.eu	SP	EDIT DELETE
4	fp1	Billy Moore	billy@t-nova.eu	FP	EDIT DELETE

4 Users

Figure 9-2 Available accounts with Roles

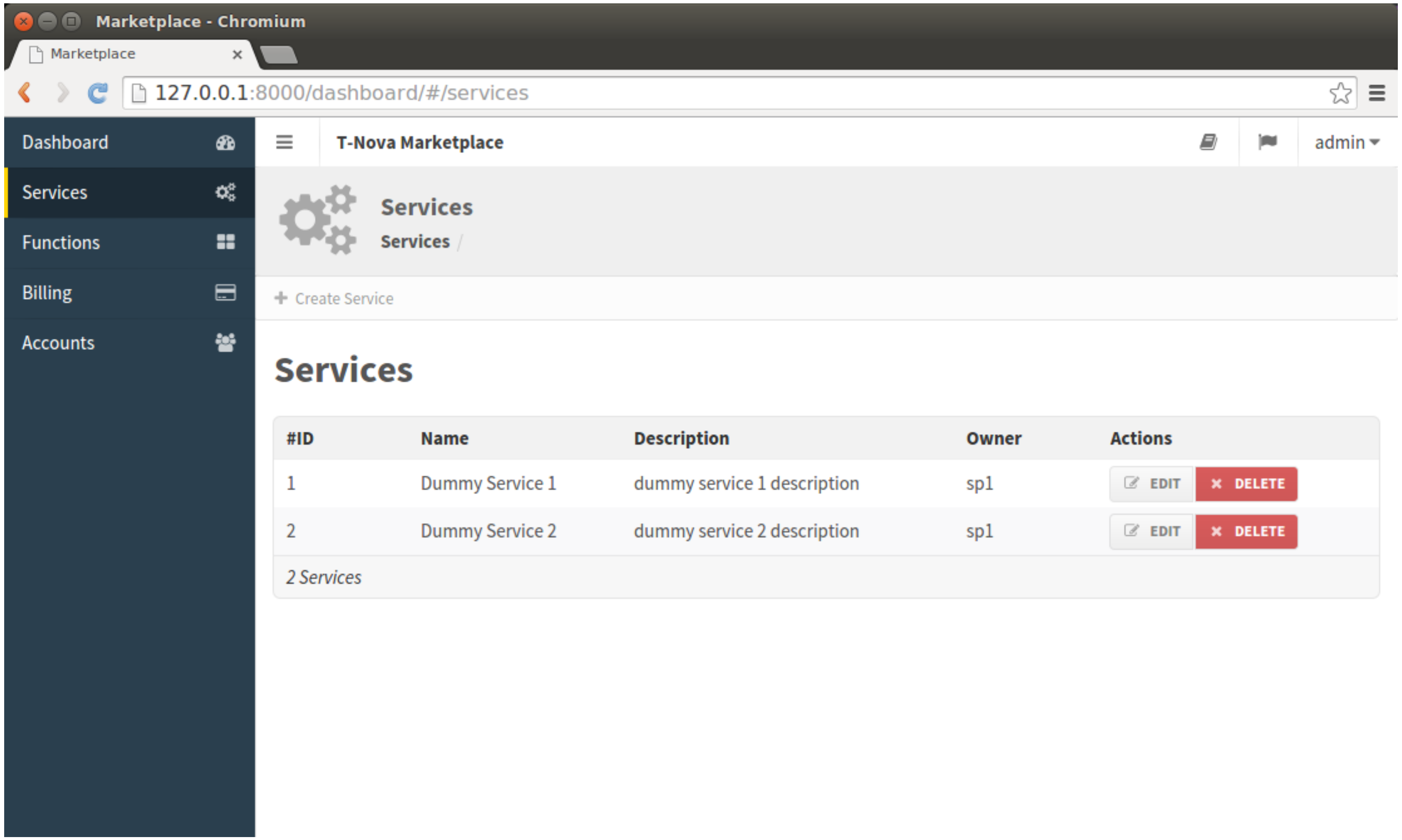


Functions

#ID	Name	Description	Owner	Actions
1	Firewall	Firewall NFV	fp1	EDIT DELETE
2	DPI	DPI NFV	fp1	EDIT DELETE

2 Functions

Figure 9-3 Available functions to edit or remove

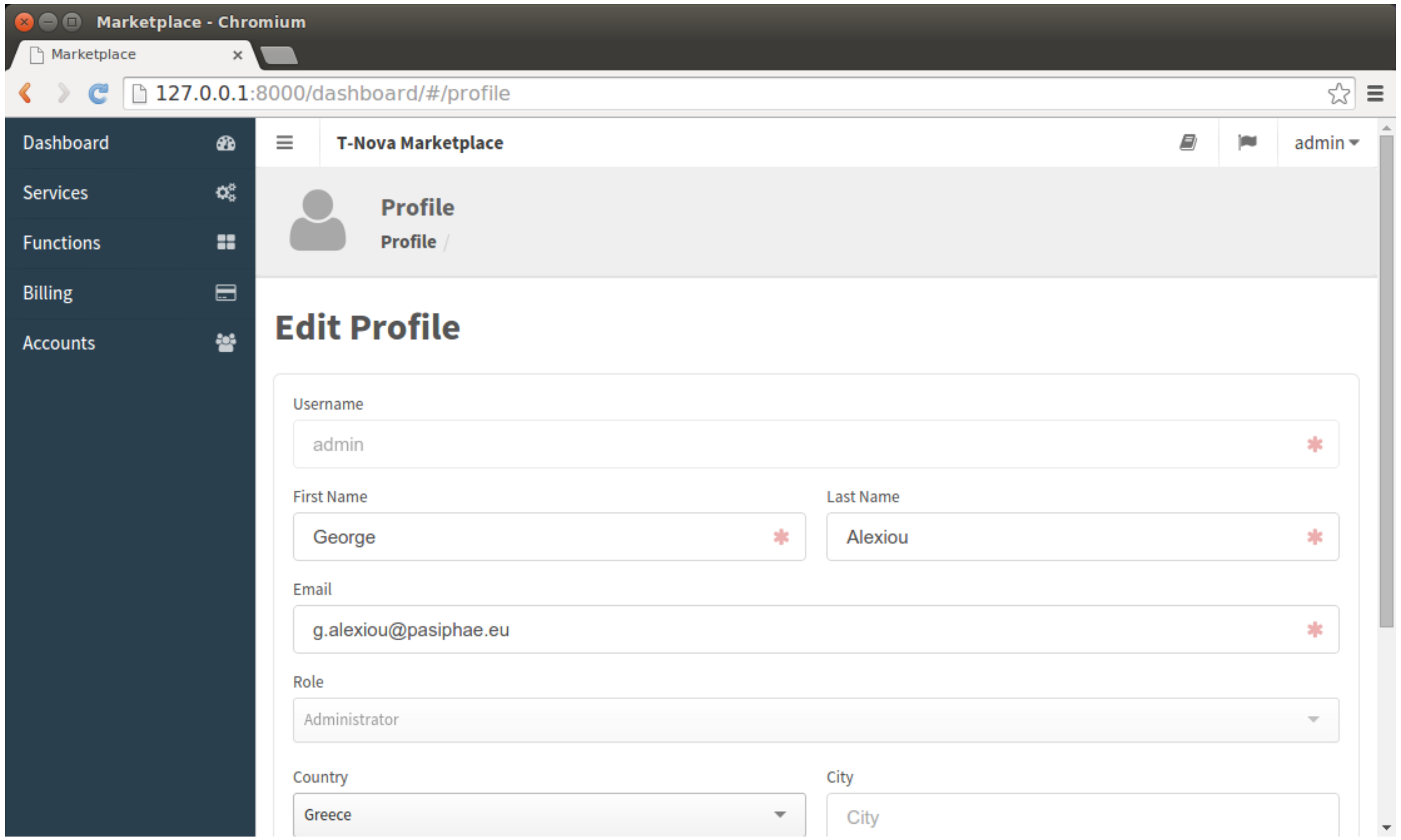


Services

#ID	Name	Description	Owner	Actions
1	Dummy Service 1	dummy service 1 description	sp1	EDIT DELETE
2	Dummy Service 2	dummy service 2 description	sp1	EDIT DELETE

2 Services

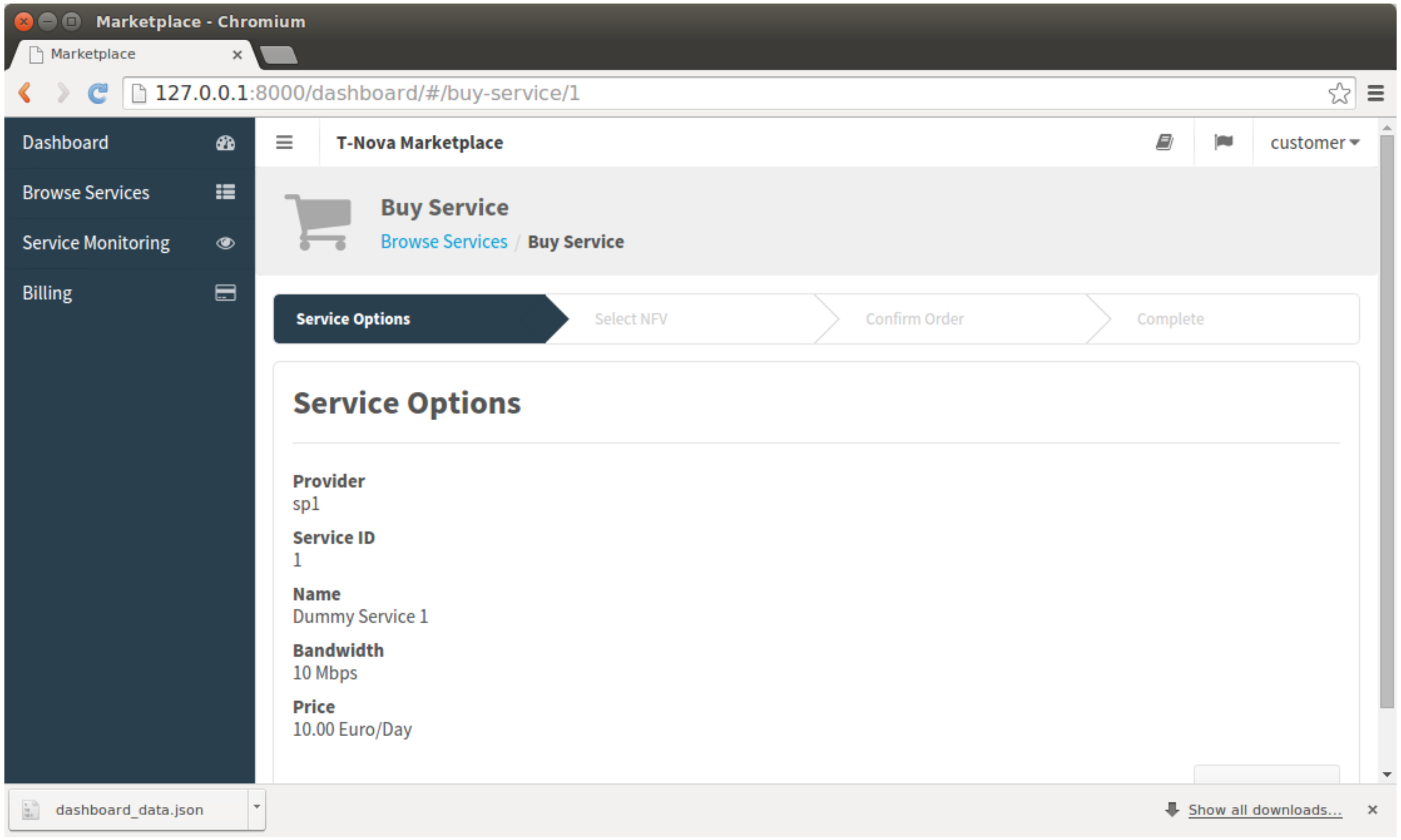
Figure 9-4 Available Services to edit or remove



Edit Profile

Username	admin
First Name	George
Last Name	Alexiou
Email	g.alexiou@pasiphae.eu
Role	Administrator
Country	Greece
City	City

Figure 9-5 Profile Management



Service Options

Provider
sp1

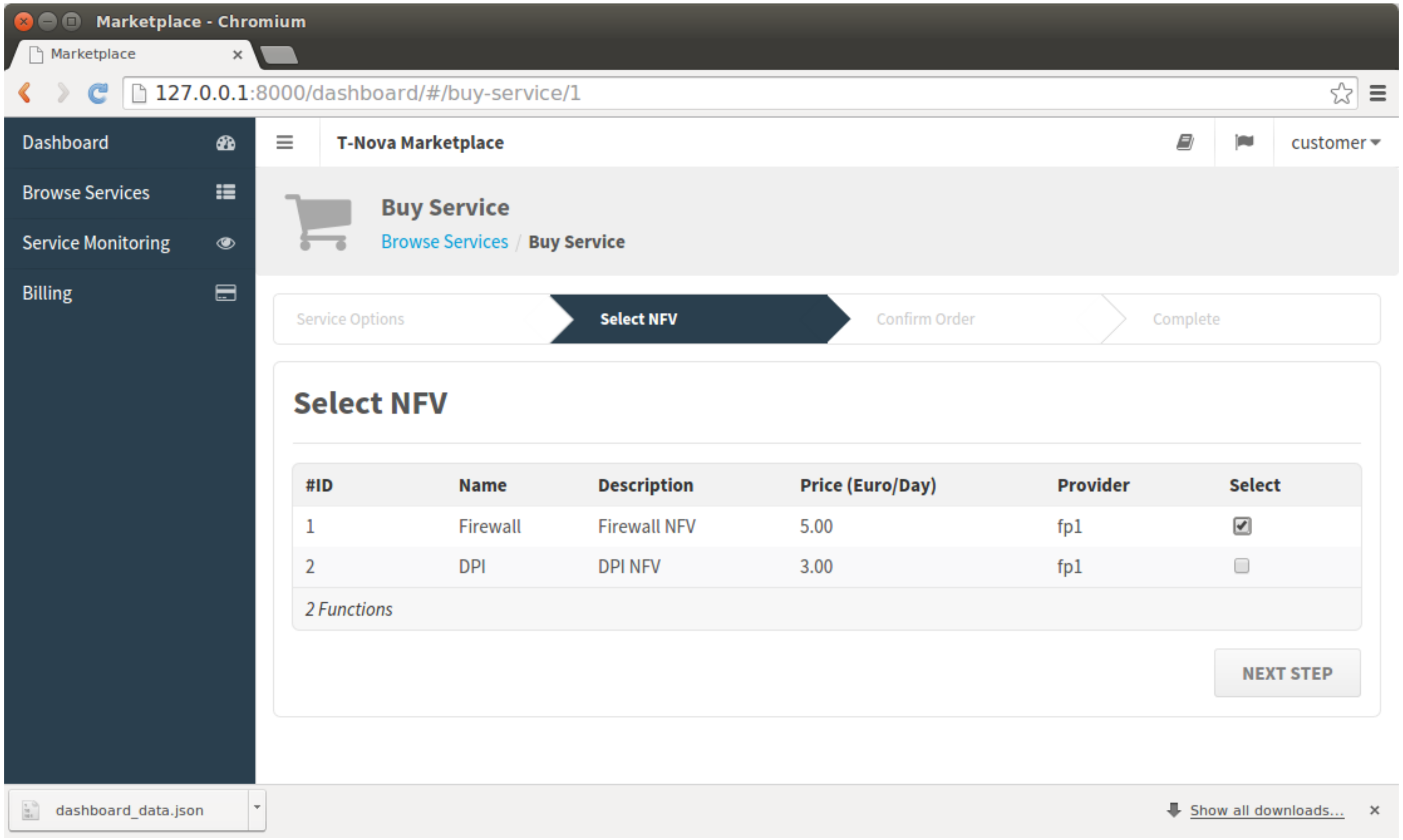
Service ID
1

Name
Dummy Service 1

Bandwidth
10 Mbps

Price
10.00 Euro/Day

Figure 9-6 Buy Service first Screen

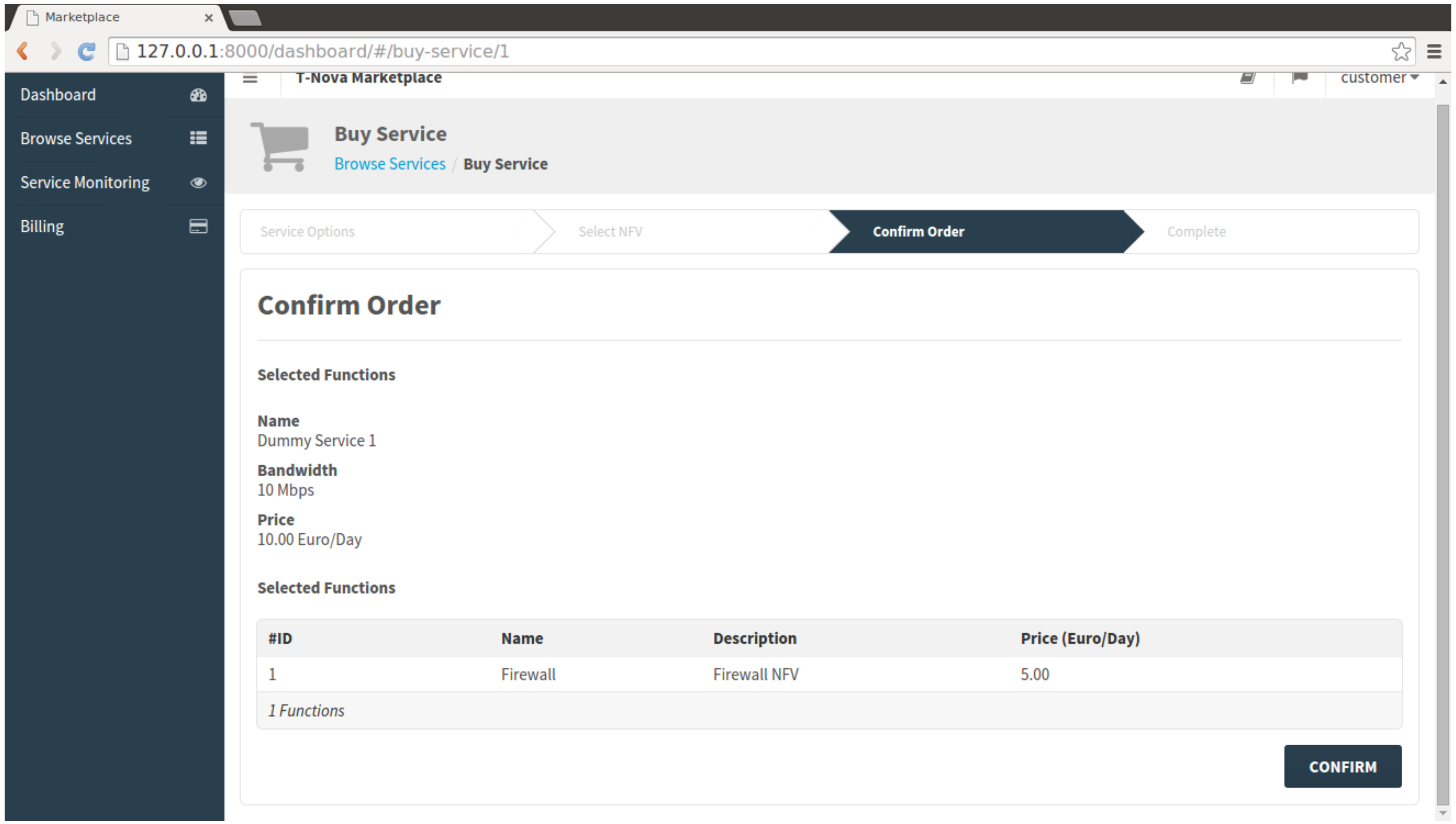


Select NFV

#ID	Name	Description	Price (Euro/Day)	Provider	Select
1	Firewall	Firewall NFV	5.00	fp1	✓
2	DPI	DPI NFV	3.00	fp1	☐

2 Functions

Figure 9-7 Select NFV for Service.



Confirm Order

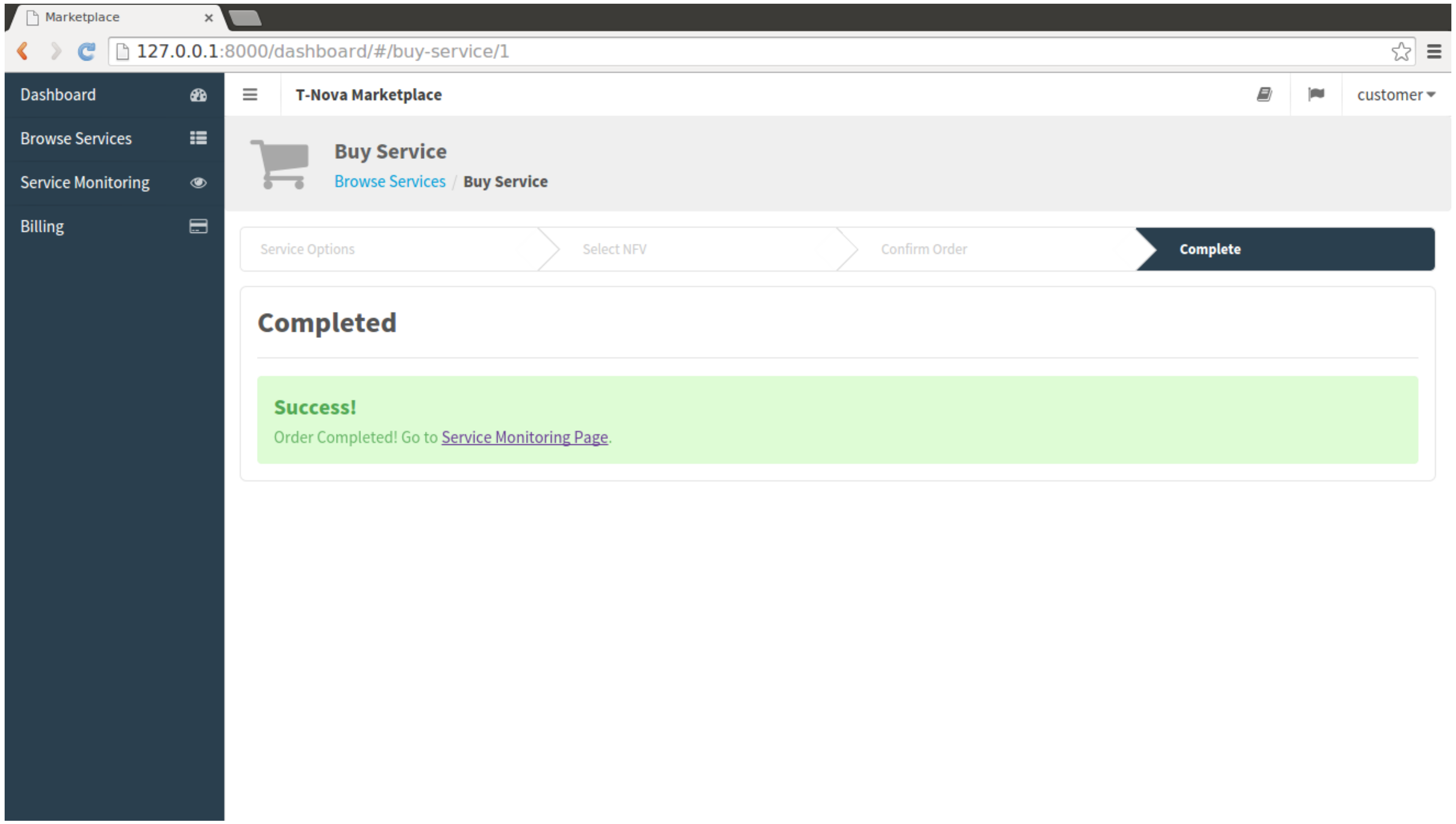
Selected Functions

Name
Dummy Service 1
Bandwidth
10 Mbps
Price
10.00 Euro/Day

Selected Functions

#ID	Name	Description	Price (Euro/Day)
1	Firewall	Firewall NFV	5.00
1 Functions			

Figure 9-8 Confirm Order for service

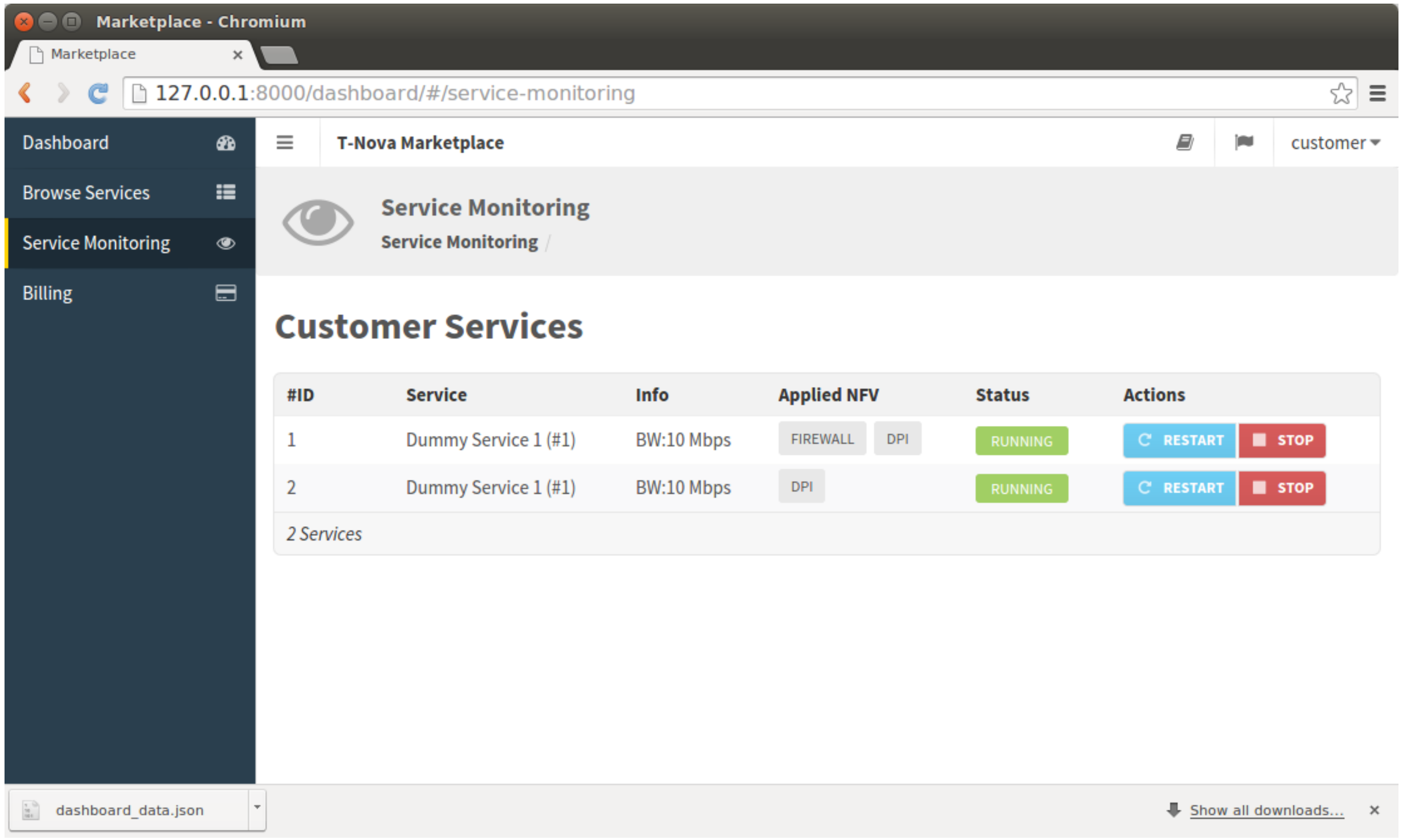


Completed

Success!

Order Completed! Go to Service Monitoring Page.

Figure 9-9 Order completed



Customer Services

#ID	Service	Info	Applied NFV	Status	Actions
1	Dummy Service 1 (#1)	BW:10 Mbps	FIREWALL DPI	RUNNING	RESTART STOP
2	Dummy Service 1 (#1)	BW:10 Mbps	DPI	RUNNING	RESTART STOP

2 Services

Figure 9-10 Service Monitoring

9.2. Annex B - Free-licensed billing applications comparison

	Solution type	Supported Platforms	Deployment Model	Number of Users	Billing and Invoicing	Payment management	Customer management	Price management	Supported Invoice Formats	Supported Payment Methods	Integration	Tech support
Bamboo Invoice	Stand-Alone	-	Online / Software as a Service	-	*Batch Invoice Creation *Customizable Invoices	-	-	-	PDF	-	-	Online self serve
Cashboard	-	Online	Online / Software as a Service	-	*Customizable Invoices *Recurring Invoices	-	-	-	Online (Website)	*Credit Cards *Paypal	-	Blog, FAQ
Paymo Free	Stand-Alone	Windows, Mac, online	Online / Software as a Service	1	*Retainer Billing *Customizable Invoices *Recurring Invoices	*Automatic Payment Processing *Automatic Receipt Generation *Recurring Payments	Customer Profiles	Complex Pricing	*Online (Website) *PDF	*Credit Cards *Paypal	API Available, Paypal	On-site
SmartInvoice Free	Stand-Alone	-	Online / Software as a Service	1	*Customizable Invoices *Recurring Invoices *Reverse/ Void Invoices	*Automatic Payment Processing *Multiple Customers Per Invoice *Recurring Payments	*Automated Followups *Customer Profiles	-	Email	*Credit Cards *Paypal	*API Available *Quickbooks	FAQ
CheddarGetter Developer	-	Online	Online / Software as a Service	-	*Customizable Invoices *Recurring Invoices	*Recurring Payments	Customer History	-	Email	credit Cards	paypal	Forums

Chargify Developer	Module	-	-	100	*Customizable Invoices *Invoice Scheduling *Recurring Invoices	*Automatic Payment Processing *Automatic Receipt Generation *Recurring Payments	*Automated Followups *Customer History *Customer Profiles *Different Rates for Different Customers	*Item Tax Specifications *Itemize Products & Services *Package Pricing *Promotional Pricing	*Email *Online (Website)	-	API Available	*Blog *FAQ *Instructional Videos *Recorded demos
AgileBill	Stand-Alone	Online	Online / Software as a Service	-	*Customizable Invoices *Recurring Invoices	Over 40 payment processors for full order and invoice automation	Customer Accounts	*Customizable Billing *Flexible Rate Tables	*Email *Online (Website) *PDF	*ACH *Credit Card Gateways *Paypal	-	Forums
Amberdms Billing System	Stand-Alone	Online	Online / Software as a Service	-	-	-	Customer record keeping features including multiple contacts, custom notes and documents and attribute store	-	PDF	API for integration with payment gateways	SOAP API	Forums
Freeside	Stand-Alone	Online	Online / Software as a Service	-	-	*Customer self-care including invoice viewing *One-time payments *Recurring payments	*Ticketing Basics *Search capabilities *Customer portal and branding	*Flexible pricing *Rating plans	*Email *Fax *Printed *Online	*Credit card *Electronic check processing with all major processing gateways	-	Forums
CitrusDB	Stand-Alone	-	-	-	Customizable invoices	-	Customer History	-	Email	-	-	Forums

Jbilling Free	Stand-Alone	Windows, Mac, Linux	Online / Software as a Service	-	Automated Invoice Generation	*Automated Payment Processing *Partial and Advanced Payments	-	-	*Email *Online (Website) *PDF	*ACH *Credit Card	-	Forums
SimpleInvoices	Stand-Alone	Windows, Mac, Linux, online	Online / Software as a Service	-	*Total invoices *Itemised invoices *Invoice templates *Customizable invoices *Recurring invoices	Payment tracking	-	-	*Email *Online (Website) *PDF	*Paypal *Eway Merchant Hosted		
BoxBilling Free	Stand-Alone	Linux	Online / Software as a Service	Unlimited	*Custom invoices *Every invoice can have custom taxation rule *Track invoice refunds *Allow your clients pay in their currency and track profit in your currency	*Order can be automatically suspended is no payment was received for X days and unsuspended as soon the late payment will be collected *Payment reminders	Integrated helpdesk allows you to communicate with clients easily	-	Email	All Payment gateways	REST API	*Knowledge base *Community support
Billy's Billing	Stand-Alone		Online / Software as a Service	-	Customizable invoices	Payment reminders	Customer History		*Email *Online (Website) *PDF		REST API	Email

Table 9-1 Existing free-licensed billing systems

10. References

[1] Deliverable D2.2 – Overall System Architecture and Interfaces. T- NOVA project.

[2] Deliverable D2.1 – System Use Cases and Requirements. T- NOVA project.

[3] D2.41 - Specification of Network Function Framework and T-NOVA Marketplace. T-NOVA project.

[4] Service Oriented Architecture: http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html.

[5] RESTFul APIs: http://www.django-rest-framework.org/.

[6] TM Forum, "TM Forum WebSite," http://www.tmforum.org.

[7] Network Description Language, F. Dijkstra and J. van der Ham, ttp://sne.science.uva.nl/ndl/.

[8] Koslovski, G., Primet, P.-B., & Charão, A. (2009). VXDL: Virtual Resources and Interconnection Networks Description Language. In P. Vicat-Blanc Primet, T. Kudoh, & J. Mambretti (Eds.), Networks for Grid Applications SE - 15 (Vol. 2, pp. 138–154). Springe.

[9] perfSONAR, http://www.perfsonar.net.

[10] cNIS Database Documentation, https://forge.geant.net/forge/display/CNIS31DOC/Database+schema.

[11] F. D. a. J. v. d. H. t. Network Description Language.

[12] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[13] Unified Service Description Language (USDL) http://www.w3.org/2005/Incubator/usdl/XGR-usdl-20111027/.

[14] Z39.50 protocol: http://www.niso.org/standards/resources/Z39.50_Resources.

[15] Open Archives Initiative Protocol for Metadata Harvesting http://www.openarchives.org/OAI/openarchivesprotocol.html.

[16] FI-WARE project: http://cordis.europa.eu/fp7/ict/netinnovation/deliverables/fi-ware/deliverables-fi-ware_en.html.

[17] Project XIFI (FI-PPP), "Official Web Site," [Online]. Available: [Accessed 2014].https://www.fi-xifi.eu/home.html.

[18] WStore http://catalogue.fi-ware.org/enablers/store-wstore.

[19] Resources and Services Virtualization without Barriers (RESERVOIR) Project. http://cordis.europa.eu/project/rcn/85304_en.html.

[20] OPTIMIS Project http://cordis.europa.eu/fp7/ict/ssai/docs/call5-optimis.pdf.

[21] ETICS project - https://www.ict-etics.eu/.

[22] ETSI NFV ISG, “NFV-MAN 001 NFV Management and Orchestration,” July 2014. [Online]. Available: http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/NFV-MAN001v061 %20management%20and%20orchestration.pdf.

[23] TMForum Information Framework. http://www.tmforum.org/InformationFramework/1684/Home.html.

[24] D3.01 - Interim Report on Orchestration Platform Implemenation. T-NOVA project.

[25] Survey of Service Description Languages and TheirIssues in Cloud Computing. Le Sun, Hai Dong, Jamshaid Ashraf. School of Information Systems, Curtin Business School, Curtin University of Technology, Perth, WA, Australia.

[26] BonFire project. http://doc.bonfire-project.eu/R4.0.5/reference/bonfire-architecture.html?highlight=enactor.

[27] ICT-COGEU project. http://www.ict-cogeu.eu/.

[28] CompatibleOne project. http://www.sucreproject.eu/content/compatibleone-open-cloud-broker.

[29] MyProjectBroker.com. http://www.myprojectbroker.com/what-is-a-project-broker/.

[30] ICT-Broker@cloud. http://www.broker-cloud.eu/.

[31] OpenNaaS. www.opennaas.org.

[32] AngularJS. https://angularjs.org.

[33] Django. https://www.djangoproject.com.

[34] JSON Web Token (JWT) draft-ietf-oauth-json-web-token-32M. Jones, J. Bradley, N. Sakimura.

[35] CLOUD4SOA project: http://cordis.europa.eu/fp7/ict/ssai/docs/call5-cloud4soa.pdf.

[36] FED4FIRE project: http://www.fed4fire.eu/.

[37] Vodafone: http://www.vodafone.co.uk/about-this-site/terms-and-conditions/mobile-broadband-via-the-phone/index.htm.

[38] Orange: http://clientes.orange.es/soporte_y_ayuda/pdf/1051479_CCGG_PD.pdf.

[39] ETSI GS NFV 004 Network Function Virtualization. Requirements.

[40] ETSI GS INF 010Network Function Virtualization. Service Quality Metrics.

[41] TMFORUM Enabling End-to-End Cloud SLA Management. October 2014.

[42] TMFORUM SLA Management Handbook.

[43] WebService-Agreement specification. http://wsag4j.sourceforge.net/site/wsag/overview.html.

[44] WebService-Policy. http://www.w3.org/TR/ws-policy/.

[45] D5.01 - Interim Report on Network Functions and associated framework. T-NOVA project.

[46] CISQ. Consortium for IT Software Quality. http://www.it-cisq.org.

[47] D4.01 - Interim Report on Infrastructure Virtualisation and Management. T-NOVA project.

[48] Amazon DevPay: http://aws.amazon.com/devpay/.

[49] Google Play. https://play.google.com/store.

[50] Apple App Store (requires iTunes): http://www.itunes.com/appstore/.

[51] BlueVia website: http://www.bluevia.com/.

[52] Orange Partner website: http://www.orangepartner.com.

[53] AWS Marketplace, https://aws.amazon.com/marketplace.

[54] W. P. http://www.windowsphone.com/.

[55] Boxbilling. http://www.boxbilling.com/.

11. GLOSSARY

Name	Description
Access Control Module	Component in the marketplace that administers security managing and enabling access authorization/control for the different T-NOVA stakeholders considering their roles and permissions.
Accounting Module	Compoment in the marketplace that stores all the information needed for later billing for each user: usage resources for the different services, SLAs evaluations, etc.
Billing Module	Compoment in the marketplace that produces the bills based on the information stored in the accounting module
Business Service Catalog	Catalog in the marketplace that store all the available offerings.
Brokerage Module	Component in the marketplace that enables trading of VNFs, facilitating the auctioning between Function Providers.
T-NOVA Customer (customer)	Stakeholder that aims to acquire T-NOVA Network Services.
Dashboard	Graphical User Interface (GUI) for the stakeholders to interact with the system. In T-NOVA has 3 different views: SP dashboard, FP dashboard and customer dashboard.
Function provider	Software developer that offer VNFs in the marketplace to be sold.
Function store (NF Store)	The T-NOVA repository holding the images and the metadata of all available VNFs/VNFCs
NFV Infrastucture (infrasctructure)	The totality of all hardware and software components which build up the environment in which VNFs are deployed
Marketplace	The set of all tools and modules which facilitate the interactions among the T-NOVA actors, including service request, offering and provision, trading, service status presentation and configuration, SLA management and billing
NS Catalog	The Orchestrator entity which provides a repository of all the descriptors related to available T-NOVA services
Offering	Each Network Service available in the marketplace together with a SLA level and price. It is created by the Service Provider and store in the Business Service Catalog to advertise the services to the customer.
Orchestrator	The highest-level infrastructure management entity which orchestrates network and IT management entities in order to

	compose and provision an end-to-end T-NOVA service.
Service Provider	Stakeholder that offer Network Services through the marketplace creating offerings in the business service catalog. To create the network services the SP acquires VNFs from the Function Providers. The VNF are deployed over the T-NOVA infrastructure.
SLA Management Module	Component in the marketplace that establishes and stores the SLAs among all the involved parties and checking if the SLAs have been fulfilled or not will inform the accounting system for the pertinent billable items.
Stakeholder	Each of the kind of actors that can use T-NOVA system: SP, FPs, customers.
T-NOVA Network Service ("service")	A network connectivity service enriched with in-network VNFs, as provided by the T-NOVA architecture.
T-NOVA Operator	The T-NOVA system administrator that owing the T-NOVA infrastructure controls the activity of all the T-NOVA users.
VNF catalog	The Orchestrator entity which provides a repository with the descriptors of all available VNF Packages.
VNF	A virtualised (pure software-based) version of a network function

12. List of Acronyms

Acronym	Explanation
AA	Authentication and Authorisation
AAA	Authentication, Authorisation, and Accounting
API	Application Programming Interface
BSC	Business Service Catalog
BSS	Business Support System
CRUD	Create Read Update and Delete
CPU	Central Processing Unit
DPI	Deep Packet Inspector
eTOM	Telecom Operations Map
GUI	Graphical User Interface
ETSI	European Telecommunication Standard Institute
EU	End User
FI	Future Internet
FP	Function Provider
HGW	Home GateWay
ISG	Industry Specification Group
IT	Information Technology
IVM	Infrastructure Virtualisation Layer
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MANO	Management and Orchestration
NFaaS	Network Functions-as-a-Service
NF	Network Function
NFC	Network Function Component
NFV	Network Functions Virtualisation
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtualization Orchestrator
NS	Network Service
NSD	Network Service Descriptor

OSS	Operational Support System
QoS	Quality of Service
RBCA	Role Based Access Control
SaaS	Software-as-a-Service
SBC	Session Border Controller
SDK	Software Development Kit
SDO	Standards Development Organisation
SID	Shared Information/Data model
SLA	Service Level Agreement
SP	Service Provider
UC	Use Case
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFaaS	Virtual Network Function as a Service
VNFD	Virtual Network Function Descriptor
VNFM	Virtual Network Function Manager
VNI	Virtual Network Interface
VNPaaS	Virtual Network Platform as a Service
WP	Work Package