TNOVA

NETWORK FUNCTIONS AS-A-SERVICE
OVER VIRTUALISED INFRASTRUCTURES

GRANT AGREEMENT NO.: 619520

Deliverable D6.1

Service Description Framework

Editor Aurora Ramos (ATOS)

Contributors Aurora Ramos, Javier Melián (ATOS), Thomas Pliakas (CLDST), Evangelos Markakis, George Alexiou (TEIC), Paolo Comi (ITALTEL)

Version 1.0

Date December 30th, 2015

Executive Summary

This document reports the results of the activities carried out in T-NOVA EU-FP7 Project "Functions as-a-Service over Virtualised Infrastructures" by Task 6.1 "Service Description Framework". Based on previous specification work in the project, the current document is delivered at the same time the T-NOVA Marketplace service description modules prototypes are finished (http://github.com/T-NOVA).

This report includes a summary of the main on-going related activities in the SOTA, providing a brief update from the analysis performed in the specification phase one year ago, describes the key points of T-NOVA service description framework at marketplace layer, and details their different modules architecture, documentation related to their implementation (UML diagrams) and their integration with the rest of T-NOVA components (including APIs definitions). Also the results of functional verifications have been included as well as the report on requirements fulfilment.

A unique Network Service Descriptor (NSD) has been created in T-NOVA to be shared as part of the same information model between the Marketplace and the Orchestrator. Due to the Orchestrator is expected to be finalized by end of March'2015 refinements on the Marketplace could be needed, for instance a new slightly different version of the T-NOVA NSD could be evolved. This could result in an updated version of the current deliverable.

T-NOVA Marketplace information model comes as a result of applying TMForum SID model to ETSI NFV information model, adding on top of the Network Service Descriptor (NSD) proposed by ETSI for orchestration, a related customer facing Network Service. Therefore, T-NOVA descriptors have been created as an extension of ETSI NSD and VNFD, adding the fields that allow business interaction among the stakeholders that interact in the T-NOVA Marketplace: SLA specification, pricing, etc.

Two main components form the T-NOVA Marketplace service description framework:

- Business Service Catalogue (BSC): this has been introduced following the TMForum recommendations for business agility. The Service Provider will create service offerings for T-NOVA to generate the NSD that will be stored in the BSC, and which will be later browsable by the customer to select.
- Service selection module (SS): this module has been designed to ease the adaptation of the generic service to the customer network and to send this customization to the Orchestrator to proceed with the service components instantiation.

All the components in the T-NOVA Marketplace (including the BSC and SS) have been developed with a Software Oriented Architecture based on microservices, in which each Marketplace component has been developed separately and communicates with the others by means of RESTful APIs. This provides flexibility and scalability to the T-NOVA Marketplace in case further functionalities may want to be added in the future.

For the integration of all the different components in the Marketplace, Docker Compose has been selected; each microservice is placed in a different container, and

they are integrated by means of Docker Compose file to coordinate the configuration of all the micro-services.

Table of Contents

Index of Figures

Index of Tables

1. INTRODUCTION

1.1. Objectives and scope

This deliverable presents the activities and results from Task 6.1 of the T-NOVA project. The overall objective of this task is to design and implement the components at marketplace level that are part of the service description schema as well as the information model for the T-NOVA Marketplace including:

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

The current work relies on previous related specification and research phases in the project, including requirements elicitation, that were explained in [1] [2]. The first release of the marketplace service framework components is due to end of December, though it is expected that after integration work between the marketplace and the rest of T-NOVA subsystems, a.k.a, orchestrator and function store, these components could be refined based on the overall integration feedback, e.g. further updates in the T-NOVA Network Service Descriptor (NSD) may be done due to still ongoing work in the T-NOVA Orchestrator platform. This will be reported in the final version of the current report that will be due by end of June 2016.

1.2. T-NOVA commercial framework overview

The T-NOVA Marketplace generic business scenario, depicted in Figure 1-1, reflects the two main commercial relationships that are in T-NOVA: one between the Service Provider (SP) and Function Providers (FPs) to acquire standalone VNFs to compose a Network Service (NS) and the second one between the SP and the Customer (C) who acquire NSs.

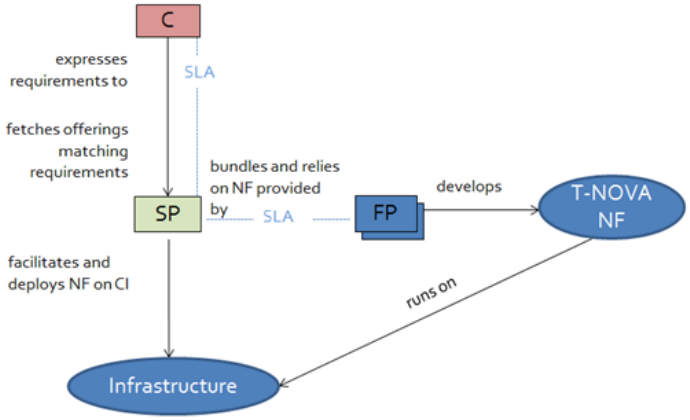

Figure 1-1 Business T-NOVA stakeholders relationships [3]

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, resource requirements, expected performance in the form of SLAs, price, etc.

The Service Provider (SP) that wants 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 and bundling them creating offerings including service description, SLA specification and pricing and that will be exposured 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.

1.3. Service description framework within T-NOVA Marketplace

As explained at specification phase [1] and based on a careful assessment of the state-of-the-art [1], it was decided that the T-NOVA Marketplace includes a catalogue in order to facilitate the exposure of the available services to the T-NOVA customer. This catalogue is named "business service catalogue" in line with TMForum [4] terminology as defined in its "integration framework", in which functional and non-functional aspects of a service based on service oriented principles are defined.

On the other hand, it has been decided to include another independent service framework component as part of the T-NOVA Marketplace, in order to facilitate the service selection management and service configuration processes involving the customer. It is the Service Selection (SS) module, which is in charge of ordering the instantiation of the service to the orchestrator once it has been configured and receiving the correct instantiation reply it updates the accounting accordingly.

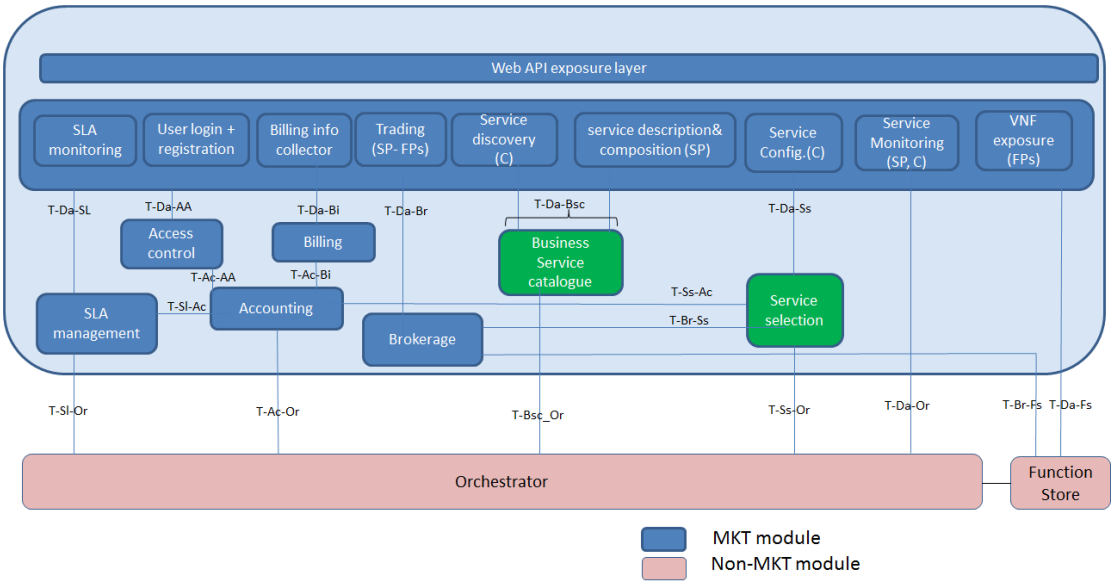

Figure 1-2 - Marketplace architecture

Therefore there are two modules in the marketplace that are in charge of managing the services at the highest level: the business service catalogue (BSC) and the service selection (SS) modules, highlighted in green in Figure 1-2. The VNFs descriptions provided by the FPs are store in the T-NOVA Function Store from where the Marketplace will take them. The Function Store is referenced in the current report but details about its architecture and implementation are in the scope of [5].

1.3.1. T-NOVA Marketplace implementation

As explained in [2], in order to enhance the T-NOVA Marketplace with the modularity specified in previous work [3], the T-NOVA Marketplace implementation is based on a microservices software architecture [2]. This kind of architecture 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(s) and be scaled, deployed and evolved independently.

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.

The Marketplace's software architecture is also REST [6]-based, a set of architectural principles by which it is possible 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. Therefore the Business Service Catalogue and Service Selection modules implementations follow this approach. Details on their request methods for each of them are detailed in this document.

1.4. Document structure

This document is structured as follows:

Firstly, section 2 gives an overview about the main outputs of the survey done in T-NOVA Marketplace specification phase in relation to service description [2], including an update on the activities by the main identified relevant bodies during the last year. Section 3 explains the details of the T-NOVA Marketplace service description framewok including the overview of the T-NOVA information model as well as the specification the T-NOVA descriptors, Network Service Descriptor (NSD) and VNF Descriptor (VNFD). Section 4 focuses in depicting the architecture of the T-NOVA Marketplace components that are part of the service description framework. Then, Section 5 provides the insights for their implementation as well as the main UML diagrams for their design. Section 6 is devoted to the integration of the service description framework modules within the rest of the system so the APIs definitions are collected. Finally section 7 contains the results of the functional verification tests that have been performed for the implemented modules, as well as the report on the fulfilment of the requirements that were gather in the specification phase. The conclusions gathered from the contents of this document are provided in section 8. Annexes A and B highlight the extensions done by T-NOVA Marketplace to ETSI NFV VNFD and NSD respectively.

2. RELEVANT PREVIOUS SOLUTIONS IN THE SOTA

In previous work in T-NOVA [2] the more relevant description languages in the state of art were studied. Also 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. As a conclusion, in the context of NFV, we have identified within T-NOVA there are two lead initiatives in this approach, namely ETSI NFV and TMForum.

2.1. Standarization bodies

2.1.1. ETSI NFV

At T-NOVA specification phase [1] ETSI NFV group had given 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 [7]. No business connotations were provided by ETSI to this respect.

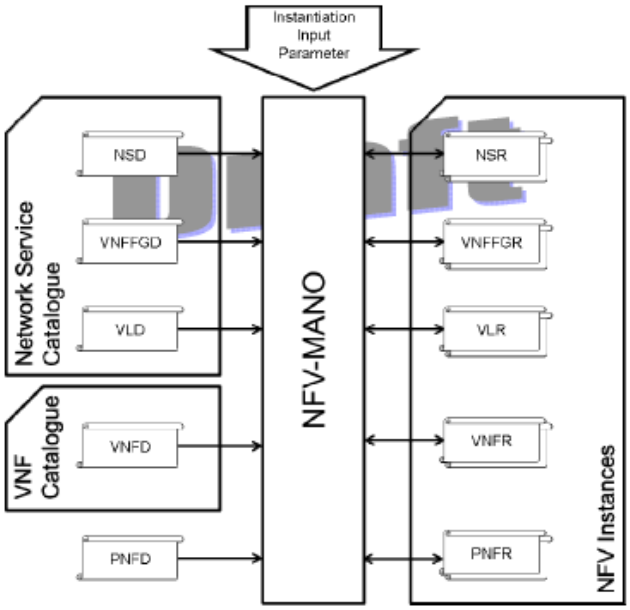

Figure 2-1 Information elements in NFV orchestration [7]

The interface that defines ETSI for NSD descriptor management with the OSS is depicted in Figure 2-, 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 2-2 ETSI MANO Os-Ma-nfvo interface

Table 2-1 collects the NSD fields provided by ETSI [7].

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_dependency	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-per second (cps), number-of-subscribers, no-of-rules, flows-per second, 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 per second.
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 2-1 ETSI NFV Network Service Descriptor fields [7]

2.1.1.1. Update on ETSI NFV phase 2

During 2015 three reports relevant to service specification and its management has been published by ETSI providing further details for the Network Service Descriptor version above as well as its management. These are:

- Management and Orchestration; Os-Ma-Nfvo reference point – Application and Service Management Interface and Information Model Specification [8].
- Report on NFV Information Model. December 2015. [9]
- Management and Orchestration Service Template Specification. December 2015. [10]

This work by ETSI has been released chronologically in parallel to the the T-NOVA Marketplace implementation phase. ETSI has provided further details on the NSD management interface as well as the NSD template, while T-NOVA had already extended ETSI NSD for the implementation of the Marketplace together with business aspects which first version of ETSI NSD was lacking. A contribution from T-NOVA was sent to ETSI NFV as feature proposal in October 2015 in relation to T-NOVA Marketplace. Future work in T-NOVA as part of the standarization efforts in the project will be comparing T-NOVA path in relation to service description and the progress done by ETSI, in order to consider potential contributions and collaborations.

2.1.2. TMForum

The TMForum by means of its *Information Framework (SID, Shared Information/Data model)* [11] 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.

2.1.2.1. Update at December 2015

TMForum released in November 2015 an updated version of the technical report "Information Framework Enhancements to Support ZOOM R15.0.1" [12]. This document describes a portion of the ZOOM information model, which has been integrated into the Information Framework. It defines four concepts fundamental to modeling NFV-based systems (VirtualResource, NetworkFunction, NetworkService, and Graph), as well as two general-purpose concepts (Catalog and Event) that have been used by existing Catalysts, and are also critical for realizing SDN and NFV systems.

This reference has been considered in T-NOVA as it is reflected in section 3.1.

2.2. Other research projects

As it was survey in [2], relevant previous research projects implementing Marketplaces such as FIWARE [13] and XIFI [14] relied on WStore [15] 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.

In WStore, at the same time of creating the offerings, the components that are part of the services of those offerings should be uploaded to the deployed WStore server (repository). In T-NOVA, the analog repository is part of the orchestrator [16]. 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 [17] 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.

Considering the modelling used in WStore, 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.

2.3. Summary about T-NOVA related to the SOTA

As explained above, there are several previous research projects implementing marketplaces, mainly in the context of Future Internet that rely on WStore [15]. In T-NOVA we build our own services storage, using a similar approach of the one provided by WStore but applied to NFV services, including also the VNF description done in the T-NOVA Function Store [5] and considering the service repositories that are needed in NFV at orchestration level. The offerings will be stored in the Business Service Catalogue (BSC) following TMForum best Practices, which will be implemented by means of a database developed with MySQL. This database will be called by the Dashboard using standardized communication by means of a REST API and the SP will create a customized description.

Once the catalog is available, the customer can browse the offerings and select one. The information of the selected service is sent to the service selection module by means, again, of a REST API set of calls. This module has been introduced to ease the adaptation of the generic service to the customer network and send this customization to the orchestrator to proceed with the service components instantiation.

The T-NOVA information model is created as a result of the application of SID TMForum model to ETSI NFV information model, and the T-NOVA descriptors are defined using an extension of ETSI NSD model approach instead of USDL given the particularities of NFV.

3. T-NOVA Service Description Framework Design

When the Service Provider (SP) wants to create a new service to be available through T-NOVA marketplace, he will create an offering including a high level description definition, SLA offer and price, and also the information required for the T-NOVA orchestrator to deploy the service such as service structure and management of behavior, among others. All this information will be structured in the form of the T-NOVA Network Service Descriptor (NSD) that will be store in the Business Service Catalogue.

T-NOVA extends the information model proposed by ETSI NFV adding relevant fields from the marketplace point of view as well as applying the TMForum SID model to the model provided by ETSI NFV.

3.1. Application of TMForum SID model to ETSI NFV information model

Based on the two main standardization bodies identified in section 2.1 as active in defining information models for NFV, T-NOVA has taken the two models in order to apply business aspects from TMForum SID model, to the model proposed by ETSI NFV. This approach is depicted in Figure 3-1.

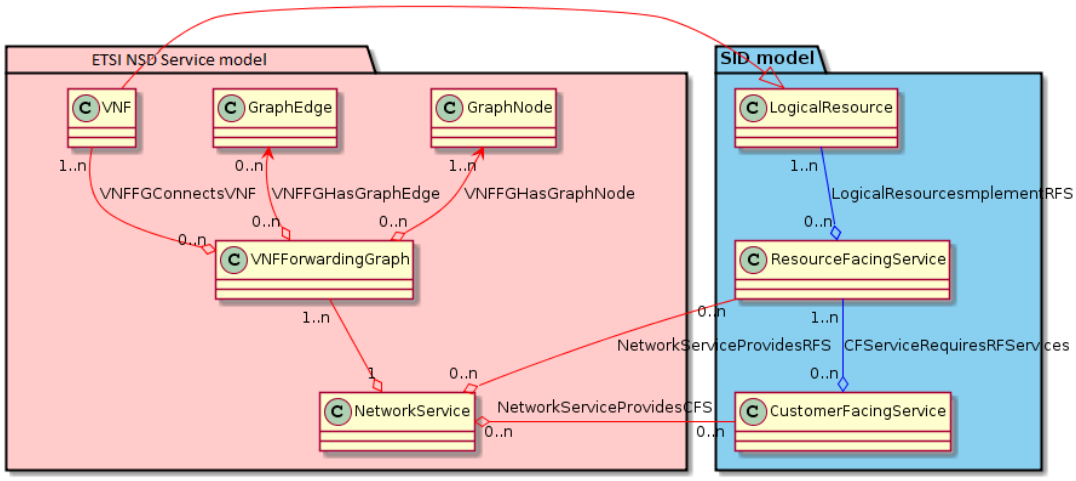

Figure 3-1 Applying SID service model to ETSI NFV information model

3.2. T-NOVA descriptors

Taken ETSI NFV Network Service Descriptor (NSD) and Virtual Network Function Descritor (VNFD) as the baseline, T-NOVA has extended them adding several fields needed according to the requirements defined for T-NOVA Marketplace. Also other fields has been modified or added in T-NOVA from the VNF and Network Services operational point of view as they are managed by the orchestrator [18] [19].

3.2.1. T-NOVA VNFD

The T-NOVA Virtual Network Function Descriptor (VNFD) is the template which describes a VNF in terms of deployment and operational behaviour requirements. The VNFD also contains connectivity, interface and KPIs requirements that can be used by NFV-MANO functional blocks to establish appropriate Virtual Links within the NFVI between VNFC instances, or between a VNF instance and the endpoint interface to other Network Functions.

The T-NOVA VNFD also includes all the information that will allow the commercial activity of VNF through a marketplace, such vendor/owner, SLA, billing models, price, etc.

Figure 3-2 represents T-NOVA VNFD information components relevant to the marketplace and their relation.

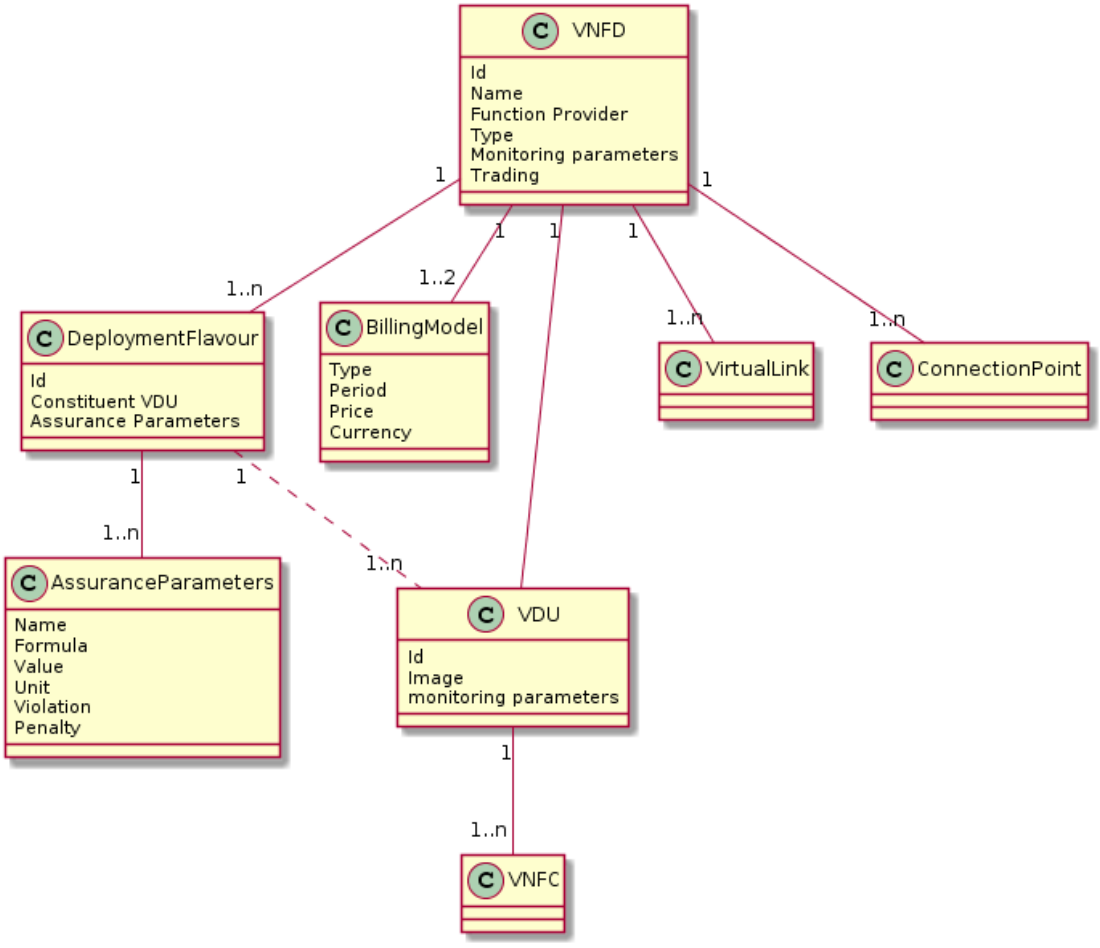

Figure 3-2 T-NOVA VNFD information model

The whole T-NOVA VNFD can be found in [18], while Annex 9 details all the fields in T-NOVA VNFD defined by T-NOVA Marketplace to allow the commercial activity and which has been added on top of ETSI NFV VNFD.

3.2.2. T-NOVA NSD

The Network Service Descriptor (NSD) is defined by ETSI as a deployment template for a network service referencing all other descriptors, which describe components that are part of the network service. The NSD contains the set of static information elements used to instantiate and manage a network service over an NFV-enabled infrastructure.

The T-NOVA NSD represents the reference data model to be considered within the orchestrator and the marketplace, including also all the information that will allow the commercial activity of VNF through a marketplace, such vendor/owner, SLA, billing models, price, etc.

Figure 3-3 represents T-NOVA NSD information components and their relation.

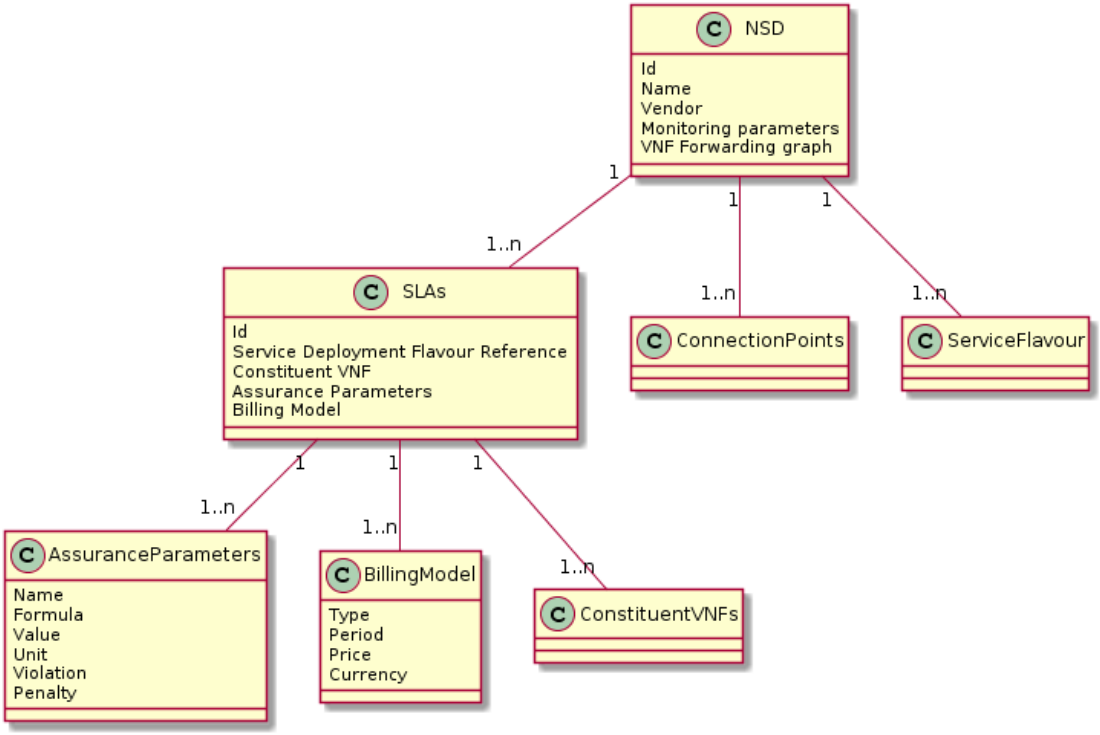

Figure 3-3 T-NOVA NSD Information model

The current version of T-NOVA Network Service Description template can be found in Annex 10, highlighting the T-NOVA Marketplace fields that have been added on top of the NSD proposed by ETSI.

4. Service Description Framework Architecture

As explained in previous sections there are two main modules in the marketplace that are in charge of managing the services at the highest level: the business service catalogue (BSC) and the service selection (SS) modules. On the other hand, the VNFs descriptions provided by the FPs are store in the T-NOVA Function Store from where the Marketplace will take them. The Function Store is referenced in the current section but details about its architecture and implementation are in the scope of [5].

4.1. Business service catalogue

The overall business service catalog internal architecture is depicted in Figure 4-1.

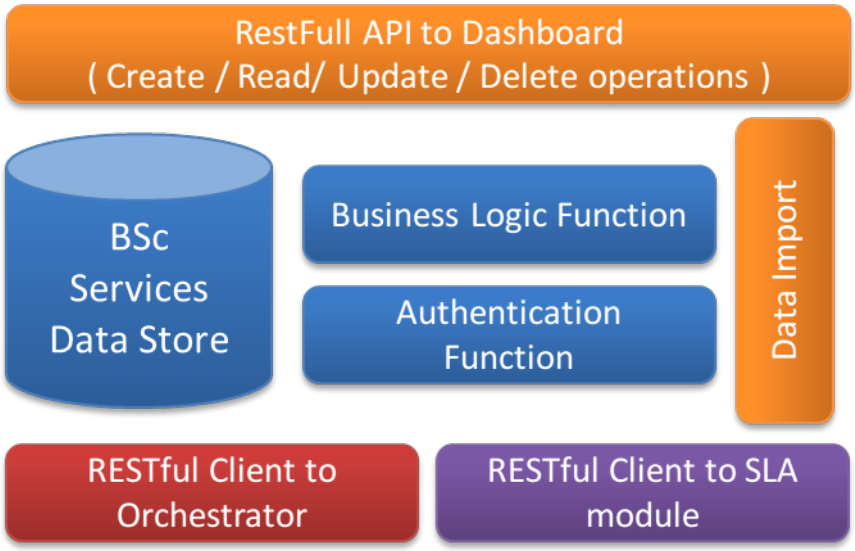

Figure 4-1 Business Service Catalog Architecture

The Business Service Catalog module is the network services repository at marketplace level. The main components of this module are:

- Services Data Store – It is a no SQL data store that contains all the created network services, from the SPs, can be stored.
- Restful API to Dashboard – This is the external Restful interface where SPs and Customers are able to perform to Create / Read / Update / Delete operations.
- Business Logic & Audit & Authentication Functions, are internal parts for the BSc module and their responsibility are to authenticate http requests, making appropriate conversions (e.g. creating SLA templates) and executing the appropriate actions (e.g. store in data store, publish to orchestrator, etc.).
- There are two RESTful clients (adapters) that are responsible for pushing the network services data and corresponding SLA templates to the appropriate modules, such as orchestrator & SLA module.

- Data Import Module – is responsible for importing large number of network services in a low priority process, in order to support bulk operations from SPs.

4.1.1. Workflow

The BSC workflow consists of the following steps:

- The SP creates a NSD and it is passed to BSC though RESTful call (HTTP POST request).
- The BSC temporarily save the provided NSD and generated a unique identifier (nsd_id).
- BSC publishes the NSD to Orchestrator over the provided Restful API.
- BSC generates the SLA template, based on saved NSD and published to SLA micro-services.
- BSC permanently saves the NSD to internal storage engine (mongoDB).

In case of failure at any stage, all the data will be rollbacked in order to avoid any data inconsistency across different micro-services.

4.2. Service Selection

The overall architecture of Service Selection (SS) service is depicted in Figure 4-2.

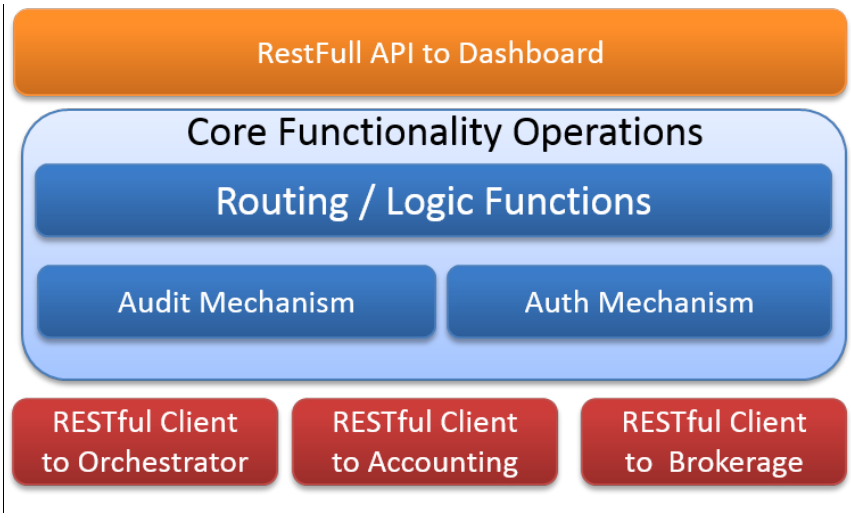

Figure 4-2 Service Selection Service Architecture

According to the requirement defined in [1], the SS service is responsible to activate/terminate a network service that is selected by Customer, to the orchestrator. Furthermore, this micro-service is also responsible to call the accounting service, in order to trigger the charging of the network service, after a successful instantiation response from the orchestrator.

The main components are:

- The Restful API provided in order to accept requests from the dashboard, for network service instantiation.

- The Routing Function is responsible to forward the request to the orchestrator and depending on the answer (in case of success), to forward it to the accounting module.
- Restful adapters to support a rest call to the orchestrator and the accounting module.
- It supports Synchronous and Asynchronous RESTful calls to the orchestrator.

4.2.1. Workflow

The SS workflow consists of the following steps:

- The Customer selects a network service and its NSD is passed to SS micro-service through a RESTful call (HTTP POST request) along with the NS custom configuration that contains mainly the NS connection point to the customer network.
- The SS temporarily saves in its internal cache the requested data, and generates a unique request id.
- The SS micro-service forwards the request to the orchestrator in order to instantiate the network service.
- The SS consults the brokerage module for changes in the final price of the service.
- The SS module publishes to accounting module the needed data as defined in deliverable 6.4 in order to track the usage of the VNFs and NSs for later billing.

In case of failure at any stage, all the data will be rollbacked in order to avoid any data inconsistency across different micro-services.

4.3. Function store

The NF Store is a repository for the VNFs' software images and their metadata descriptions in the form of VNF Descriptors (VNFDs). Figure 4-3 shows a high level architectural description of the relationships of the NF Store and VNFs with the other elements of T-NOVA architecture, namely the Orchestrator and the Marketplace.

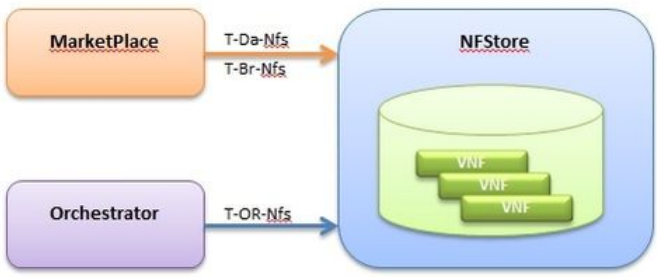

Figure 4-3. Function Store interfaces

Software developers can upload VNF images into the NF Store along with metadata descriptor containing both technical and business related information, such as business description of the cost of using such VNF. The description of the VNFs in the NF Store is made available to the T-NOVA Marketplace. Further details on Function Store architecture and implementation are in the scope of [5], and VNF description will be used by the brokerage module when the trading process takes place before as part of the service composition by the SP [20].

5. IMPLEMENTATION

5.1. Business Service Catalogue

As mentioned in Section 4.1, the BSC acts as a storage engine for network services created by the SP.

The BSC provides the following features:

- It uses JSON format to store and retrieve information.
- It is a document oriented storage engine, meaning that only one document (network service template) is capable of storing all the information required to define each network service.
- The schema can also be design on-the-fly, leading to a much faster development.
- Dynamic queries, and text search is also supported in order to retrieve network services based on complicated constraints, such as SLA parameters (e.g. cost, penalties, etc.).

The micro-service is majorly implemented in Java as a standalone application and uses embedded tomcat, to support RESTful API. Furthermore, the BSC micro-service provided an internal documentation for an easy integration with other modules by developers.

The UML class diagrams below show the implementation of the NSD (Figure 5-1) and the generated SLA template (Figure 5-2) that is used internally.

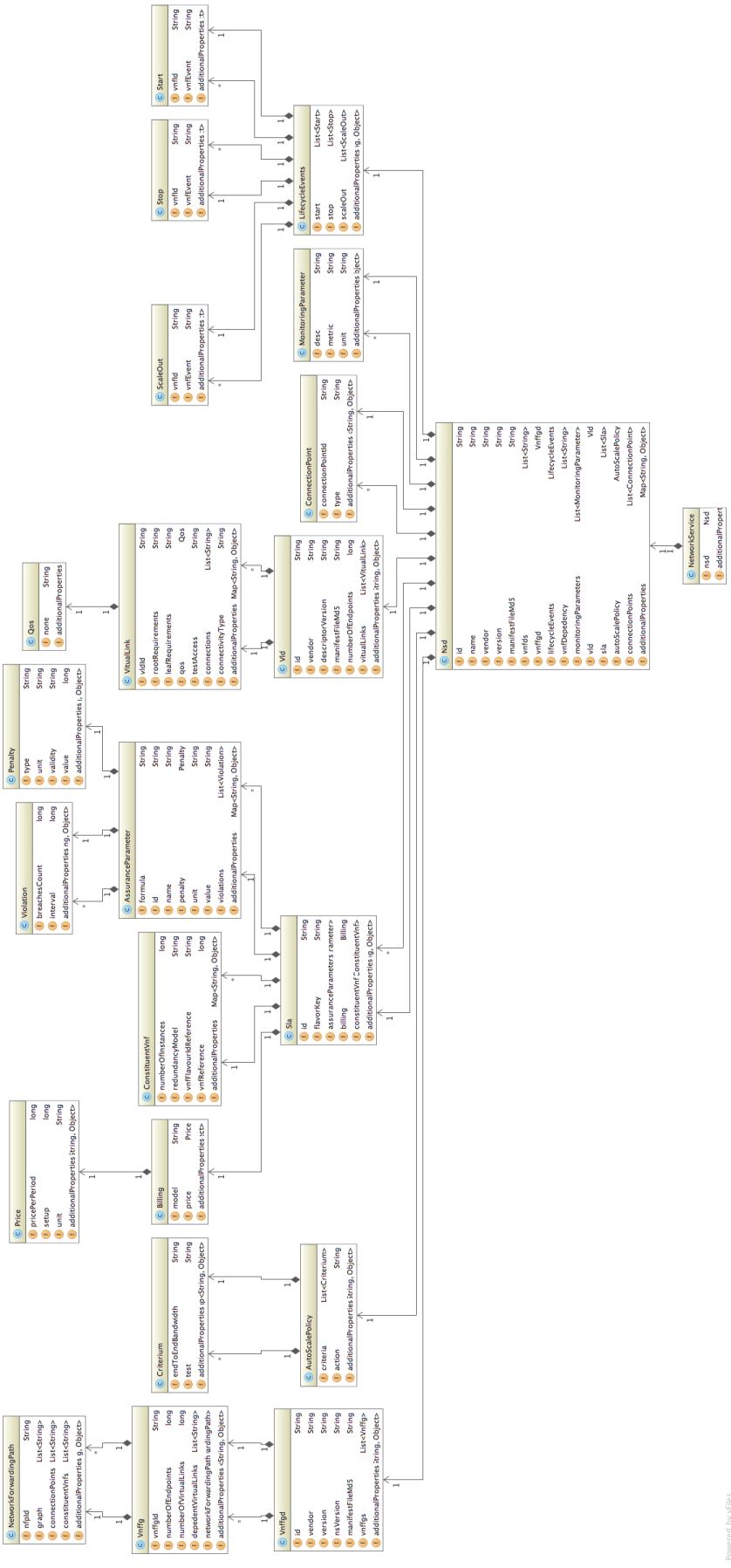

Figure 5-1. UML Diagram of NSD

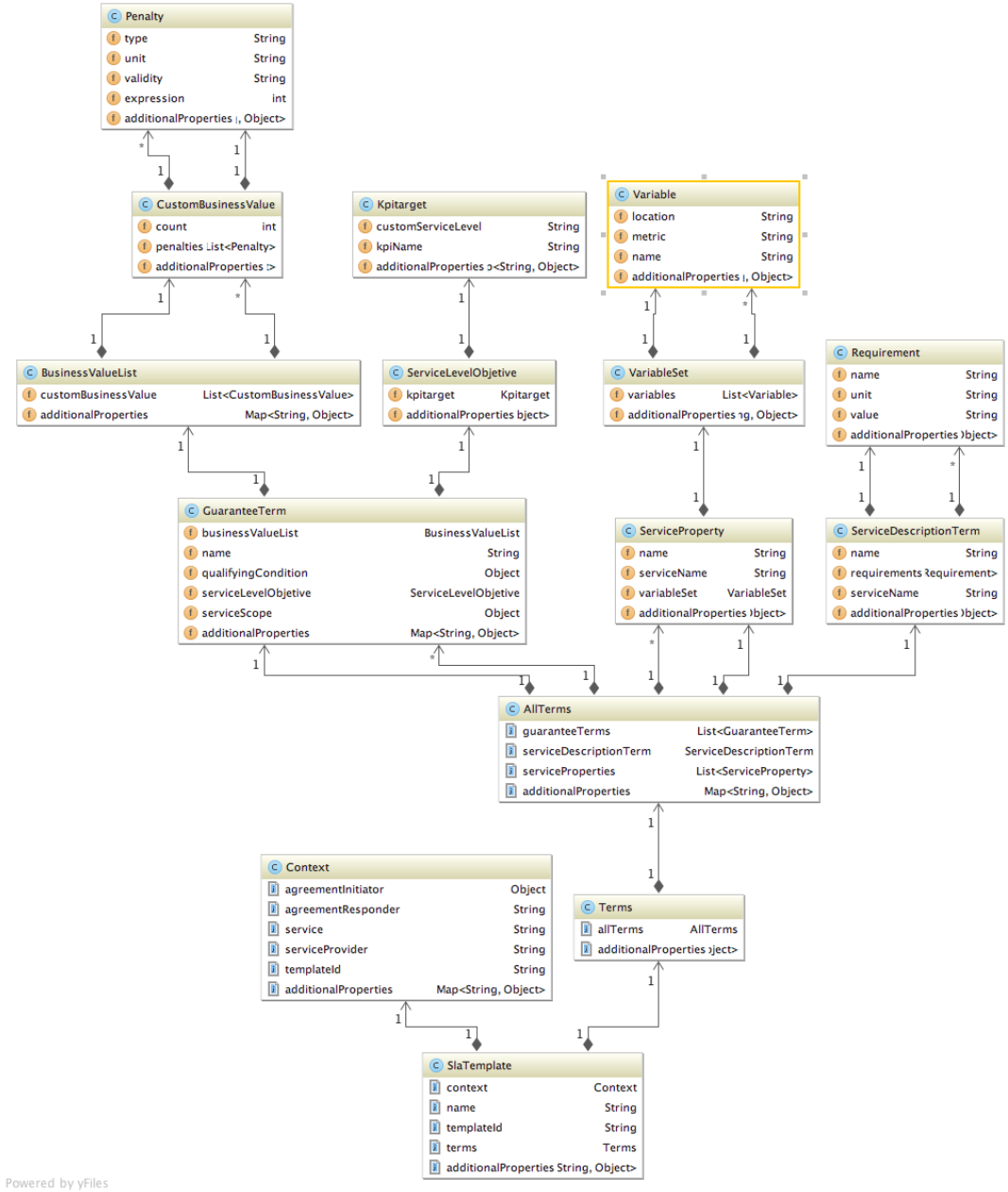

Figure 5-2 Generated Sla Template

As it can be observed from the class diagrams, both are supporting «additionalProperties» field. This feature enables the microservice to support additional fields, without the need of the modification of the micro-service.

5.2. Service selection module

This micro-service is majorly implemented in Java as a standalone application and uses embedded tomcat, to support a RESTful API. Furthermore, the SS micro-service provides an internal documentation for an easy integration with other modules by the developers.

The following UML class diagrams show the implementation of different requests and replies of SS micro-service.

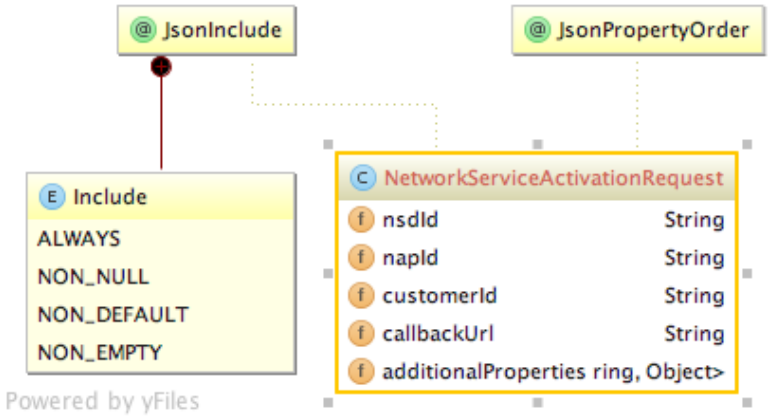

Figure 5-3 SS Network Service Instantiation Request

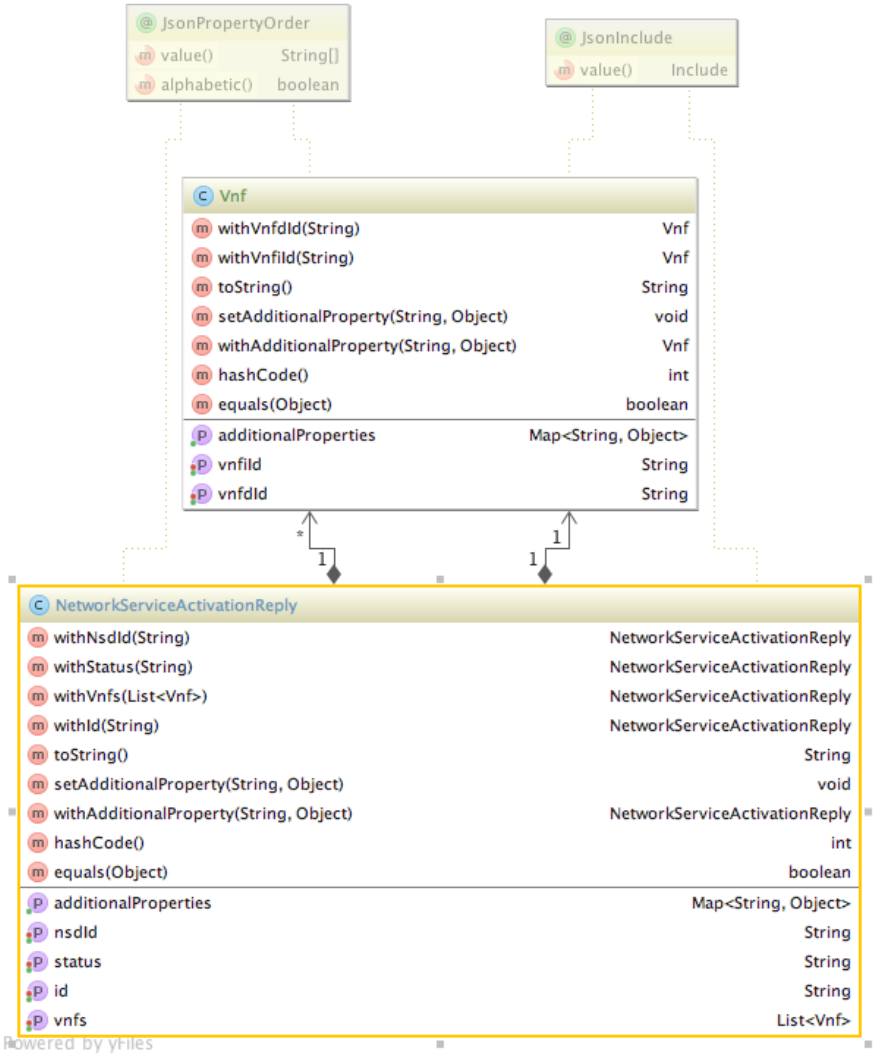

Figure 5-4 SS Network Service Instantiation Orchestrator response

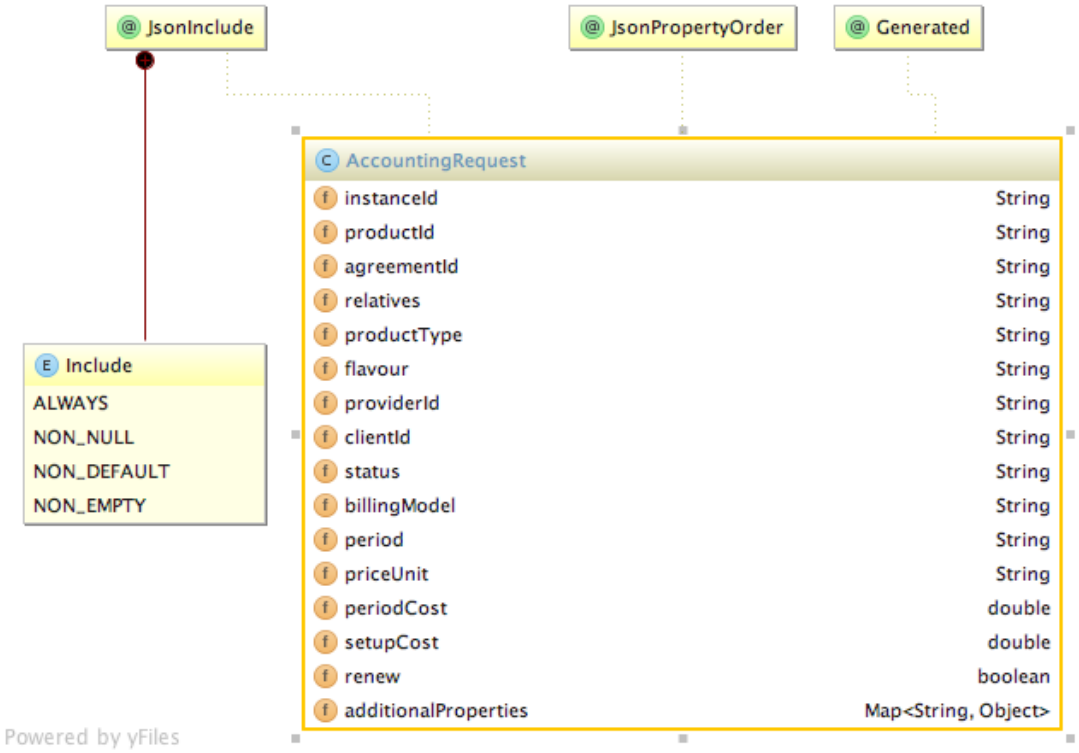

Figure 5-5 SS Request to Accounting micro-service

As can observed from the class diagrams, an «additionalProperties» field exists to support additional parameters to the Restful requests/replies in order to support additional modifications

The SS micro-services, as it is mentioned above, support also asynchronous Restful operations to the orchestrator, in order to avoid possible timeouts during network service instantiation by the orchestrator.

6. INTEGRATION

6.1. Business Service Catalogue

The BSC main interfaces are with the Dashboard and Orchestrator. Also the SLA module has a less relevant interface to receive the SLA template generated by the Business Service Catalogue based on the NSD.

The Business Service Catalogue microservice is deployed within the Marketplace docker structure in a separate container. To be able to do so, the general *docker-compose* file needs to have a section dedicated to the BSC module and its dependencies. Once we have the dependencies fulfilled, it's time to configure the container. A *dockerfile* tells how to create the container and the script *DockerStart.sh* tells how to execute it.

The BSC relies on a noSQL mongoDB database that is deployed in a different container (it is used by several modules) and database schema is generated or updated automatically upon the start-up of the microservice.

6.1.1. Business Service Catalogue module API definition

The operations supported by the Business Service Catalogue (BSC) API are exposed to the following modules: dashboard (T-Da-Bs), orchestrator (T-Or-Bs) and SLA (T-Sl-Bs).

6.1.1.1. Dashboard interface (T-Da-Bs)

The following operations supported by the BSC API are exposed to the Dashboard.

(a) Stores NSD to BSC

URL	/service/catalog
Type	POST
Headers	Accept: application/json Content-type: application/json
Parameters	N / A
Response code	▪ 200: OK ▪ 409: HTTP CONFLICT ▪ 400: Bad Request, due to problems in body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`POST /service/catalog HTTP/1.1`

POST Item Example


```
{
  "nsd" : {
    "name" : "network-service-1",
    "vendor" : "T-NOVA",
    "version" : "1.0",
    "manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2",
    "vnfds" : [ "52439e7c-c85c-4bae-88c4-8ee8da4c5485" ],
    "vnffgd" : {
      "id" : "418420e3-e2a8-4338-bc8c-be685548bad2",
      "vendor" : "NCSRD",
      "version" : "1.0",
      "ns_version" : "1.0.0",
      "manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2"
,
      "vnffgs" : [ {
        "vnffg_id" : "vnffg0",
        "number_of_endpoints" : 2,
        "number_of_virtual_links" : 2,
        "depedent_virtual_links" : [ "vld1", "vld2", "vld3" ],
        "network_forwarding_path" : [ {
          "nfp_id" : "nfp1",
          "graph" : [ "vld1", "vld2" ],
          "connection_points" : [ "data0", "data1" ],
          "constituent_vnfs" : [ "vnfd0:flavour0" ]
        } ]
      } ]
    },
    "lifecycle_events" : {
      "start" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "start"
      } ],
      "stop" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "stop"
      } ],
```

```
      "scale_out" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "scale_out"
      } ]
    },
    "vnf_depedency" : [ "52439e7c-c85c-4bae-88c4-8ee8da4c5485:
52439e7c-c85c-4bae-88c4-8ee8da4c5485" ],
    "monitoring_parameters" : [ {
      "desc" : "Availability",
      "metric" : "availability",
      "unit" : "%"
    }, {
      "desc" : "CPU Usage",
      "metric" : "cpu_usage",
      "unit" : "%"
    } ],
    "vld" : {
      "id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
      "vendor" : "T-NOVA",
      "descriptor_version" : "1.0",
      "manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2"
,
      "number_of_endpoints" : 2,
      "vitual_links" : [ {
        "vld_id" : "vld0",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
        "qos" : {
          "none" : "not used"
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data0", "data0" ],
        "connectivity_type" : "E-Line"
      }, {
        "vld_id" : "vld1",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
```

```
        "qos" : {
          "average" : "test",
          "peak" : "peak_test",
          "burst" : "burst"
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data1" ],
        "connectivity_type" : "E-Line"
      } ]
    },
    "sla" : [ {
      "id" : "flavor0",
      "flavor_key" : "Basic",
      "assurance_parameters" : [ {
        "formula" : "MIN(VNF:623.availability)",
        "id" : "availability",
        "name" : "availability",
        "penalty" : {
          "type" : "discount",
          "unit" : "%",
          "validity" : "P1D",
          "value" : 10
        },
        "unit" : "%",
        "value" : "LT(99)",
        "violations" : [ {
          "breaches_count" : 2,
          "interval" : 360
        } ]
      } ],
      "billing" : {
        "model" : "PAYG",
        "price" : {
          "price_per_period" : 20,
          "setup" : 5,
          "unit" : "EUR"
```

```
        }
      },
      "constituent_vnf" : [ {
        "number_of_instances" : 1,
        "redundancy_model" : "Active",
        "vnf_flavour_id_reference" : "gold",
        "vnf_reference" : 623
      } ]
    } ],
    "auto_scale_policy" : {
      "criteria" : [ {
        "end-to-end bandwidth" : "10Mbps"
      }, {
        "test" : "test"
      } ],
      "action" : "upgrade"
    },
    "connection_points" : [ {
      "connection_point_id" : "mgnt0",
      "type" : "ip"
    }, {
      "connection_point_id" : "data0",
      "type" : "bridge"
    }, {
      "connection_point_id" : "stor0",
      "type" : "bridge"
    } ]
  }
}
```


Table 6-1 API operation to store NSD to BSC

(b) Modify a Network Service

URL	/service/catalog/{id}
Type	PUT
Headers	Accept: application/json

	Content-type: application/json
Parameters	• Id – Network Service identifier (unique)
Response code	▪ 200: OK ▪ 404: NOT_FOUND ▪ 400: Bad Request, due to problems in the body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`PUT /service/catalog HTTP/1.1`

PUT Item Example	{
	"nsd" : {
	“id” : “1232323”,
	"name" : "network-service-1",
	"vendor" : "T-NOVA",
	"version" : "1.0",
	"manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2",
	"vnfds" : ["52439e7c-c85c-4bae-88c4-8ee8da4c5485"],
	"vnffgd" : {
	"id" : "418420e3-e2a8-4338-bc8c-be685548bad2",
	"vendor" : "NCSRD",
	"version" : "1.0",
	"ns_version" : "1.0.0",
	"manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2"
	,
	"vnffgs" : [{
	"vnffg_id" : "vnffg0",
	"number_of_endpoints" : 2,
	"number_of_virtual_links" : 2,
	"depedent_virtual_links" : ["vld1", "vld2", "vld3"],
	"network_forwarding_path" : [{
	"nfp_id" : "nfp1",
	"graph" : ["vld1", "vld2"],
	"connection_points" : ["data0", "data1"],
	"constituent_vnfs" : ["vnfd0:flavour0"]
	}]
	}]
	},

```
    "lifecycle_events" : {
      "start" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "start"
      } ],
      "stop" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "stop"
      } ],
      "scale_out" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "scale_out"
      } ]
    },
    "vnf_depedency" : [ "52439e7c-c85c-4bae-88c4-8ee8da4c5485:
52439e7c-c85c-4bae-88c4-8ee8da4c5485" ],
    "monitoring_parameters" : [ {
      "desc" : "Availability",
      "metric" : "availability",
      "unit" : "%"
    }, {
      "desc" : "CPU Usage",
      "metric" : "cpu_usage",
      "unit" : "%"
    } ],
    "vld" : {
      "id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
      "vendor" : "T-NOVA",
      "descriptor_version" : "1.0",
      "manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2"
,
      "number_of_endpoints" : 2,
      "vitual_links" : [ {
        "vld_id" : "vld0",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
        "qos" : {
```

```
          "none" : "not used"
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data0", "data0" ],
        "connectivity_type" : "E-Line"
      }, {
        "vld_id" : "vld1",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
        "qos" : {
          "average" : "test",
          "peak" : "peak_test",
          "burst" : "burst"
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data1" ],
        "connectivity_type" : "E-Line"
      } ]
    },
    "sla" : [ {
      "id" : "flavor0",
      "flavor_key" : "Basic",
      "assurance_parameters" : [ {
        "formula" : "MIN(VNF:623.availability)",
        "id" : "availability",
        "name" : "availability",
        "penalty" : {
          "type" : "discount",
          "unit" : "%",
          "validity" : "P1D",
          "value" : 10
        },
        "unit" : "%",
        "value" : "LT(99)",
        "violations" : [ {
          "breaches_count" : 2,
```

```
          "interval" : 360
        } ]
      } ],
      "billing" : {
        "model" : "PAYG",
        "price" : {
          "price_per_period" : 20,
          "setup" : 5,
          "unit" : "EUR"
        }
      },
      "constituent_vnf" : [ {
        "number_of_instances" : 1,
        "redundancy_model" : "Active",
        "vnf_flavour_id_reference" : "gold",
        "vnf_reference" : 623
      } ]
    } ],
    "auto_scale_policy" : {
      "criteria" : [ {
        "end-to-end bandwidth" : "10Mbps"
      }, {
        "test" : "test"
      } ],
      "action" : "upgrade"
    },
    "connection_points" : [ {
      "connection_point_id" : "mgnt0",
      "type" : "ip"
    }, {
      "connection_point_id" : "data0",
      "type" : "bridge"
    }, {
      "connection_point_id" : "stor0",
      "type" : "bridge"
    } ]
```

	<pre> } }</pre>

Table 6-2 API operation to modify a Network Service

(c) Delete a Network Service

URL	/service/catalog/{id}
Type	DELETE
Headers	Content-type: application/json
Parameters	• Id – Network Service identifier (unique)
Response code	▪ 201: OK ▪ 404: NOT_FOUND ▪ 400: Bad Request, due to problems in the body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	<pre>DELETE /service/catalog/1232323 HTTP/1.1</pre>

Table 6-3 API operation to delete a Network Service

(d) Retrieve a Network Service Based on NSD_ID

URL	/service/catalog/{id}
Type	GET
Headers	Content-type: application/json
Parameters	• Id – Network Service identifier (unique)
Response code	▪ 200: OK ▪ 404: NOT_FOUND ▪ 400: Bad Request, due to problema in body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	<pre>GET /service/catalog/1232323 HTTP/1.1</pre>

Response element example	<pre>{
	  "nsd" : {
	    "id" : "1232323",
	    "name" : "network-service-1",
	    "vendor" : "T-NOVA",
	    "version" : "1.0",
	    "manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2",
	    "vnfds" : ["52439e7c-c85c-4bae-88c4-8ee8da4c5485"],
	    "vnffgd" : {
	      "id" : "418420e3-e2a8-4338-bc8c-be685548bad2",
	      "vendor" : "NCSRD",
	      "version" : "1.0",</pre>

```
      "ns_version" : "1.0.0",
      "manifest_file_md5" :
"fa8773350c4c236268f0bd7807c8a3b2",
      "vnffgs" : [ {
        "vnffg_id" : "vnffg0",
        "number_of_endpoints" : 2,
        "number_of_virtual_links" : 2,
        "depedent_virtual_links" : [ "vld1", "vld2", "vld3" ],
        "network_forwarding_path" : [ {
          "nfp_id" : "nfp1",
          "graph" : [ "vld1", "vld2" ],
          "connection_points" : [ "data0", "data1" ],
          "constituent_vnfs" : [ "vnfd0:flavour0" ]
        } ]
      } ]
    },
    "lifecycle_events" : {
      "start" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "start"
      } ],
      "stop" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "stop"
      } ],
      "scale_out" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "scale_out"
      } ]
    },
    "vnf_depedency" : [ "52439e7c-c85c-4bae-88c4-
8ee8da4c5485:52439e7c-c85c-4bae-88c4-8ee8da4c5485" ],
    "monitoring_parameters" : [ {
      "desc" : "Availability",
      "metric" : "availability",
      "unit" : "%"
    }, {
      "desc" : "CPU Usage",
      "metric" : "cpu_usage",
      "unit" : "%"
    } ],
    "vld" : {
      "id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
      "vendor" : "T-NOVA",
      "descriptor_version" : "1.0",
      "manifest_file_md5" :
"fa8773350c4c236268f0bd7807c8a3b2",
      "number_of_endpoints" : 2,
      "vitual_links" : [ {
        "vld_id" : "vld0",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
        "qos" : {
          "none" : "not used"
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data0", "data0" ],
        "connectivity_type" : "E-Line"
      }, {
```

```
        "vld_id" : "vld1",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
        "qos" : {
          "average" : "test",
          "peak" : "peak_test",
          "burst" : "burst"
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data1" ],
        "connectivity_type" : "E-Line"
      } ]
    },
    "sla" : [ {
      "id" : "flavor0",
      "flavor_key" : "Basic",
      "assurance_parameters" : [ {
        "formula" : "MIN(VNF:623.availability)",
        "id" : "availability",
        "name" : "availability",
        "penalty" : {
          "type" : "discount",
          "unit" : "%",
          "validity" : "P1D",
          "value" : 10
        },
        "unit" : "%",
        "value" : "LT(99)",
        "violations" : [ {
          "breaches_count" : 2,
          "interval" : 360
        } ]
      } ],
      "billing" : {
        "model" : "PAYG",
        "price" : {
          "price_per_period" : 20,
          "setup" : 5,
          "unit" : "EUR"
        }
      },
      "constituent_vnf" : [ {
        "number_of_instances" : 1,
        "redundancy_model" : "Active",
        "vnf_flavour_id_reference" : "gold",
        "vnf_reference" : 623
      } ]
    } ],
    "auto_scale_policy" : {
      "criteria" : [ {
        "end-to-end bandwidth" : "10Mbps"
      }, {
        "test" : "test"
      } ],
      "action" : "upgrade"
    },
    "connection_points" : [ {
      "connection_point_id" : "mgnt0",
      "type" : "ip"
```

	<pre>}, {
	"connection_point_id" : "data0",
	"type" : "bridge"
	}, {
	"connection_point_id" : "stor0",
	"type" : "bridge"
	}]
	}
	}</pre>

Table 6-4 API operation to retrieve a Network Service Based on NSD_ID

(e) Retrieve all Network Services

URL	/service/catalog/
Type	GET
Headers	Content-type: application/json
Parameters	• Id – Network Service identifier (unique)
Response code	▪ 201: OK ▪ 204: NO_CONTENT ▪ 400: Bad Request, due to problema in body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`GET /service/catalog/ HTTP/1.1`
Response element example	`JSON array of Network Services.`

Table 6-5 API operation to retrieve all Network Services

(f) Retrieve all Network Services based on maximum price

URL	/service/catalog/price/max={max_price}
Type	GET
Headers	Content-type: application/json
Parameters	• Max_price - Maximum price
Response code	▪ 201: OK ▪ 204: NO_CONTENT ▪ 501: Internal Server Error
Request example	`GET /service/catalog/price/max=1000 HTTP/1.1`
Response element example	`JSON array of Network Services meet the constraint of max price.`

Table 6-6 API operation to retrieve all Network Services based on maximum price

(g) Retrieve all Network Services based price range

URL	/service/catalog/price/from={from}&to={to}
Type	GET
Headers	Content-type: application/json
Parameters	
Response code	▪ 201: OK ▪ 204: NO_CONTENT ▪ 501: Internal Server Error
Request example	`GET /service/catalog/price/from=100&to=100 HTTP/1.1`
Response element example	`JSON array of Network Services meet the constraint of the price range`

Table 6-7 API operation to retrieve all Network Services based price range

6.1.2. Calls to other APIs

6.1.2.1. Orchestrator interface (T-Or-Bs)

In order to implement T-Or-Bs it is the BSC module the one that calls the Orchestrator API for monitoring information. The format of this call is:

(a) BSC publishes a network service to the orchestrator.

URL	http://<orchestrator_url/network-services
Type	POST
Headers	Accept : application/json Content-type: application/json
Parameters	N / A
Response code	▪ 200: OK ▪ 400: Bad Request, due to problema in body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`POST /<orchestrator_url/network-services HTTP/1.1`
POST Request Example	`{ "nsd" : { "id" : "10000000000000", "name" : "network-service-1", "vendor" : "T-NOVA", "version" : "1.0", "manifest_file_md5" : "fa8773350c4c236268f0bd7807c8a3b2",`

```
    "vnfds" : [ "52439e7c-c85c-4bae-88c4-8ee8da4c5485" ],
    "vnffgd" : {
      "id" : "418420e3-e2a8-4338-bc8c-be685548bad2",
      "vendor" : "NCSRD",
      "version" : "1.0",
      "ns_version" : "1.0.0",
      "manifest_file_md5" :
"fa8773350c4c236268f0bd7807c8a3b2",
      "vnffgs" : [ {
        "vnffg_id" : "vnffg0",
        "number_of_endpoints" : 2,
        "number_of_virtual_links" : 2,
        "depedent_virtual_links" : [ "vld1", "vld2", "vld3" ],
        "network_forwarding_path" : [ {
          "nfp_id" : "nfp1",
          "graph" : [ "vld1", "vld2" ],
          "connection_points" : [ "data0", "data1" ],
          "constituent_vnfs" : [ "vnfd0:flavour0" ]
        } ]
      } ]
    },
    "lifecycle_events" : {
      "start" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "start"
      } ],
      "stop" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "stop"
      } ],
      "scale_out" : [ {
        "vnf_id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
        "vnf_event" : "scale_out"
      } ]
    },
    "vnf_depedency" : [ "52439e7c-c85c-4bae-88c4-
8ee8da4c5485:52439e7c-c85c-4bae-88c4-8ee8da4c5485" ],
    "monitoring_parameters" : [ {
      "desc" : "Availability",
      "metric" : "availability",
      "unit" : "%"
    }, {
      "desc" : "CPU Usage",
      "metric" : "cpu_usage",
      "unit" : "%"
    } ],
    "vld" : {
      "id" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485",
      "vendor" : "T-NOVA",
      "descriptor_version" : "1.0",
      "manifest_file_md5" :
"fa8773350c4c236268f0bd7807c8a3b2",
      "number_of_endpoints" : 2,
      "vitual_links" : [ {
        "vld_id" : "vld0",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
        "qos" : {
          "none" : "not used"
```

```
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data0", "data0" ],
        "connectivity_type" : "E-Line"
      }, {
        "vld_id" : "vld1",
        "root_requirements" : "10Gbs",
        "leaf_requirements" : "10Gbs",
        "qos" : {
          "average" : "test",
          "peak" : "peak_test",
          "burst" : "burst"
        },
        "test_access" : "none",
        "connections" : [ "vnf0:data1" ],
        "connectivity_type" : "E-Line"
      } ]
    },
    "sla" : [ {
      "id" : "flavor0",
      "flavor_key" : "Basic",
      "assurance_parameters" : [ {
        "formula" : "MIN(VNF:623.availability)",
        "id" : "availability",
        "name" : "availability",
        "penalty" : {
          "type" : "discount",
          "unit" : "%",
          "validity" : "P1D",
          "value" : 10
        },
        "unit" : "%",
        "value" : "LT(99)",
        "violations" : [ {
          "breaches_count" : 2,
          "interval" : 360
        } ]
      } ],
      "billing" : {
        "model" : "PAYG",
        "price" : {
          "price_per_period" : 20,
          "setup" : 5,
          "unit" : "EUR"
        }
      },
      "constituent_vnf" : [ {
        "number_of_instances" : 1,
        "redundancy_model" : "Active",
        "vnf_flavour_id_reference" : "gold",
        "vnf_reference" : 623
      } ]
    } ],
    "auto_scale_policy" : {
      "criteria" : [ {
        "end-to-end bandwidth" : "10Mbps"
      }, {
        "test" : "test"
      } ],
```

	<pre>"action" : "upgrade"
	},
	"connection_points" : [{
	"connection_point_id" : "mgnt0",
	"type" : "ip"
	}, {
	"connection_point_id" : "data0",
	"type" : "bridge"
	}, {
	"connection_point_id" : "stor0",
	"type" : "bridge"
	}]
	}
	}</pre>

Table 6-8 API call fro BSC to publishes a network service to the orchestrator

Orchestrator API can be found in [19].

6.1.2.2. SLA interface (T-Or-Bs)

In order to implement T-Bs-SL it is the BSC module the one that calls the SLA API to sent SLA template. The format of this call is:

(a) BSC publishes a SLA template to the SLA module.

URL	http://<sla_modue>/templates
Type	POST
Headers	Accept: application/json Content-type: application/json
Parameters	N / A
Response code	▪ 200: OK ▪ 400: Bad Request, due to problema in body.
Request example	<pre>POST /<sla_host>/templates HTTP/1.1</pre>

POST Item Example	<pre>{
	"context" : {
	"agreementInitiator" : null,
	"agreementResponder" : "T-NOVA",
	"service" : "network-service-11.0",
	"serviceProvider" : "AgreementResponder",
	"templateId" : "ns10000000000000Basic"
	},
	"name" : "network-service-1Basic",
	"templateId" : "ns10000000000000Basic",
	"terms" : {
	"allTerms" : {
	"guaranteeTerms" : [{
	"businessValueList" : {
	"customBusinessValue" : [{
	"count" : 1,</pre>

```
            "penalties" : [ {
              "type" : "discount",
              "unit" : "%",
              "validity" : "P1D",
              "expression" : 10
            } ]
          } ]
        },
        "name" : "availability",
        "qualifyingCondition" : null,
        "serviceLevelObjetive" : {
          "kpitarget" : {
            "customServiceLevel" : "{\"policies\":
[{\"count\": 2, \"interval\": 360}],\"constraint\" :
\"availability LT 99\"}",
            "kpiName" : "availability"
          }
        },
        "serviceScope" : null
      } ],
      "serviceDescriptionTerm" : {
        "name" : "requirements",
        "requirements" : [ {
          "name" : "VNF",
          "unit" : "-",
          "value" : "52439e7c-c85c-4bae-88c4-8ee8da4c5485"
        } ],
        "serviceName" : "Basic"
      },
      "serviceProperties" : [ {
        "name" : "MonitoredMetrics",
        "serviceName" : "default",
        "variableSet" : {
          "variables" : [ {
            "location" : "/monitor/availability",
            "metric" : "xs:double",
            "name" : "availability"
          } ]
        }
      } ]
    }
  }
}
```


Table 6-9 API call for the BSC to publish a SLA template to the SLA module

The SLA API can be found in [21].

6.2. Service selection module

The SS micro-services interfaces are with the Dashboard, Accounting, Brokerage module and Orchestrator.

The Service Selection (SS) microservice is deployed within the Marketplace docker structure in a separate container. To be able to do so, the general *docker-compose* file needs to have a section dedicated to the Accounting module and its dependencies.

Once we have the dependencies fulfilled, it's time to configure the container. A *dockerfile* tells how to create the container and the script *DockerStart.sh* tells how to execute it.

6.2.1. Service Selection module API definition

The operations supported by the Service Selection module (Ss) API are exposed to the dashboard (T-Da-Ss)

6.2.1.1. Dashboard interface (T-Da-Ss)

The following operations supported by the SS API are exposed to the Dashboard.

(a) Instantiate a Network Service

URL	/service/selection
Type	POST
Headers	Accept: application/json Content-type: application/json
Parameters	N / A
Response code	▪ 200: OK ▪ 404: NOT_FOUND ▪ 400: Bad Request, due to problems in the body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`POST /service/selection HTTP/1.1`
POST Item Example	`{ "nsd_id" : "1232323", "customer_id" : "network-service-1", "nap_id" : "T-NOVA", "callback_url" : "1.0" }`

Table 6-10 API operation for Network Service Instantiation

(b) Terminate a Network Service

URL	/service/selection/{id}/terminate
Type	GET
Headers	Content-type: application/json
Parameters	N / A
Response code	▪ 200: OK ▪ 404: NOT_FOUND ▪ 400: Bad Request, due to problems in the body, or failure to publish in other modules.

	▪ 501: Internal Server Error
Request example	`GET /service/selection/{1234}/terminate HTTP/1.1`

Table 6-11 API operation for Network Service Termination

6.2.2. Calls to other APIs

6.2.2.1. Accounting interface (T-Ac-Ss)

In order to implement T-Ac-Ss it is the SS module the one that calls the Accounting API. The format of this call is:

URL	http://<accouting>/accounts
Type	POST
Headers	Accept: application/json Content-type: application/json
Parameters	N / A
Response code	▪ 200: OK ▪ 400: Bad Request, due to problems in the body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`POST /<accouting_url/accounts HTTP/1.1`

POST Item Example	<pre>{
	"instanceId" : "id01"
	"productId": "vnf1",
	"agreementId":: "s1vnf",
	"relatives": "id02",
	"productType": "nvf",
	"flavour": "qold",
	"providerId": "f1",
	"clientId": "p1",
	"status": "stopped",
	"billingModel": "PAYG",
	"period": "P1D",
	"priceUnit","EUR",
	"periodCost","1.5",
	"setupCost","2",
	"renew": true
	}</pre>

Table 6-12 API call from the SS to accounting for instances tracking

More details about accounting API can be found in [21].

6.2.2.2. Orchestrator interface (T-Or-Ss)

In order to implement T-Or-Ss it is the SS module the one that calls the Orchestrator API:

(a) SS requests to instantiate a Network Service to the Orchestrator

URL	http://<orchestrator_url/ns_instances
Type	POST
Headers	Accept: application/json Content-type: application/json
Parameters	N / A
Response code	▪ 200: OK ▪ 404: NOT_FOUND ▪ 400: Bad Request, due to problema in body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`POST /<orchestrator_url/ns-instances HTTP/1.1`
POST Item Example	`{ "nsd_id" : "1232323", "customer_id" : "network-service-1", "nap_id" : "T-NOVA", "callback_url" : "1.0" }`
Orchestrator Reply	`{ "id":1, "nsd_id":"test -1", "status":"INSTANTIATED", "created_at":"2015-11-18T14:35:46.478Z", "updated_at":"2015-11-18T14:35:46.478Z", "vnfs":[ {"id":1, "vnfi_id":"aaabbbbccccdddd", "vnfd_id":"vnf_demo1_0" }, { "id":2, "vnfi_id":"aaabbbbccccdddd", "vnfd_id":"vnf_demo2_0" }]}`

Table 6-13 API call from SS to request Network Service instantiation to the Orchestrator

URL	/orchestrator_url/{id}/terminate
Type	PUT
Headers	Content-type: application/json

Parameters	N / A
Response code	▪ 200: OK ▪ 404: NOT_FOUND ▪ 400: Bad Request, due to problems in the body, or failure to publish in other modules. ▪ 501: Internal Server Error
Request example	`PUT/<orchestrator_url/{1234}/terminate HTTP/1.1`

Table 6-14 API call from SS to request Network Service termination to the Orchestrator

6.2.2.3. Brokerage interface (T-Br-Ss)

In order to implement T-Br-Ss it is the SS module the one that calls the Brokerage API to get any possible update on the price as a result of the trading process:

URL	broker_base_url>/trade/?id&vnf_id
Type	GET
Headers	Accept: application/json Content-type: application/json
Parameters	N / A
Response code	HTTP/1.1 OK

Brokerage reply	`{`
	`created_at: " ",`
	`modified_at: " ",`
	`provider_id: "provider id number",`
	`vnf_id: "vnf id number",`
	`price_override: "new price",`
	`status:”status of the new price”,`
	`}`

Table 6-15 API call from SS to the Brokerage to get update on the price

7. VALIDATION

7.1. Functional verification

To test the functionality of the Business Service Catalogue and Service Selection module we have created some sample VNFs and Network Services with real data to make it as close to a real scenario as possible, trying to cover all the functionalities with one example. Table 7-1 collects the results for this verification tests.

#	Functionality	Action	Call from	Request to	Verification	Ok?
1	NSD storing	Store the recently created NSD in the BSC.	Dashboard	BSC	Verify the new NSD is present in the BSC database. Return code CREATED with the body including the autogenerated id after the call for NSD storage	Yes
2	Send the NSD to the Orchestrator	Store the recently created NSD in the BSC.	BSC	Orchestrator	Check the module logs for a code success 200 in this call.	Yes
3	Introduction of the NS SLA template	Store the recently created NSD in the BSC.	BSC	SLA mgmt.	Check the module logs for a code success 200 in this call or the SLA mgmt. database and see the new template us present.	Yes
4	Network services listing	List all available network services	Dashboard	BSC	Click on the NS catalogue and see the complete list on screen.	Yes
5	Filtered NS listing	List all available network services under a certain price	Dashboard	BSC	Set a maximum price, retrieve the filtered list of services and see all the network services in the list are below that price.	Yes
6	NSD modification	Select a NS from the BSC, modify any field	Dashboard	BSC	Verify the NSD has been updated in the BSC database.	Yes
7	NS deletion	Mark a NS for deletion from the BSC	Dashboard	BSC	Verify it has been deleted by retrieving again the complete list of services.	Yes
8	Instantiate a network service	Choose a NS from the BSC, complete the configuration (local network access point) and deploy it	Service Selection	Orchestrator	Check the module logs for a code success 200 in this call.	Yes

9	Update the prices after the negotiation	Choose a NS from the BSC, complete the configuration (local network access point) and deploy it	Service Selection	Brokerage	Check the module logs and see the response from the Brokerage module should be code 200 OK	Yes
10	Register the instantiated NSs and VNFs	Choose a NS from the BSC, complete the configuration (local network access point) and deploy it	Service Selection	Accounting	Check the module logs for a code success 200 in this call or the Accounting database and see an entry for the NS and each of the VNFs has been created.	Yes
11	Terminate a network service	Select a running NS from the instances list and terminate it	BSC	Orchestrator	Check the module logs for a code success 200 in this call.	Yes

Table 7-1 Business Service Catalogue and Service Selection module funtional verification

Further validation tests will be performed in T-NOVA under WP7 (Pilot integration and field trials).

7.2. Requirements fulfilment

Following the successful execution of the aforementioned functional tests Table 7-2 explains how the implemented T-NOVA service framework fulfills the requirements which were set in the specification phase [1].

Req. id	Requirement Name	Requirement Description	Met	Implementation
SC.1	SLAs and pricing description	The service catalogue SHALL be able to store all the available NSs in the T-NOVA marketplace, specifying SLA level and price.	YES	The services are represented using an enhanced ETSI NSD information model that includes, among its many fields, information related to the SLA and price.
SC.2	Catalogue browsable	The service catalogue SHALL be browsable by the SP	YES	Once logged in, a customer can go through the list of available services in the catalogue by means of T-NOVA dashboard. On the other hand, a SP can only see his own services. Since in T-NOVA we have only considered one provider, customer and SP will be able to see the same catalogue.
SC.3	Update of service lists	The service catalogue SHALL support operations of creating/updating /deleting NSs.	YES	The BSC modules exposes a REST API to interact with the published services in the catalogue, including operations such as creating, updating and deleting NSs.

Table 7-2 Business Service Catalog basic requirements

8. CONCLUSIONS

This document reports the outputs of T-NOVA Task 6.1 – Service Description Framework, which aimed to design and implement the necessary components in the service description framework within the T-NOVA Marketplace.

Based on the State of Art survey performed, where two main standardization bodies were considered as the most active for service description in NFV (ETSI NFV and TMForum), and based on the requirements elicitation in the specification phase [1], the following main decisions has been taken to implement T-NOVA Marketplace service framework which have been detailed in this report:

- T-NOVA Marketplace information model comes as a result of applying TMForum SID model to ETSI NFV information model, adding on top of the Network Service Descriptor (NSD) proposed by ETSI for orchestration, a related customer facing Network Service to create a unique NSD to be shared between orchestration and marketplace layers.
- T-NOVA descriptors have been created as an extension of ETSI NSD and VNFD, adding the fields that allow business interaction among the stakeholders that interact in the T-NOVA Marketplace: SLA specification, pricing, etc.
- Two main components has been designed to be part of the T-NOVA Marketplace dealing with service information management:
    - Business Service Catalogue (BSC): this has been introduced following the TMForum recommendations for business agility. The Service Provider will create service offerings for T-NOVA to generate the NSD that will be stored in the BSC, and which will be later browsable by the customer to select.
    - Service selection module (SS): this module has been designed to ease the adaptation of the generic service to the customer network and to send this customization to the Orchestrator to proceed with the service components instantiation.
- T-NOVA Function Store is the repository where the VNFs software images together with their descriptions (VNFD) are store and access by the marketplace. Details of this component can be found in [5].
- All the components (including BSC and SS) in the T-NOVA Marketplace have been developed with a Software Oriented Architecture based on microservices, in which each Marketplace component has been developed separately (UML diagrams in this report) and communicates with the others by means of RESTful APIs (documented in this report). This provides flexibility and scalability to the T-NOVA Marketplace in case further functionalities may want to be added in the future and also this also facilitated the development process by different developers in T-NOVA.
- For the integration of all the different components in the Marketplace Docker has been selected; each microservice relies in a different container, and they are integrated by means of Docker file that coordinates the integration.

It has been included in this report also some basic functional verification tests that have been performed in order to validate the developments and check the

requirements fulfilments. Further validation tests will be done in the project within the specific WP for that purpose.

The T-NOVA Business Service Catalogue and Service Selection module prototypes will be uploaded to http://github.com/T-NOVA.

8.1. Future work

We have identified one item as possible/optional implementation future work in relation to the service composition process. In the current version the service composition process is triggered by the Service Provider (SP) based on acquiring VNFs from different FPs whose characteristics are advertised by means of T-NOVA Function Store through the brokerage process. End-to-end network service requires for some mechanisms for service verification so we rely on this approach for the SP to assure this. However, it could be considered optionally that the customer could be more involved in the service composition process, not only providing service configuration parameters, but also requesting specific standalone VNFs to be part of the final NS, even if the final service verification would be than by the SP.

On the other hand, year 2016 in T-NOVA project will be devoted to the system integration and testing of all its components, e.g. with the T-NOVA Orchestrator and Virtualized Infrastructure Management. We do expect that system integration may detect some gaps or need of fine tuning the interfaces. Moreover, testing the whole T-NOVA system can identify some non-functional aspect that could suggest refining some part of the current work. Furthermore, the complete implementation of T-NOVA Orchestrator, main component interfacing T-NOVA Marketplace, is expected to be finalized by end of March. Therefore, finally integration tests between Marketplace and Orchestrator should be done at that date, and consequently refinements on the Marketplace could be needed, for instance a new slightly different version of the T-NOVA NSD could be evolved. This may result in an updated and final version of this deliverable.

8.1.1. 5G projects

Interesting complementary work to the current work in T-NOVA in relation to service description will be the provision of further automatization tools for service composition and description. For instance in the form of a Service Development Kit that facilitates service programmability as an extension of the service description process within T-NOVA Marketplace through the dashboard. This is in the scope of the SONATA project [22] which aims to design and implement an enhanced NFV orchestrator together with and SDK on top for service programmability. T-NOVA Marketplace can be a very good reference for SONATA to build on, as well as the T-NOVA descriptors.

5GEx project [23] aims to build a sandbox to extend software networks in a multi-domain/operator environment. Although 5GEx is not expected to implement a full marketplace layer, it should specify a northbound API for end users to access the

multi-domain service catalogue. T-NOVA marketplace can be also a good reference to look at for deriving requirements on Business to Customer interface.

8.2. Contributions to standards

A feature proposal about T-NOVA Marketplace and their descriptors was submitted as a potential contribution for ETSI NFV in October 2015 by partners involved in T-NOVA Marketplace. However, we have identified that business aspects fall with more suitability under the activities of TMForum, more concretely TMForum ZOOM project when talking about NFV specificities. This is why T-NOVA is preparing a potential contribution to TMForum in relation to service description aspects to be submitted in the following months.

9. ANNEX A. T-NOVA MARKETPLACE CONTRIBUTION TO ETSI VNFD

In the following specification of the VNF descriptor, the T-NOVA Marketplace additions to the current ETSI model are represented in blue color.

VNFD (base information elements)

Identifier	Type	Cardinality	Explanation
Id	Leaf	1	Numeric ID of this VNFD (automatically created by the FS when uploading the VNFD)
VNF Name	Leaf	1	
Function Provider Id	Leaf	1	Numeric ID of this FP
Function Provider name	Leaf	1	
Description	Leaf		Text description of the VNF
Type	Leaf	1	4 initial types: TC, SA, SBC, HG
Creation date	Leaf	1	
Modification date	Leaf	1	
descriptor_version	Leaf	1	Version of the VNF Descriptor. This version changes whenever there's a new version of XML or JSON document supporting this VNFD, because an error describing the VNF was corrected or any other change, enhancement, etc., is needed in the VNFD (e.g., a flavour not described but already implemented in the VNF is added)
version	Leaf	1	Version of VNF software, described by the descriptor under consideration. This version changes whenever there's a new image or VFNC version, because a bug was corrected (VNF Update, in ETSI) or the FP issues a more featured VNF (VNF Upgrade, in ETSI), e.g., with more scaling-out possibilities.
Vdu	Element	1...N	This describes a set of elements related to a particular VDU, see **section 3**
virtual_link	Element	0...N	Represents the type of network connectivity mandated by the VNF vendor between two or more Connection Points (see **section 4**)
connection_point	Element	1...N	This element describes an external interface exposed by this VNF enabling connection with a VL, see **section 5**. NOTE: The connection between the VNF and the VL is expressed by the VLD referencing this Connection Point. The Connection Point may also be attached to internal Virtual Vinks (vnfd:virtual_link:id).
lifecycle_event	Leaf	0...N	Defines VNF functional scripts/workflows for specific lifecycle events (e.g.. initialization, termination, graceful shutdown, scaling out/in)
dependency	Leaf	0...1	Describe dependencies between VDUs. Defined in terms of source and target VDU, i.e., target VDU "depends on" source

			VDU. In other words sources VDU must exists before target VDU can be initiated/deployed,
monitoring_parameter	Leaf	0...N	Monitoring parameters which can be tracked for this VNF. Some of them will be used for specifying different deployment flavours for the VNF in a VNFD (see section 6). 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-per-second (cps), number-of-subscribers, no-of-rules, flows-per-second etc. One or more of these parameters could be influential in determining the need to scale.
deployment_flavour	Element	1...N	Represents the assurance parameter(s) and its requirement for each deployment flavour of the VNF being described (see **section 6**)
auto_scale_policy	Leaf	0...N	Represents the policy meta data, which may include the criteria parameter and action-type. The criteria parameter should be a supported assurance parameter (vnf:monitoring_parameter). Example of such a descriptor could be Criteria parameter → calls-per-second Action-type → scale-out to a different flavour ID, if exists
Trade	Leaf	1	2 values: off/on

Billing model	Element	1, 2	A set of elements related to a particular billing model (billing period and price)
			2 possible billing models supported: PAYG and RS (see **section 7**)
			JSON example:
			
			"Billing": [
			{
			"model": "PAYG",
			"period": "P1W",
			"price": {
			"unit": "EUR",
			"min_per_period": 5,
			"max_per_period": 10,
			"setup": 0
			}
			},
			{
			"model": "RS",
			"period": NULL,
			"price": {
			"unit": "%",
			"min_per_period": 8,
			"max_per_period": 10,
			"setup": 5
			}
			}
			]

manifest_file	Leaf	0...1	The VNF package may contain a file that lists all files in the package. This can be useful for auditing purposes or for enabling some security features on the package.
manifest_file	Leaf	0...N	The manifest file may be created to contain a digest of each

_security			file that it lists as part of the package. This digest information can form the basis of a security mechanism to ensure the contents of the package meet certain security related properties. If the manifest file contains digests of the files in the package, then the manifest file should also note the particular hash algorithm used to enable suitable verification mechanisms. Examples of suitable hash algorithms include, but are not limited to SHA-256, SHA-384, SHA-512, and SHA-3. In conjunction with an appropriate security signing mechanism, which may include having a security certificate as part of the VNF package, the digest information can be used to help ensure the contents of the VNF package have not been tampered with.

VNFD : deployment_flavour

Identifier	Type	Cardinality	Description
id	Leaf	1	ID of the VNF flavour
assurance parameters	Element	1...N	A set of elements related to a particular monitoring paramenter (see section 6.2). The parameters should be present as a vnfd:monitoring_parameter: - value against this flavour is being described. - Violation - penalty An example is a flavour of a virtual PGW could be described in terms of the parameter "calls per second" There could be a flavour describing what it takes to support a VPGW with 10k calls per second. For each threshold of the monitoring parameter violations and penalties are defined. JSON example: `"Assurance_params": [ { "param-id": "CPU-utilisation", "value" : 60, "unit" "%", "violation": [ { "breaches_count": 5, "interval": 120 } ], "penalty": { "type" : "discount", "value": 30, "unit": "%", "validity": "P1D" } ]`
constraint	Leaf	0...N	Constraint that this deployment flavour can only meet the requirements on certain hardware
constituent_vdu	Element	1...N	Represents the characteristics of a constituent flavour element (see section 6.1) Examples include Control-plane VDU & Data-plane VDU & Load Balancer VDU Each needs a VDU element to support

			the deployment flavour of 10k calls-per-sec of vPGW, Control-plane VDU may specify 3 VMs each with 4 GB vRAM, 2 vCPU, 32 GB virtual storage etc. Data-plane VDU may specify 2 VMs each with 8 GB vRAM, 4 vCPU, 64 GB virtual storage etc.

VNFD : deployment_flavour : constituent_vdu

Identifier	Type	Cardinality	Description
vdu_reference	Reference	1	References a VDU which should be used for this deployment flavour by vnfd:vdu:id
number_of_instances	Leaf	1	Number of VDU instances required
constituent_vnfc	Reference	1...N	References VNFCs which should be used for this deployment flavour by vnfd:vdu:vnfc:id

VNFD : deployment_flavour : assurance_params

Identifier	Type	Cardinality	Description
id	Leaf	1	Id of the parameter, corresponding 1:1 with the ones specified in the monitoring parameters field.
value	Leaf	1	Range of values the metric should take to meet the SLA: LT(50) → lower than 50
unit	Leaf	1	Unit of the values: %, Mb, Gb, etc
formula	Leaf	1	Formula to know whether the current value of the parameter is meeting the SLA.
scalein	Leaf	1	Formula to know whether the current value of the parameter is good enough for scaling in back the function.
violation	element	1..N	Definition of SLA violations
penalty	element	1	Definition of the penalties in case the SLA is not met.

VNFD : deployment_flavour : assurance_params : violation

Identifier	Type	Cardinality	Description
Breaches_count	Leaf	1	Amount of times the thresholds are breached
interval	Leaf	1	Time interval

VNFD : deployment_flavour : assurance_params : penalty

Identifier	Type	Cardinality	Description
type	Leaf	1	Type of penalty ("discount")
value	Leaf	1	Value of the penalty
Unit	Leaf	1	Unit of the value
validity	Leaf	1	Period during which the penalty should be applied

VNFD : billing_model

Identifier	Type	Cardinality	Description
model	Leaf	1	2 available: PAYG and RS (Revenue Sharing)
period	Leaf	1	Definition of the billing period: 3 characters in one string: PNU P: stands for Period N: Number of periods. U: Period unit (H-hours, D-days, W-weeks, M-months, Y-years).
price	Element	1	Price to charge for the specified period

VNFD : billing_model : price

Identifier	Type	Cardinality	Description
unit	Leaf	1	Currency (“EUR”)
setup	Leaf	1	Initial cost for setting up the service
min_per_period	Leaf	1	Minimum charge to apply per period (for auctioning).
max_per_period	Leaf	1	Maximum charge to apply per period (for auctioning).

10. ANNEX B: T-NOVA MARKETPLACE CONTRIBUTION TO ETSI NSD

In the following specification of the NS descriptor, the T-NOVA Marketplace additions to the current ETSI model are represented in blue color.

NSD (base information elements)

Identifier	Type	Cardinality	Description
Id	Leaf	1	ID of this Network Service Descriptor
name	Leaf	1	Name of the service
vendor	Leaf	1	Provider or vendor of the Network Service
version	Leaf	1	Version of the Network Service Descriptor
published	Leaf	1	On/off
Manifest_file_md5	Leaf	1	
vnfds	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.
vnffgds	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.
Vlds	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_dependency	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_parameters	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 SLAs 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-per second (cps), number-of-subscribers, no-of-rules, flows-per second, etc. 1 or more of these parameters could be influential in determining the need to scale-out
service_ flavour	Element	1..N	Service flavour (neworking) – future work
SLA	Element	1…N	A set of elements that define a SLA contract. (it may include service deployment flavor and other possible SLA metrics)
auto_scale_p olicy	Element	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.
pnfds	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.

NSD : connection point

Identifier	Type	Cardinality	Description
Id	Leaf	1	ID of this Connection Point
type	Leaf	1	This may be for example a virtual port, a virtual NIC address, a physical port, a physical NIC address of the endpoint of an IP VPN enabling network connectivity.
Interface	Leaf	1	Control plane, data plane, …

NSD : SLA

Identifier	Type	Cardinality	Description
SLA_Id	Leaf	1…N	ID of this SLA specification
Service flavour networking	Leaf		T-NOVA Future work
Constituent_v nf	Element	1….N	Represents the characteristics of a constituent flavor element. See section 0
Assurance_par ameters	Element	1..N	Assurance parameter or a combination of multiple assurance parameters with a logical relationship between them and values against which this SLA is being described. The parameters should be present as a

			NSD:monitoring_parameter, and they can be: - generic metrics common to all the network services that will be monitored (e.g. network throughput metrics). The basic metrics will have an associated threshold for action initiation. - service specific metrics For each threshold of the assurance parameters violations and penalties are defined.
billing	Element	1	Billing information of the flavour.

NSD : SLA : constituent VNFs

Identifier	Type	Cardinality	Description
Vnf_reference	Leaf	1	Reference to a VNFD declared as VNFD in the network service via vnf:id.
Vnf-flavour_id_reference	Leaf	1	References a VNF flavor (vnfd:deployment_flavour:id) to be used for this service flavour.
Redundancy_model	Element	1….N	Represents the redundancy of instances, for example, "active" or "standby"
Affinity	Leaf	1	Specifies the placement policy between this instance and other instances, if any.
capability	Leaf	1	Represents the capabilities of the VNF instances. An example of capability is instance capacity (e.g. capability = 50% * NS capacity)
Number_of_instances	Leaf	1	Number of VNF instrances satisfying this service assurance. For a gold flavor of the vEPC network service that needs to satisfy an assurance of 96K cps, 2 instances of the vMME VNFs will be required.

NSD : SLA : assurance_parameters

Identifier	Type	Cardinality	Description
name	Leaf	1	Name of the metric
value	Leaf	1	Range of values the metric should take to meet the SLA: LT(50) → lower than 50
unit	Leaf	1	Unit of the values: %, Mb, Gb, etc
formula	Leaf	1	Formula that aggregates the metrics of the VNFs in order to calculate the value above.
violation	element	1..N	Definition of SLA violations
penalty	element	1	Definition of the penalties in case the SLA is not met.

NSD : SLA : assurance_parameters : violation

Identifier	Type	Cardinality	Description
Breaches_count	Leaf	1	Amount of times the thresholds are breached
interval	Leaf	1	Time interval

NSD : SLA : assurance_parameters : penalty

Identifier	Type	Cardinality	Description
type	Leaf	1	Type of penalty (“discount”)
value	Leaf	1	Value of the penalty
Unit	Leaf	1	Unit of the value
validity	Leaf	1	Period during which the penalty should be applied

NSD : SLA : billing

Identifier	Type	Cardinality	Description
type	Leaf	1	Billing model (PAYG)
period	Leaf	1	Definition of the billing period: 3 characters in one string: PNU P: stands for Period N: Number of periods. U: Period unit (H-hours, D-days, W-weeks, M-months, Y-years).
price	Element	1	Price to charge for the specified period

NSD : SLA : billing : price

Identifier	Type	Cardinality	Description
currency	Leaf	1	Currency (“EUR”)
setupCost	Leaf	1	Initial cost for setting up the service
Price_per_perior	Leaf	1	Charge per period

NSD : auto_scale_policy

Identifier	Type	Cardinality	Description
Criteria_parameter	element	1	
Action_type	Leaf	1	

NSD : auto_scale_policy : criteria_parameter

Identifier	Type	Cardinality	Description
parameter	Leaf	1	Monitoring parameter name (“end_to_end_bandwidth”)
threshold	Leaf	1	
unit	Leaf	1	

11. REFERENCES

[1] D2.42 - Specification of Network Function Framework and T-NOVA Marketplace. T-NOVA project.

[2] D6.01 - Interim report on T-NOVA Marketplace implementation. T-NOVA project.

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

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

[5] D5.1 - Function Store. T-NOVA project.

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

[7] 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.

[8] ETSI NFV - Management and Orchestration; Os-Ma-Nfvo reference point – Application and Service Management Interface and Information Model Specification. July 2015.

[9] ETSI NFV - Management and orchestration; Report on NFV Information Model. December 2015.

[10] ETSI GS IFA v14. Network Functions Virtualisation; Management and Orchestration Service Template Specification. December 2015.

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

[12] TM Forum Information Framework Enhancements to Support ZOOM R15.0.1. November 2015.

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

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

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

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

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

[18] D5.31 - Network Functions Implementation and testing. T-NOVA project.

[19] D3.41 - Service Provisioning, Management and Monitoring. T-NOVA project.

[20] D6.2 - Brokerage module. T-NOVA project.

[21] D6.4 - SLAs and billing. T-NOVA project.

[22] SONATA project - Service Programing and Orchestration for Virtualized Software Networks - https://5g-ppp.eu/sonata/.

[23] 5GEx project - 5G Exchange - https://5g-ppp.eu/5gex/.

12. GLOSSARY

Name	Description
Access Control Module	Compoent 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

13. LIST OF ACRONYMS

Acronym	Explanation
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

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