T N O V A

SEVENTH FRAMEWORK
PROGRAMME

NETWORK FUNCTIONS AS-A-SERVICE
OVER VIRTUALISED INFRASTRUCTURES

GRANT AGREEMENT NO.: 619520

Deliverable D6.2

Brokerage Module

Editor	Evangelos K. Markakis (TEIC)
Contributors	Evangelos Markakis, Athina Burdena, George Alexiou, Anargiros Siders, Evangelos Pallis (TEIC), Aurora Ramos, Javier Melián (ATOS), Thomas Pliakas (CLDST),
Version	1.0
Date	April 30th, 2016

Executive Summary

The DoW describes this deliverable as:"*D6.2: Brokerage Module (M24) – Report+Prototype Implementation of the Brokerage Service, including the trading algorithms and the appropriate interfaces.*"

In this context, this document presents an overview of the current technologies related to the development of a brokerage Module. Furthermore, a deep study of the state of the art will be presented focusing specifically to that applicable to T-NOVA brokering. Then the T-NOVA brokering implementation and integration will be explained highlighting the particularities for its implementation. Finally, the definition of the operations that the Brokerake module invokes will be explained.

Table of Contents

Index of Figures

Index of Tables

1. INTRODUCTION

1.1. Objectives and scope

This deliverable presents the current activities of Task 6.2 for the implementation of a Brokerage module inside the T-NOVA Marketplace. The current activities focus on the study, identification and implementation based on the requirements gathered in T-NOVA Deliverable D2.42 and on the study of previous solutions in order to find the most suitable background to build on for implementing the T-NOVA Brokerage Module.

1.2. T-NOVA Marketplace overview

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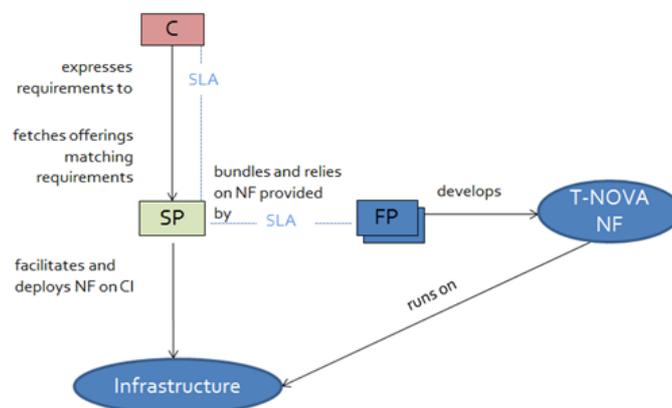
Figure 1-1 Business T-NOVA stakeholders relationships

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

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

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

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

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

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

1.3. Document structure

This report is structured as follows:

Firstly, section 1 provides an overview of the T-NOVA Marketplace and the interaction of the Brokerage module inside the Marketplace.

Section 2 focuses in the study of the state of the art and a specific analysis for the T-NOVA Brokerage Module. For that, a first draft of the available Brokerage platforms is presented as well as the first technologic decisions to implement the Brokerage Modules and its interfaces.

Section 3 is devoted to the brokerage module Implementation. It presents T-NOVA brokering scenario and provides the algorithm behind its implementation. Finally the internal architecture of the brokerage module is addressed.

Section 4 focuses in the Integration of the T-NOVA Brokerage and how it is integrated inside the T-NOVA Marketplace and the Front edge Dashboard.

Section 5 focuses in the initial validation of the Brokerage module and the function and Requirement validation of it.

Finally, section 6 summaries the conclusions of the work presented in this report.

2. BROKERAGE MODULE SoTA

2.1. Introduction

With the expanding introduction of cloud computing, the IT environment is dynamically changed into a lattice of intertwined infrastructure, platform and application services, which are conveyed from different administration suppliers. As the quantity of cloud administration suppliers (or service providers) is increased, as well as the prerequisites of customers become unpredictable, the requirement for on-screen characters to accept a part of intermediation in the middle of suppliers/providers and consumers is getting to be more grounded. Cloud service intermediation is turning out to be progressively perceived, as a key part of cloud computing value chain.

Samples of cloud service intermediation offerings include administrations/services, towards helping to discover and think about cloud services (e.g. marketplaces/stores), to create and customize services (e.g. aPaaS - application Platform as a Service offerings), to coordinate services (e.g. iPaaS - integration Platform as a Service offerings), as well as to manage and monitor services.

Such cloud service intermediation offerings usually vary according to the types of capabilities they offer, or how these abilities are consolidated. However, they have one thing in common. This common ability is the unified characteristic that they make it less demanding, more secure and more gainful for cloud computing adopters to explore, incorporate, consume, extend and keep up cloud services. According to Daryl [1], this is unequivocally the quality suggestion of offerings under the general class of "Cloud Services Brokerage", a term that was authored in 2010 to refer to the emerging part of brokers in the connection of cloud computing.

2.1.1. Different roles of Brokers

Gartner [1] defines "brokerage" as a model of business. This term is used to refer to "the purpose of a business that operates as an intermediary". In more details, Gartner [2] exploits the term to denote "any type of intermediation that adds value to the consumer's use of a service". According to the same analysts, a business cannot be considered a Cloud Service Brokerage, if it does not have a "direct contractual relationship with the consumer(s) of a cloud service".

Moreover, the same analysts define a useful distinction among the terms "brokerage" and "broker", which are often alternatively used. However, they actually refer to different meanings. More specifically, according to Gartner, a broker is "a person, company or a piece of technology that delivers an instance of brokerage or, the specific application of a mechanism that performs the intermediation among consumers and providers". This analysis also states that a Broker delivers value via three primary roles: service aggregation; service integration; and service customization, while additional roles, such as service arbitrage are also possible. These roles of a broker are explained below.

- Aggregation broker: This broker delivers two or more services to consumers and providers. It does not involve any integration or customization of services. Its capabilities are to support large scale cloud provisioning, normalized discovery, access, billing; and to support centralized management, SLAs, and security.
- Integration broker: This broker makes independent services work together for customers. It can allow process integrations, creating new value through integrated results, one-to-many, many-to-one or many-to-many. It is implemented as a PaaS with capabilities, including messaging adapters, orchestrations and translation of event tasks. It can use policies, such as governance policy and API management, shared services for security. This broker allows cloud to cloud integration, such as synchronizing between different applications, or cloud to on-premises integration, like netsuite and quickbooks synchronizing spread sheets.
- Customization broker: This broker can alter or add to the capabilities of a service to improve it. Characteristics include new functionality or new modified service. It uses the original cloud serviced enhanced, one-to-many or many-to-one capabilities to include modifications or combining services, and as a basis for implementation of new services. User interfaces, analytics messages services are typical scenarios of new and composite applications such as reports for salesforce.com. Other examples include price comparisons for bookings, business process services, and configurable processes.

With the definition of broker that Gartner defined, it basically considers any intermediation offering that increases the value of a cloud service to qualify as a cloud service broker. Any supplier of relevant services, even with the most essential intermediation abilities and a "basic" worth recommendation as now qualifies as a service broker. Some different opinions state that this definition is too comprehensive to be in any way valuable. However, Gartner states that it is a vendor-driven market research, instead of a vendor-independent assessor of best practice, and that the views are forcibly shaped by the needs of constituencies that pay for its research: distributors, system integrators, and independent software vendors.

Alternately, in [3], cloud brokers characterizes the term of the Broker as a complex business model that offers a high value commitment in the rising cloud space. Basically, this model influences skills and abilities from every one of the three of the conventional business models; of software, consulting, and infrastructure. In Forrester's view, only integrated or aggregated services, which bring some kind of value out of the composition may qualify as a broker, as well as an intermediary has to provide a certain complex "combined" worth recommendation so as to qualify also as broker. Forrester [4] also distinguishes three types of Cloud Brokers, as indicated by the level of the cloud stack at which they operate:

- simple cloud broker — dynamic sourcing of public IaaS services;
- full infrastructure broker — dynamic sourcing across public, virtual private, and private IaaS;
- SaaS broker — unified provisioning, billing, and contract management with multiple SaaS offerings, potentially including integration of services.

In addition, the work in [5] presents the term of Broker as "an entity that manages the use, performance and delivery of cloud services, while also negotiates relationships among service providers and customers". This work also separates brokers into another three categories, according to their functionality:

- Service Intermediation: A cloud broker enhances a given service, by improving some specific capability and providing value-added services to cloud consumers. The improvement can be managing access to cloud services, identity management, performance reporting, enhanced security, etc.
- Service Aggregation: A cloud broker combines and integrates multiple services into one or more new services. The broker provides data integration and ensures the secure data movement between the cloud consumer and multiple cloud providers.
- Service Arbitrage: Service arbitrage is similar to service aggregation except that the services being aggregated are not fixed. Service arbitrage means a broker has the flexibility to choose services from multiple agencies. The cloud broker, for example, can use a credit-scoring service to measure and select an agency with the best score.

2.1.2. Categorization/classification of Brokerage

The classification comprises two dimensions.

The first dimension concerns the type of **brokerage capability** concerned, and includes discovery, integration, aggregation, customization, quality assurance and optimization.

- **Discovery** deals with the provision of service that helps end users to identify and select the cloud services. For instance, through the use of marketplaces offering listings of cloud services from different providers, direct comparison of similar cloud services, ratings of cloud services, and other relevant features assisting discovery and selection.
- **Integration** is related to the provision of cloud-based software environment in order to integrate separate software systems. The integration aims at either facilitating data exchange between separate systems or realizing collaborative business processes.
- **Aggregation** concerns the provision of a cloud service that is comprised of multiple third-party services. An aggregate cloud service may allow users to interact with the interfaces of the third-party services directly (for instance, through a dashboard-like user interface), or indirectly, through a common interface that encapsulates the individual services and possibly adds common functionality such as authentication, billing, or SLA management across those services.
- **Customization** enables the implementation of new functionality to enrich a cloud service, by means of extension rather than modification of that service's implementation.
- **Quality assurance** is capable to ensure that one or more cloud services obtain specific quality expectations. This can be performed by service testing,

policy enforcement, SLA monitoring, and possibly by self-management mechanisms triggered to restore service quality.

- **Optimisation** enables the opportunistic improvement of the consumption or provisioning of a cloud service with respect to various criteria, such as cost, functionality or performance.

The second dimension is the type of **cloud service** being brokered and includes the Four standard cloud computing service models, i.e. Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Network Function Virtualisation as a Service(NFVaaS).

- **Software as a Service (SaaS)** concerns the provisioning of software application functionality that can be accessed through a web browser or a web API, and can be paid for in a subscription-based or usage-based scheme. The granularity of the service can range from a complete software application for customer relationship management, to a single REST/SOAP web service for obtaining stock market quotes or translating a document.
- **Platform as a Service (PaaS)** concerns the cloud-based provisioning of tools and components for the development, deployment and execution of software applications, ranging from simple utility services to fully fledged development and runtime environments.
- **Infrastructure as a Service (IaaS)** concerns the provisioning of computational resources on demand following a usage-based payment scheme.
- **Network Function Virtualisation as a Service (NFVaaS)** concerns the provision of Network components inside already build infrastructures.
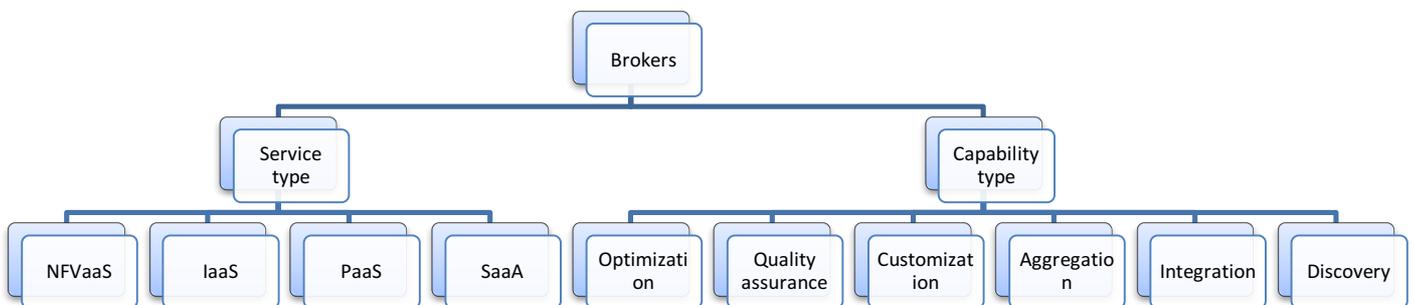
Figure 2-1 Brokers Categorization based on Service/Capability Type

2.1.3. Providers of Brokerage modules

This section presents the brokerage providers that are classified considering the taxonomy presented above. A short description is presented for each provider.

1. Appirio [6] offers enhancements over existing SaaS offerings, such as services for contact and calendar synchronization between Salesforce CRM and Google Mail and/or Google Calendar, file management for Salesforce CRM, and more.

2. Amazon Web Services (AWS) [7] Marketplace offers discovery services for third-party SaaS and PaaS offerings. It is an online store for consumers to identify and select business applications and software infrastructure. It allows to users to find, compare, and immediately deploy a SaaS or PaaS to the Amazon Elastic Compute cloud.

3. Boomi [8] aims at offering services for seamless integration between third-party SaaS offerings. Boomi comprises an integration front-end where someone can build, deploy, and manage their integration processes, and a runtime engine that executes a complete end-to-end integration process.

4. Cloudability [9] offers enhancements for managing existing SaaS, PaaS and IaaS offerings. It enables consumers to track Key Performance Indicators (KPIs) for services, to create customizable cost and usage reports, to receive daily email updates on usage and cost predictions, alerts, and more.

5. CloudKick[10] offers quality assurance services for IaaS offerings. It provides a monitoring dashboard for overseeing various resources across different IaaS providers, with integrated metrics-based data collection and visualization.

6. GetApp [11] offers discovery services for SaaS. It is a free marketplace that helps consumers to identify and evaluate cloud business apps for their needs. Consumers are aided in their search with a recommendation tool, product reviews, comparison tables, and app evaluation resources.

7. Google[12] allows customization of the SaaS offerings in its Google Apps suite of services (GMail, Calendar, Drive, Docs, etc), aggregation of third-party SaaS offerings, as well as discovery of third-party offerings through its Google Apps Marketplace.

8. Heroku[13] is the dominant platform for developing applications in the Ruby programming language. Heroku offers an add-on provider program for third-party Independent Software Vendors (ISVs) to offer services that extend its capabilities.

9. Hojoki[14] offers aggregation services primarily for SaaS, although some IaaS services are also included. The dashboard offered by Hojoki enables consumers to launch their cloud services from within Hojoki, to keep themselves up to date regarding any service updates of interest, and to receive notifications in real-time of any changes in the cloud services they are using.

10. Jitterbit[15] offers integration services for SaaS through a cloud-based data and application integration platform. Jitterbit supports several integration types such as application integration, cloud/SaaS integration, ETL & data integration and business process integration.

11. Kaavo[16] offers aggregation and quality assurance services for IaaS offerings, as well as enhancements for those offerings. It provides a common API and a dashboard for managing resources across IaaS providers, offers performance monitoring, and management of service-level agreements.

12. New Relic [17] offers quality assurance services for SaaS, particularly application performance management for Ruby, PHP, .Net, Java and Python apps. Consumers can leverage New Relic's services to get real-time end user experience monitoring for their apps and visibility across all layers of the app.

13. Rightscale [18] offers discovery, aggregation, quality assurance and optimisation services for IaaS. Through a marketplace, Rightscale allows consumers to identify and select computational resources and virtual server configuration templates, which can be deployed to different IaaS providers through a common API and UI.

14. Salesforce[19] provides not only a SaaS offering, but also a cloud application platform supporting the development of custom applications by third parties, which can be either extensions to the core CRM service by Salesforce, or used independently.

15. SnapLogic[20] offers integration services allowing users to connect any combination of Cloud, SaaS or on-premise applications and data sources. The aim of SnapLogic is to reduce vendor lock-in, providing an open and extensible integration platform for all applications and infrastructure.

16. SpotCloud[21] offers discovery and aggregation services for IaaS that assist with the identification of geographically targeted computational resources at the lowest possible cost, as well as the use of those resources through a common API.

17. StrikeIron[22] offers aggregation services for SaaS, by provisioning a common API for accessing various utility services from third-party SaaS applications. Some of the services StrikeIron offers include email verification, reverse phone lookup, postal address verification, sales tax calculation, geo IP location and more.

18. Tapp[23] offers aggregation and quality assurance services for IaaS offerings, as well as enhancements for those offerings. Tapp introduces a management layer between a virtual infrastructure and the underlying IaaS providers, providing a common API and a dashboard for interacting with the different IaaS providers. In addition, Tapp can monitor the performance of the virtual resources across the IaaS offerings, as well as it can enhance existing IaaS offerings with migration capabilities.

19. Microsoft Azure[24] is an open cloud platform that enables developers to build, deploy and manage applications across a global network of Microsoft-managed data centres. The platform supports development in a range of languages, tools and frameworks. Through Windows Azure Marketplace Microsoft allows third-party SaaS offerings that are hosted on Azure or integrated with Azure to be discovered by Azure users.

20. Equinix Cloud Exchange Platform[27] is a flexible inter-connection solution that provides virtualized, private direct connections that bypass the Internet, in order to provide better security and performance with a range of bandwidth options. It provides direct connectivity to several cloud providers (AWS, Google, Oracle Cloud, Microsoft Azure), and enables buyers and sellers to quickly provision to cloud connections through its portal or programmatically through APIs.

21. CoreSite Open Cloud Exchange[28] provides a portal to establish direct, secure virtual connections to cloud providers and IT service providers. It support a wide area of Cloud Providers (AWS, Microsoft Azure, etc) and several IT providers.

	SaaS	PaaS	IaaS	NFVaaS
Discovery	GetApp, Google Apps, Salesforce, Windows Azure, AWS, Marketplace	Heroku, AWS Marketplace	SpotCloud, Rightscale	T-NOVA
Integration	Boomi, SnapLogic, Jitterbit	SnapLogic		T-NOVA
Aggregation	Hojoki, Google Apps, Salesforce, StrikeIron		SpotCloud, Hojoki, Tapp, Kaavo, Rightscale	
Customisation	Appirio, Cloudability, Google Apps, Salesforce	Cloudability, Heroku	Cloudability, Tapp, Kaavo	T-NOVA
Quality Assurance	New Relic		Tapp, CloudKick, Rightscale, Kaavo, Cloud Exchange Platform, OpenCloud Exchange	T-NOVA
Optimisation			Rightscale	T-NOVA

Table 2-1 Technology Selection

2.1.4. Conclusion of the state of the art

As demonstrated in the table above, there are plenty brokerage modules that offer different combinations of cloud service brokerage capabilities. The majority of the brokerage service providers seem to focus on capabilities for service discovery, integration, aggregation and customization, with a particular emphasis on SaaS services.

Only few brokerage providers enable quality assurance capabilities (New Relic, Tapp, CloudKick, Rightscale, and Kaavo, Cloud Exchange Platform). With the exception of one (New Relic), all of those offerings focus on IaaS, which happens to be the most commoditized category of cloud services today. Coverage of optimization capabilities

is even sparser. Moreover, only one brokerage provider addressing this type of capability (RightScale), which also happens to focus on only one type of cloud service (IaaS).

The T-NOVA Brokerage module is the only brokering platform working with the NFVaaS concept achieving in this way benefits for the SP. More specifically, the T-NOVA SP has the ability to trade among a variety of FP's and receive the best available NFV for his service by taking into accounts the Infrastructure cost and the expected performance (SLA) of the NFV.

3. BROKERAGE MODULE IMPLEMENTATION

3.1. Introduction

In this Section the Brokerage module implementation is detailed.

3.1.1. Architecture of the brokerage module

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

Figure 3-1 depicts the high-level architecture of the T-NOVA Marketplace.

The Brokerage module is in the Bellow center.
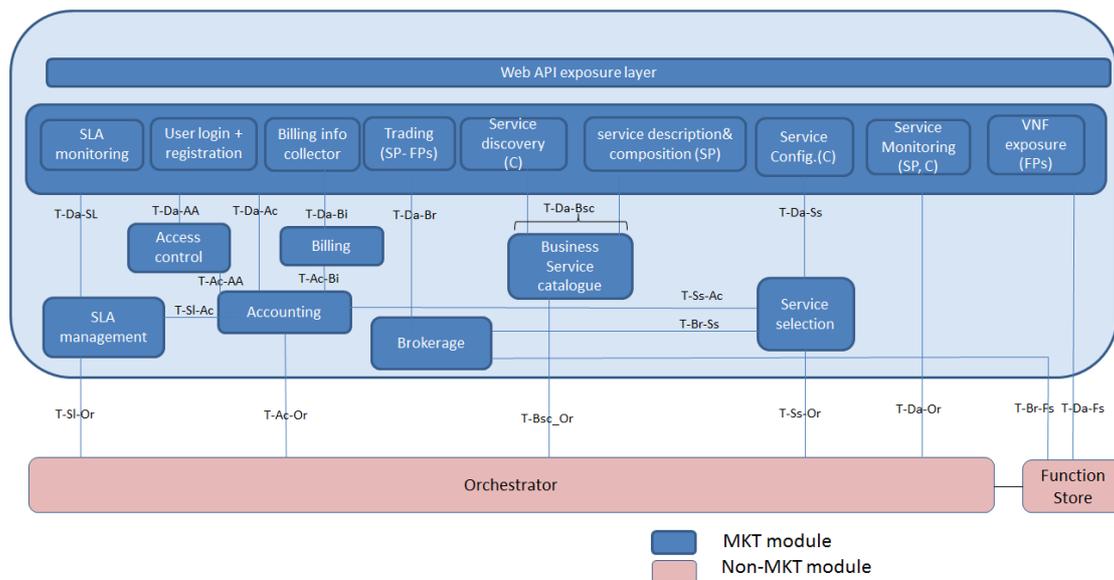
Figure 3-1 Brokerage module and its interfaces in the marketplace architecture

The overall architecture of the brokerage module is depicted in Figure 3-2.

Figure 3-2 Brokerage module internal architecture
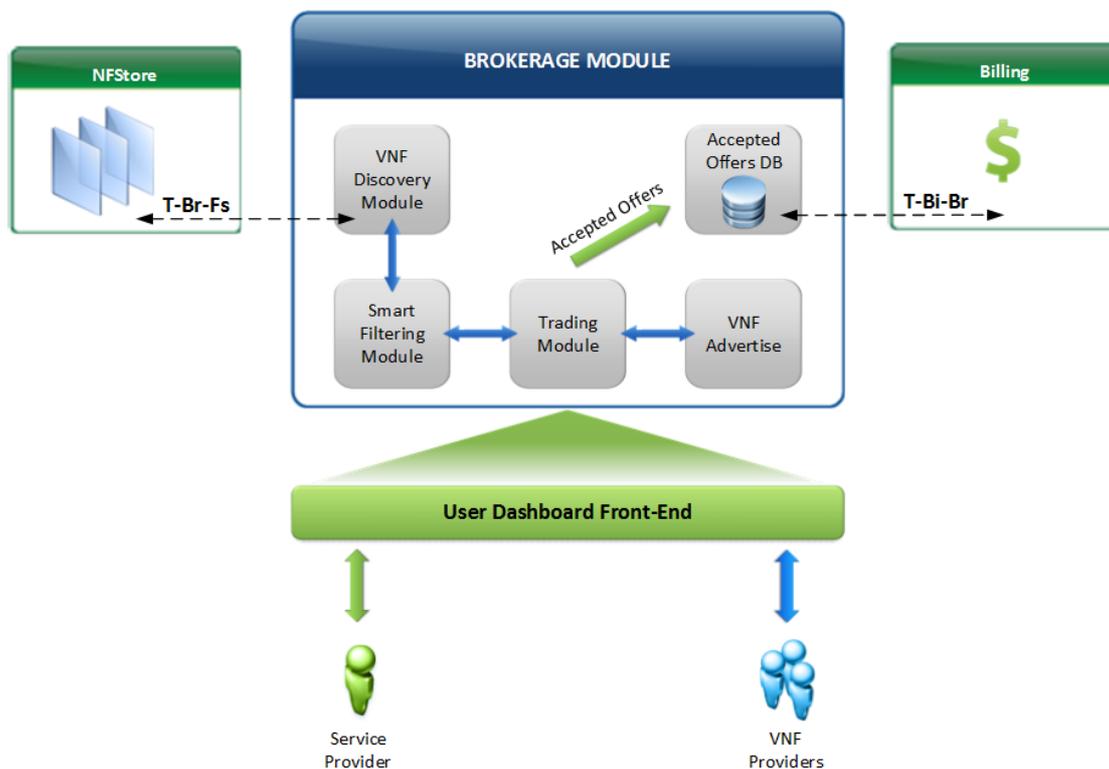
It consists of five main modules that are used in various interactions, as shown in the Figure above:

- **VNF Discovery Module**

   This module is retrieving all the available and tradable VNFs from the NFStore.

- **Smart Filtering Module**

   This module applies a smart filtering and listing to the list of the available VNFs based on the users preferences and the SLA parameters.

- **Trading Module**

   This module is providing all the interaction between the service provider and the function providers. The Trading module is used for requesting a new trade/offer from the SP.

- **VNF Advertise Module**

   This module is responsible to advertise/return all tradable VNFs to the SP's.

- **Accepted Offers DB**

   This module is responsible to store the accepted offers in order to be available for the accounting module when this required for billing purposes.

According to the proposed mechanism, the T-NOVA SP browse the offerings from the Service catalog that match his requirements. If the requested function supports Brokering/Trading the internal modules will try to fulfill the criteria set by the SP. Furthermore, the brokerage initiates the appropriate bid/trading policies according to the T-NOVA SP request inside the trading mechanism in collaboration with the NF Discovery.

The high level architecture of the brokering module along with the interaction inside the Marketplace is depicted in Figure 3-3.
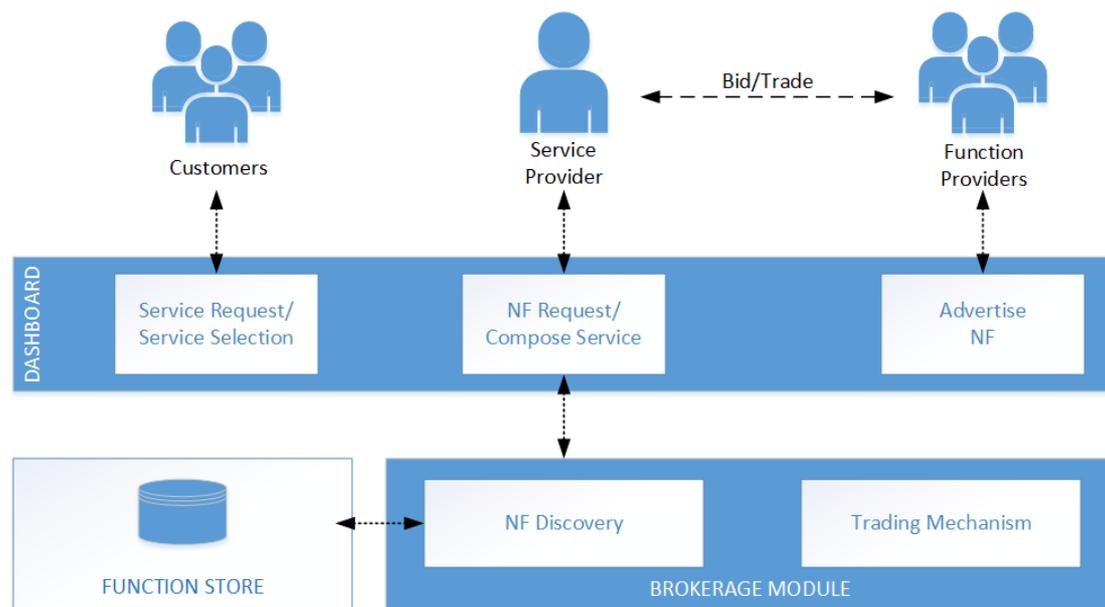
Figure 3-3 Brokerage module internal architecture

In the above figure all interactions are described below:

- The SP provides to the brokerage module the VNF request and the initial price.
- The brokerage module informs the FPs regarding the request and the initial price
- FP sends their bids for the functions (Price, Infrastructure cost, Setup price+ SLA specification)
- The brokerage module solves an auction to maximize its revenue based on the. Price, Setup price, Infrastructure cost and SLA specification
- The brokerage module informs the bid results.
- Depending on the type of auction, an iteration continues until the bid winner is found.
- The brokerage module announces the final results.
- The winner acknowledges the results.
- The brokerage module indicates the VNF's price, which is provided by the FP that won the bidding, to the SP.
- The SP accepts the price.
- The SP receives the VNF.

Finally all Setup Prices and the Price will be stored in the accounting module through the SLA management module.

3.2. Trading Mechanism

According to the trading process (i.e. auction-based algorithm) T-NOVA brokerage module determines the optimal allocation solution, considering the maximization of Service Provider (SP) income. For this, the brokerage module undertakes the trading mechanism that collects bids from Function Providers (FPs), in order to lease the

VNFs to the T-NOVA customers, through the SP. The brokerage module computes the assigning solution through this mechanism together with price and SLA per network service.

In order to calculate an Infrastructure Cost that will be used for the Trading algorithm we have used the calculation of cost based on the following Pseudo algorithm:

```
vdu_cost = 0.03425;

cpu_cost = 0.034;

ram_gb_cost = 0.02125;

storage_gb_cost = 0.0003;

number_of_vdus = 0;

number_of_cores = 0;

number_of_ram_gb = 0;

number_of_storage_gb = 0;

    for Each(vnf.vdu, function (vdu, vdu_key) {

            number_of_vdus += vdu.resource_requirements.vcpus || 0;

            number_of_ram_gb+= vdu.resource_requirements.memory || 0;

            number_of_storage_gb+= vdu.resource_requirements.storage.size || 0;

            number_of_vdus += 1;

 });

    Infrastracture.Cost = (number_of_vdus * vdu_cost) + (number_of_cores *
    cpu_cost) + (number_of_ram_gb * ram_gb_cost) + (number_of_storage_gb *
    storage_gb_cost)

return (fv = Infrastructure.cost);
```

Table 3-1 Infrastructure Cost Calculation

The vdu_cost, cpu_cost, rab_gb_cost and the storage_gb_cost is calculated based on the price stemming from various Cloud Infrastructure providers on the internet [25]

Furthermore, when the auction-based algorithm is followed, the sellers (i.e. FPs) that are denoted as S = {1,2,...,s} lease the VNFs that denoted as V = {1,2,...,v} to b=1 buyers, which is the SP. The SP, is able to buy/lease x_v VNFs for a specific time period t_i, by reporting a price $P(b) = \{x_v, t_i\}$ (i.e. bid price of VNFs considering specific requirements), while the FPs lease y_v VNFs providing a function cost f_v, for a specific time t_i and with a specific SLA L_v, by reporting a price $P(S) = \{f_v, y_v, t_i, L_v\}$ (i.e. asking price of VNFs considering specific requirements). Finally, the pair (b,v) in the pseudo-code of Table 3- 2 represents possible combinations of solutions, regarding "v" VNF to SP. In case that SP benefit has to be maximized, an optimization problem is formulated as follows, based on linear programming, i.e. the following equation:

$$max: \sum_{s=1}^{s}(|P(b) - P(S)|) \quad (1)$$

1: **Inputs: VNFs**, Demand$_{SP}$

2: Access service catalog store

3: Estimate the initial price per VNF

4: Create and advertise price-portfolio

5: Receive FPs offers $P(S)$ and SP bids P(b), where $P(S) = \{f_v, y_v, t_i, L_v\}$ and $P(b) = \{x_v, t_i\}$

6: **for** all offers and *bids* **do**

7:     Sort P(S) and *P(b)* in descending order based on price, function cost and SLA and create the auction-portfolio

8: **end for**

9: Calculate the highest valuation S[b,v] for all VNFs (i,v) $\ni$ {1, 2,..., v}

10: set S$_{optimal}$ = S[b,v] //Random solution for algorithm initiation

11: **for** each bid P(b)**do //**Iteration process in order to find the best solution

12:   **if** (S[b,v]) ≤ (S[b+1, v+1]) // Check if the current solution is better or not to the neighbor solution

13:        **then** save the new solution (S[b+1, v+1]) to the best found

14:   **end if**

15: **end for**

16: **return** Best Solution

Table 3-2 Infrastructure Cost Calculation

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

Furthermore, while the provision of VNFs encompasses several system functionalities, VNF trading can be regarded as one part of the process that deals with the economic aspects. The trading process determines all the issues related with VNFs selling and buying (e.g., direct trading between service provider and function provider or via a brokerage module), while pricing is a major issue that determines the value (or worth) of the VNFs to the service provider and the function provider.

Another issue is the competition/cooperation among function and service providers, as well as customers involved in VNF trading. Depending on the VNF trading model, the VNF access may require permission through the cooperation of service provider and function provider, through a payment process. To determine the optimal network function provision during the trading process, optimization and decision theory techniques can be used.

3.3. Implementation and Technology Selection

The Broker component is implemented with Flask.

Flask is a lightweight python micro-framework with small core and easily extensible. This framework was chosen as an ideal framework for the broker micro service. The main aspect taken into account in order to select Flask was that the Brokerage Module is a small application with simpler requirements unlike the other popular python web frameworks like Pyramid, Django etc. that both aimed at larger and more complex applications.

Furthermore, by using Flask there is no need to follow the Model View Controller (MVC) or the Model Template View (MTV) model that other web frameworks need. Flask can be a perfect backend for that using a robust, tried and lightweight restful API. Additionally Flask provides good documentation which is a straightforward help for using Flask.

The Database that the Brokerage uses is MongoDB which is a NoSQL (non-relational, next-generation operational data store and databases) database that creates a schema-less database providing huge advantages compared to Relational Database Management System (RDMS) databases. The main advantages of MongoDB are the following:

- Flexibility: Easy to Implement
- Performance: Fast response time
- Scalability: Easy to extend the brokering plarform.

Finally the Brokerage as a part of the T-NOVA Marketplace is inserted in a container based on Docker implementation[26].

The Docker image is build following the Dockerfile description depicted in Figure 3-4 which is a text document that contains all the commands a user could call on the command line to assemble an image. The docker builds the Brokerage image automatically by reading the instructions from this Dockerfile.
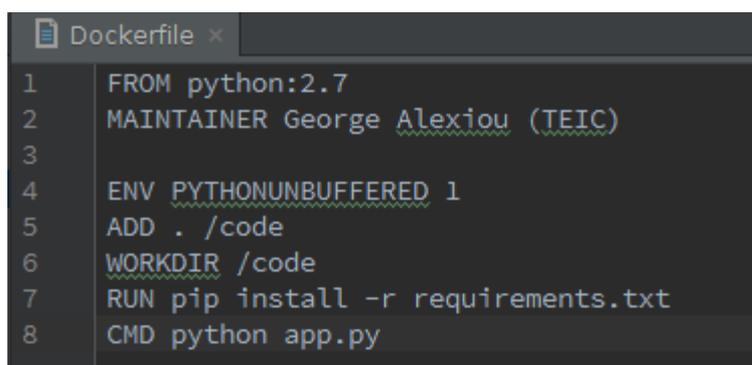

```
Dockerfile ×
1    FROM python:2.7
2    MAINTAINER George Alexiou (TEIC)
3
4    ENV PYTHONUNBUFFERED 1
5    ADD . /code
6    WORKDIR /code
7    RUN pip install -r requirements.txt
8    CMD python app.py
```

Figure 3-4 Broker Dockerfile

4. BROKERAGE MODULE INTEGRATION

4.1. Introduction

The description of the integration process is presented in this section together with dashboard overview of the Brokerage.

4.2. Summary of Interfaces

The required interfaces of the brokerage module for the proper communication with the other parts of the T-NOVA system are:

- **Interface to the dashboard**: this interface is required in order for the users of T-NOVA system (i.e. SP, FPs) to be allowed to trade. For this purpose, functionalities such as service composition/VNF request by the SP and advertise VNF/trading by the FPs are exploited.
- **Interface to Service Selection module**: This interface is used by the Service selection module to consult any change in the information regarding the price as a result of the trading process in the brokerage module before creating the accounting entry.
- **Interface to the Function Store**: this interface is required in order for the brokerage module to retrieve information about the available VNFs for a service composition.

4.3. APIs Definition

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

4.3.1. Dashboard-to-Broker

The communication API between the Dashboard and the Brokerage provides the necessary functionality to the SP and FP. These operations are invoked by the necessary interface (T-Da-Br):

4.3.1.1. Get Available VNFs

In order to get the available VNFs the dashboard can request the available VNFs through the following call:

GET /broker/vnfs/?<filters>
Response Body:

```
[
{
provider: "TEIC",
release: "T-NOVA",
type: "FW",
id: 562,
description: "PFSense is a firewall..."
...
},{
provider: "TEIC",
release: "T-NOVA",
type: "FW",
id: 563,
description: "UNTagle is a firewall..."
...
}
]
```

4.3.1.2. New Trade Request

The request of a new trade is initiated by the SP by providing the FP Provider ID, the VNF id, and the price that the Service provider is willing to pay.

In the example bellow we see a SP requesting a new price for a VNF.

POST /broker/vnfs/trade/

Request Body:
{
provider_id: 6,
vnf_id: 563,
price_override: 12.0
}

4.3.1.3. Get Trade Request

After posting the trade request the FP receives a trade request for his VNF. The status of the VNF is pending showing that a reaction is needed from the FP.

The example bellow show the Pending status of a Trade request

GET /broker/vnfs/trade/4

Response Body:
 	
{
 	
created_at: %Y-%m-%dT%H:%M:%SZ,
 	
modified_at: %Y-%m-%dT%H:%M:%SZ,
 	
provider_id: 6,
 	
vnf_id: 563,
 	
price_override: 12.0,
 	
status:"pending",
 	
id:4
 	
}

In the example bellow we can see that multiple request can be done to a multiple FPs

GET /broker/vnfs/trade/

Response Body:
 	
[
 	
{
 	
created_at: %Y-%m-%dT%H:%M:%SZ,
 	
modified_at: %Y-%m-%dT%H:%M:%SZ,
 	
provider_id: 6,
 	
vnf_id: 563,
 	
price_override: 12.0,
 	
status:"pending",
 	
id:4
 	
}
 	
{
 	
created_at: %Y-%m-%dT%H:%M:%SZ,
 	
modified_at: %Y-%m-%dT%H:%M:%SZ,
 	
provider_id: 6,
 	
vnf_id: 564,
 	
price_override: 6.0,
 	
status:"accepted",
 	
id:5

```
}

]
```

```python
280    @app.route('/broker/vnfs/trade/<int:trade_id>/accept/', methods=['GET'])
281    @jwt_required()
282    def trade_accept(trade_id, vnfd_id):
283
284        if has_perm('broker.trade_vnfs'):
285
286            mongo.db.trades.update_one({
287                'id': trade_id,
288                'vnfd_id': vnfd_id,
289                'provider_id': request.user['user_id']
290            }, {
291                '$set': {
292                    'status': 'accepted'
293                }
294            }, upsert=False)
295
296            return Response('', mimetype='application/json', status=200)
297
298        else:
299            response = jsonify(detail="You do not have permission to perform this action.")
300            response.status_code = 401
301            return response
```

Broker, Accept Trading offer – Code Snippet

4.3.1.4. Accept/Reject Trade Requests

The acceptance or rejection of a trade request is initialized by the FP and the ID of vnf is provided to the Brokering Module.

An acceptance of a VNF.

GET /broker/vnfs/trade/5/accept/

A rejections of a trade request offer

GET /broker/vnfs/trade/5/reject/

```
333     @app.route('/broker/vnfs/trade/<int:trade_id>/reject/', methods=['GET'])
334     @jwt_required()
335   def trade_accept(trade_id, vnfd_id):
336
337         if has_perm('broker.trade_vnfs'):
338
339             mongo.db.trades.update_one({
340                 'id': trade_id,
341                 'vnfd_id': vnfd_id,
342                 'provider_id': request.user['user_id']
343             }, {
344                 '$set': {
345                     'status': 'rejected'
346                 }
347             }, upsert=False)
348
349             return Response('', mimetype='application/json', status=200)
350
351         else:
352             response = jsonify(detail="You do not have permission to perform this action.")
353             response.status_code = 401
354             return response
355
```

Broker, Reject Trading offer – Code Snippet

4.3.2. Billing-to-Broker

Upon acceptance of a trade the new prices must be provided to the T-NOVA Billing module in order this to be done we use an interface between the Brokering module and the Service Selection module providing the following calls

4.3.2.1. Get Trade Request

The acceptance of a trade initiate a request for price override and this is done by providing the Provider ID the VNF id and when this approve of price is done.

GET /internal/broker/trade/?id=5&vnf_id=564
Response Body: { created_at: %Y-%m-%dT%H:%M:%SZ, modified_at: %Y-%m-%dT%H:%M:%SZ, provider_id: 6, vnf_id: 564, price_override: 6.0, status:"accepted", id:5

```
}
```

4.3.3. Function Store-to-Broker

The Function store is an intermediate entity between the Marketplace and the Orchestrator and upon request it is able to provide a variety of stored VNF to the Brokering module for processing them and initiate the Auction/trade. The interface used is T-Br-Fs API (**GET all VNFs)**

The following method is used in order to return all description of the VNFs and in order in second time to process them and take a decision based on the trading mechanism.

```
GET /vnfs/
```

Response Body:

```
[

{

provider: "TEIC",
release: "T-NOVA",
type: "FW",
id: 562,
description: "PFSense is a firewall…"
…
},{

provider: "TEIC",
release: "T-NOVA",
type: "FW",
id: 563,
description: "UNTagle is a firewall…"
…
}
]
```

```
316    @app.route('/broker/vnfs/', methods=['GET'])
317    @jwt_required()
318    def vnfs():
319        if has_perm('broker.view_all_vnfs'):
320
321            code, nfs_response = nfsapi.get_vnfds()
322            print code, nfs_response
323            if code != 200:
324                response = jsonify(detail="NFS POST request failed, code:%s" % code, nfs_error=nfs_response)
325                response.status_code = 404
326                return response
327
328            return Response(nfs_response, mimetype='application/json', status=200)
329
330        else:
331            response = jsonify(detail="You do not have permission to perform this action.")
332            response.status_code = 401
333            return response
334
```

T-Br-Fs Retreives all VNFs from the NFStore – Code Snippet

4.4. Dashboard Integration

4.4.1. VNF Trading

In Figure 4-1 the initial screen were the SP is able to select among the available VNFs and initiate a trade request between him and the FPs.

Figure 4-1 VNF Trading

4.4.2. Trade Request

In Figure 4-2 we see the pop up that is used in order to simplify the procedure of the Brokering. In the new windows we can set new price and or Setup price. In Figure 4-2 we can see a request for a New Price for the VNF.
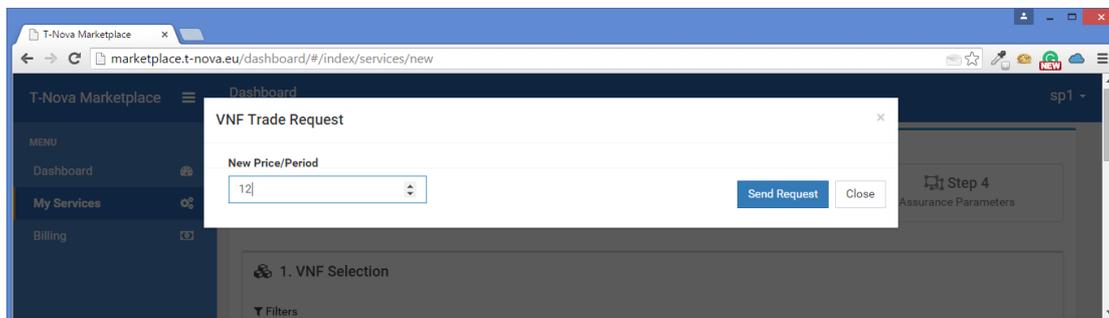
Figure 4-2 Trade Request

4.4.3. Pending Trade Request

In Figure 4-3 we can see that the SP has provided the new price and Setup Price and the status of the VNF has changed to Pending. The pending view provides the necessary time in order the Brokering module to process the request and upon the reaction of the FP to provide the necessary answer to the SP.
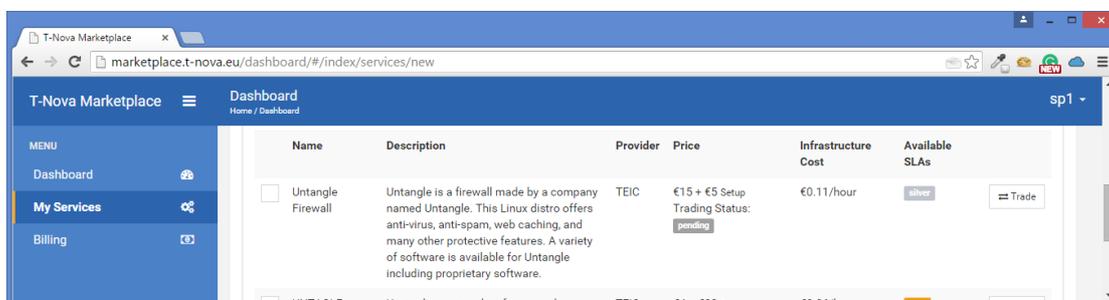
Figure 4-3 Pending Trade Request

4.4.4. Request FP View

In Figure 4-4 the multiple offers from the Brokering module to the FP are depicted with the answers provided. Furthermore we can see a pending Auction request were the FP is able to accept or reject.
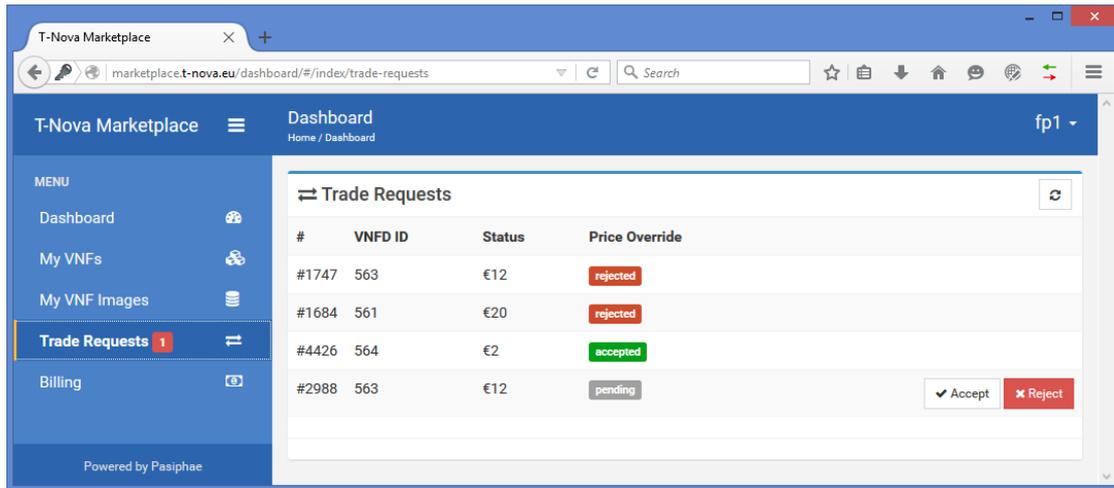
Figure 4-4 Request FP View

4.4.5. Accepted Trade Offer

In Figure 4-5 the acceptance of new price is depicted. The colors Green for Accept or red for reject provide a quick view to the SP if multiple offers are done.

Figure 4-5 Accepted Trade Offer

5. VALIDATION

5.1. Functional verification

To test the functionality of the Brokering 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 5-1 collects the results for this verification tests.

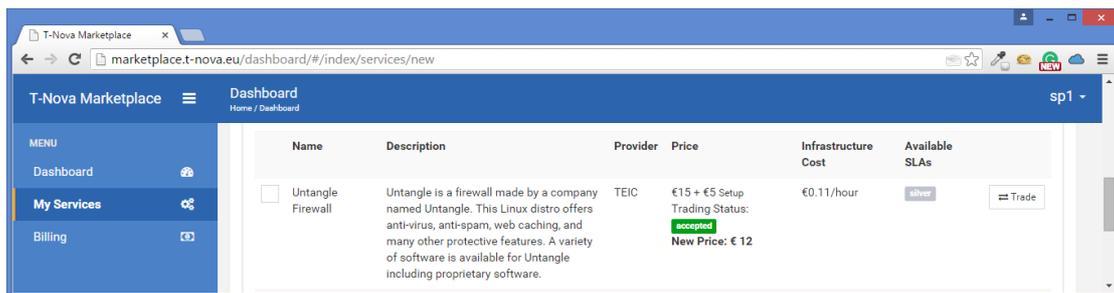| #	Functionality	Action	Call from	Request to	Verification	Ok?
9	Update the prices after the negotiation	Choose a NS from the BSC, complete the configuration (local network access point) and deploy it	Service Selection	Brokerage	Check the module logs and see the response from the Brokerage module should be code 200 OK	Yes

Table 5-1 Brokerage Module functional verification

5.2. Requirements fulfillment

The requirements for the T-NOVA Brokerage module gathered in the specification task are related to the need to provide a brokering mechanism that can support Services and Functions while on the same time to optimize the decision. In the following table the gathered requirement from Deliverable D6.01 are presented while their implementation status is provided in the columns Implementation status and justification.

Req. id	Requirement Name	Requirement Description	Justification of Requirement	Implementation Status	Implementation Justification
B.1	Trading/Bidding	The Brokerage SHALL be able to perform auctions among Service provider and Function providers	The Brokerage must be able to initiate auctions whenever it is required.	YES	The Auction is initiated always by the SP in the Setup Service Step.
B.2	Trading/Bidding	The Brokerage SHOULD take into account the SLA parameters	The Brokerage must be able to initiate auctions by using the SLA parameters as a prerequisite	YES	Compare table presenting the main difference on SLA realisation

B.3	Trading/Bidding	The Brokerage SHOULD return the best possible Function to the SP	The brokerage must be able to auction and return the best possible results to the SP	YES	The results of Trading are sorted based on the Best possible Function based on the Cost, Setup Cost Infrastructure Cost and SLA parameters.
B.4	Trading/Bidding	The Brokerage SHOULD be able to accommodate multiple FP to a SP	The brokerage must be able to auction among the available FPs	YES	Selection of Multiple VNF's from Multiple FPs is available
B.5	Trading/Bidding	The Brokerage SHOULD be able to take into consideration predefined Atomic Trading Units	The brokerage must be able to auction among the available Functions with predefined Parameters	YES	The Atomic Trading Units are defined as a MUST provide parameters inside each VNF.

Table 5-2 Brokerage Requirements

6. CONCLUSIONS

This document has introduced the work done in of the implementation of T-NOVA brokerage module and its interactions with other parts of T-NOVA system: Function Store, dashboard and the service selection module. Next the requirements gathered in previous work have been reviewed, amended if necessary, and consolidated. Then, a deep study of the state of the art has been explained focusing specifically to that applicable to T-NOVA brokering. Then the T-NOVA brokering implementation and integration has been explained highlighting the particularities for its implementation. In this document, the definition of the operations that each REST API will have to support has been also included.

7. REFERENCE

[1]   Plummer, D., B. Lheureux, M. Cantara and T. Bova. "Cloud Services Brokerage Is Dominated by Three Primary Roles". Gartner Research Note G00226509 (2011).

[2]   Plummer, D., B. Lheureux, and Frances Karamouzis. "Defining cloud services brokerage: taking intermediation to the next level." Gartner Research Note G00206187 (2010).

[3]   Cloud Broker — A New Business Model Paradigm by Stefan Ried. Published: August 10, 2011, Updated: September 22, 2011.

[4]   Cloud Brokers Will Reshape The Cloud - Getting Ready For The Future Cloud Business Models.Forrester, Sep. 2012.

[5]   NIST, Cloud Computing Reference Architecture, National Institute of Standards and Technology, Special Publication 500-292, September 2011. Available on: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505

[6]   http://appirio.com/

[7]   https://aws.amazon.com/

[8]   http://www.boomi.com/

[9]   https://www.cloudability.com/

[10]   https://www.crunchbase.com/organization/cloudkick

[11]   https://www.getapp.com/

[12]   http://google.com

[13]   https://www.heroku.com/

[14]   http://hojoki.com

[15]   http://www.jitterbit.com/

[16]   http://www.kaavo.com/

[17]   http://newrelic.com/

[18]   http://www.rightscale.com/

[19]   http://www.salesforce.com/eu/

[20]   http://www.snaplogic.com/

[21]   http://www.spotcloud.com/

[22]   https://www.strikeiron.com/

[23]   https://www.flexiant.com/tapp/

[24]   https://azure.microsoft.com/en-us/]

[25]   https://www.arubacloud.com/cloud-computing/configure-your-cloud.aspx

[26]   https://www.docker.com/

[27]   Equinix Cloud Exchange Platform - http://www.equinix.com/services/interconnection-connectivity/cloud-exchange/

[28]   CoreSite OpenCloud Exchange - http://www.coresite.com/solutions/interconnection/open-cloud-exchange

GLOSSARY

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

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

8. LIST OF ACRONYMS

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

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