Grant Agreement No.: 604590
Instrument: Large scale integrating project (IP)
Call Identifier: FP7-2012-ICT-FI

eXperimental Infrastructures for the Future Internet

D3.1: XIFI infrastructure adaptation components API open specification

Work package	WP3
Task	Tasks T3.1, T3.2, T3.3
Due date	30/10/2013
Submission date	30/10/2013
Deliverable lead	Universidad Politecnica de Madrid (UPM)
Authors	Federico Alvarez (UPM), Jose Gonzalez (UPM), Luis Contreras (TID), Oscar Gonzalez (TID), Andreas Papadakis (SYN), Panos Trakadas (SYN), Adel Al-Hezmi (FhG), Bernd Bochow (FhG), Giuseppe Cossu (CNET), Silvio Cretti (CNET), Federico Facca (CNET), Carlo Cinato (TI), Luca Fantolino (TI), Jose I. Aznar (i2CAT), Eduard Escalona (i2CAT), Eamonn Power (TSSG), Richard Hughes-Jones (DANTE), Matthias Baumgart (DT), Ivan Biasi (TN), Dave Wilson (HEAnet)
Reviewers	Silvio Cretti (CNET), Federico Facca (CNET), Thierry Milin (FT), Bernd Bochow (FhG)
Abstract	This document contains the specification of the infrastructure

	adaptation components, which are going to be developed in WP3, in the form of APIs open specifications. The adaptation components consist of three kinds of different adapters: for networking, monitoring and GE management and configuration respectively. They allow the infrastructures to offer the full capacity of XIFI in a unified manner, and to achieve the necessary degree of compatibility with FI-WARE and with the rest of infrastructures of the XIFI pan-European federation ensuring the highest possible quality to the end-users for experimentation.
Keywords	Future Internet, Adapter, API, Infrastructure, Monitoring, Network, Datacenter, FI-WARE, Generic Enabler, Management, Federation

Document Revision History

Version	Date	Description of change	List of contributor(s)
V1.0	20.09. 2013	Version ready for internal review	Federico Alvarez (UPM), Jose Gonzalez (UPM) et al.
V1.1	30.09.2013	Final revision	Federico Alvarez (UPM), Jose Gonzalez (UPM) et al.
Final	8.10.2013	Final version	Maurizio Cecchi (TI)

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Nature of the deliverable:	R	
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to bodies determined by the XIFI project	
CO	Confidential to XIFI project and Commission Services	

*R: report, P: prototype, D: demonstrator, O: other

EXECUTIVE SUMMARY

This document contains the specification of the infrastructure adaptation components, which are going to be addressed within the XIFI project, in the form of APIs open specifications. The adaptation components consist of an abstraction layer which allows the infrastructures to offer the full capacity of XIFI in a unified manner, and to achieve the necessary degree of compatibility with FI-WARE and with the rest of infrastructures of the XIFI pan-European federation, ensuring the highest possible quality to the end-users.

The specifications will cover three different kinds of adapters which XIFI designed to the former defined objectives.

- Infrastructure Adaptation mechanisms to provide unified control and access to network components of the infrastructure. The mechanism will support the monitoring and control of quality of network services interconnecting the federated infrastructures. The goal will be the availability of a network operational model and a set of adaptation enablers to sustain the seamless interaction with the network substrate of the federation platform created by XIFI.
- Infrastructure Adaptation components for the monitoring of the infrastructures at different levels to expose and control the relevant parameters of the infrastructures to assure the fulfilment of the required level of interoperability with the enablers and external services and applications,
- Infrastructure Adaptation managers to provide enablers management to assure a correct access to the enablers' resources in the different infrastructures and a common seamless access to the functionality irrespective of the enabler infrastructure location.

This document is acting as the base for a subsequent development of the components resulting from development activities in the correspondent task force after the first stage of the project. But not only is the utility of the document for XIFI developers since it provides the definitions and adaptation framework, including the single APIs specifications, to allow the understanding of the concepts and architecture for the integration of these components with the current XIFI infrastructures. In addition, this deliverable is prepared to be a solid base for future joining infrastructures and support the integration to make them fully compatible with the XIFI federation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application Programming Interface
BoD	Bandwidth on Demand
DC	Datacenter
DCA	Deployment and Configuration Adapter
DEM	Datacenter and Enablers Monitoring
E2E	End to End
FI-PPP	Future Internet - Private Public Partnership
FMC	Fundamental Modelling Concepts
GE	Generic Enabler
IaaS	Infrastructure as a Service
IoT	Internet of Things
NaaS	Network as a Service
NAM	Network Active Monitoring
NPM	Network Passive Monitoring
NREN	National Research and Education Network
PaaS	Platform as a Service
PCE	Path Computation Element
QoS	Quality of Service
SaaS	Software as a Service
SDN	Software Defined Networking
SE	Specific Enabler
SNMP	Simple Network Management Protocol
UC	Use Case
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
XIMM	XIFI Infrastructure Monitoring Middleware

1 INTRODUCTION

The XIFI pan-European federation aims to be the community cloud for European FI-PPP developers enabled by advanced Future Internet (FI) infrastructures in Europe. The FI-PPP [25] is an ambitious programme by the European Commission part of the 7th Framework Programme aiming at exploring the potential of a common platform for Future Internet technologies to establish new business ecosystems. XIFI, through this community cloud, will provide a market place to access: i) the web-based services offered by FI-PPP (i.e. the Generic Enablers developed by FI-WARE [22] and the Specific Enablers provided by Use Case Trials), ii) advanced Future Internet infrastructures that provide capacities, and iii) data to empower the applications developed by early adopters of FI-PPP technologies.

This document will describe the abstraction layer to be adopted by a specific FI infrastructure in order to become part of the federation proposed by XIFI. It is mandatory to define those adaptation components which allow an infrastructure to offer the full capacity of XIFI in a unified manner, and to achieve the necessary degree of compatibility with FI-WARE and with the rest of participating infrastructures, ensuring the highest possible quality to the end-users. This assessment must conclude in the form of APIs open specifications.

The specifications will cover three different kinds of adapters which XIFI designs to the former defined objectives.

- Infrastructure Adaptation mechanisms to provide unified control and access to network components of the infrastructure. The mechanisms will support the monitoring and control of quality of network services interconnecting the federated infrastructures. The goal will be the availability of a network operational model and a set of adaptation enablers to sustain the seamless interaction with the network substrate of the federation platform created by XIFI.
- Infrastructure Adaptation components for the monitoring of the infrastructures at different levels to expose and control the relevant parameters of the infrastructures to assure the fulfilment of the required level of interoperability with the enablers and external services and applications,
- Infrastructure Adaptation managers to provide enablers management to assure a correct access to the enablers' resources in the different infrastructures and a common seamless access to the functionality irrespective of the enabler infrastructure location.

This document is acting as the base for a subsequent development of the components resulting from development activities in the correspondent task force after the first stage of the project. But not only is the utility of the document for XIFI developers since it provides the definitions and adaptation framework, including the single APIs specifications, to allow the understanding of the concepts and architecture for the integration of these components with the current XIFI infrastructures. In addition, this deliverable is prepared to be a solid base for future joining infrastructures and support the integration to make them fully compatible with the XIFI federation.

The following section will introduce a map of XIFI system and services that allows the readers to orientate across the different building blocks of XIFI (Figure 1). The same high-level picture will be available in each XIFI technical-related deliverable in order to provide a common overview to the audience. Furthermore, it is planned to adopt the same principle to guide external stakeholders across the different public documents that will be exposed in the XIFI Wiki [67]. This collaborative wiki is a live entity and represents the “working” documentation where all the latest evolutions of requirements, derived scenarios and architectures are available.

1.1 A visual map of XIFI main building blocks (you are here!)

XIFI offers a marketplace to European large-scale trial developers to access FI-PPP technologies and Future Internet infrastructures. The marketplace provides access to Generic Enablers developed in FI-WARE (the horizontal platform), and potentially (to be checked when they will be provided) to the Specific Enablers developed by Use Case projects (the vertical platforms), through a highly available and reliable "federation" of infrastructures. To complement and support the above-mentioned technologies (GEs and SEs), XIFI leverages on FI infrastructures with different characteristics (e.g. in terms of location, user community, quality of service, special hardware).

XIFI aims at illustrating the potential of such marketplace through different showcases that will act as demonstrators of XIFI offer. For example, one of our showcases will illustrate how developers can take advantage of XIFI multi-site infrastructure to build distributed applications with high-availability set-up and QoS controlled across the different used sites.

XIFI offer comprises two main parts: **the platform**, i.e. the "virtual" market place that allows end-users to browse through, configure and access enablers and infrastructures in preparation for their experimentations and **the operational services**, i.e. the set of activities that go around the platform to provide a comprehensive "package" to XIFI end-users. The operational services and the platform go beyond pure technical considerations: they offer a summary vision of technical and business aspects that constitute the XIFI offering.

The **XIFI platform** is conceived with the context of the community cloud deployment model, and offers all the three traditional services models of cloud platforms: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [63]. The XIFI platform is composed of different elements, such as:

- **User Interfaces** comprising tools for browsing, discovering, recommending, configuring, allocating and deploying resources; such interfaces provides as well means to interact with operational services provided by infrastructure owners, such as developers support and SLA management;
- **Federated Cloud and Service Management** for the aggregation of different resources available through the federation, the shared security and identity management across the federation, the software automation for the installation of new nodes and new services on top of nodes;
- **Dynamic Network Management** to support connectivity configuration across federation nodes at the level of single services, fulfilling changing user demand;
- **Resource Monitoring** that supports the active and passive collection of data from physical and virtual sources, providing the capacity to gain access to meaningful information on infrastructure and service availability;

The **XIFI operational services** are a set of fundamental services to support the management of the XIFI platform as well as address the needs of different actors, i.e. infrastructure owners and application developers. Most of these services are governed by protocols and procedures to ensure the operational continuity of XIFI platform. Such operational activities cover management of the nodes – in particular Level 1 and Level 2 support to developers–, support to infrastructure owners in deployment and maintenance of the platform (with special focus on new infrastructures joining XIFI during its second year) and training for developers and infrastructures owners (it will be built exploiting the showcases and referring to the documentation available from the technical activities over the project).

XIFI building blocks are showed in Figure 1. In this deliverable the authors will introduce general aspects related to the XIFI Platform and XIFI Services.

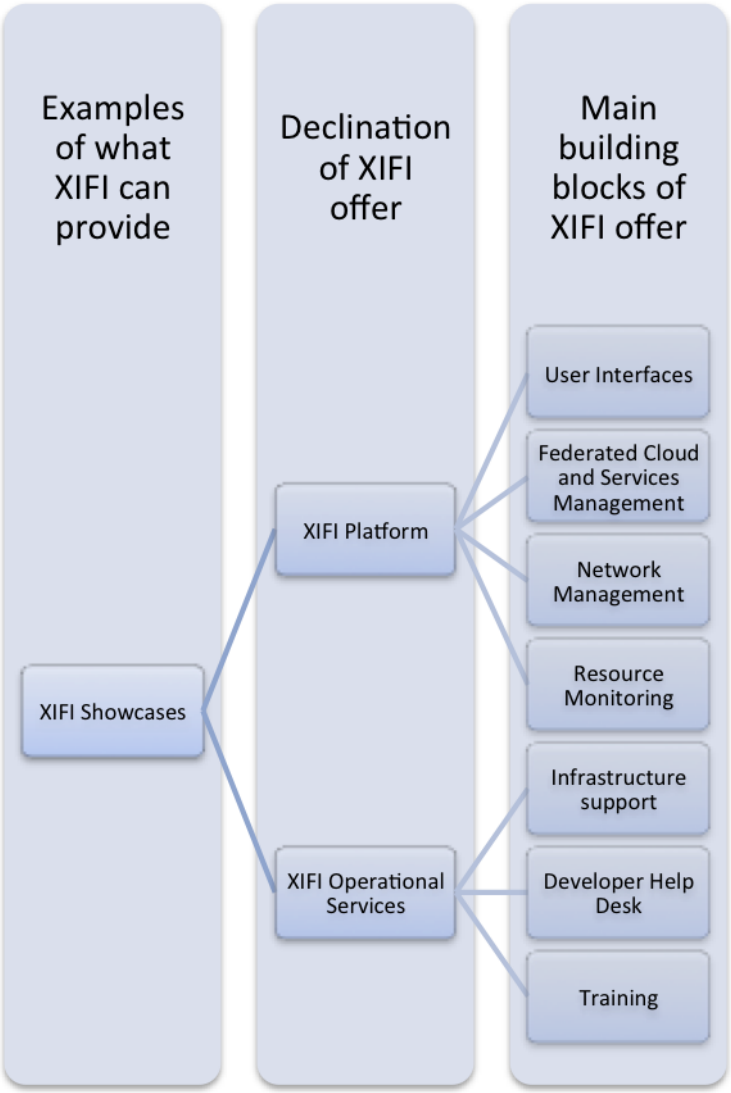


Figure 1: Visual Map of XIFI main building blocks

This deliverable focuses on the definition of architectural models and APIs specifications of the adaptation components within "Network Management" and "Resource Monitoring" main blocks.

1.2 Scope

This deliverable focuses on the definition of the tools and APIs required for supporting infrastructure connection to XIFI federation services. In particular, this deliverable aims to describe: a) the solutions to be adopted to interconnect the different XIFI nodes and manage network services across infrastructure domain boundaries; b) the architecture to gather "physical appliance" monitoring data from the infrastructures in a consistent way with monitoring solution developed by FI-WARE; c) the

procedure to ensure that participating infrastructures can easily adopt specific XIFI and FI-WARE tools.

1.3 Introduction to XIFI Adapters

Since an ever-growing set of multi-domain infrastructures aims to work in a collaborative manner, the existence of an abstraction level results essential to provide a common overview to the end users. This standardization layer will be provided by the XIFI Adapters, software components in charge of supporting the integration of infrastructure into de XIFI framework and federation. In order to fulfil such integration, it will be necessary to define adaptation mechanisms to control and access network resources and assure the control of the required quality of the service. It will offer a common interface to the network layer functions and parameters, thereby implementing the Network-as-a-Service (NaaS) concept. Furthermore it is required to develop some adaptation components for monitoring several parameters of the infrastructures, both network and datacenter-based resources, along the different domains. Finally it will design and develop the tools that are necessary to facilitate the management of the XIFI federation and the Generic Enablers deployed in the different nodes of the federation.

1.4 General Architecture Overview

Beginning from a general perspective, the XIFI architecture aims to be a robust framework and provide high-availability. Based on these main requirements, the XIFI Architects foresee a deployment configuration as depicted in Figure 2**.** According to the initial set of 5 infrastructure nodes participating in the first stage of the project, two of them will hold a *master* role whereas the other three will be stated as *slaves*. A *slave* node is the node where only the required software for deploying and managing user services is installed. On the other hand, a *master* node is the one where, in addition to the features deployed on the *slave* nodes, the centralized parts of the federation services (i.e. the components needed to manage the federation) are deployed.

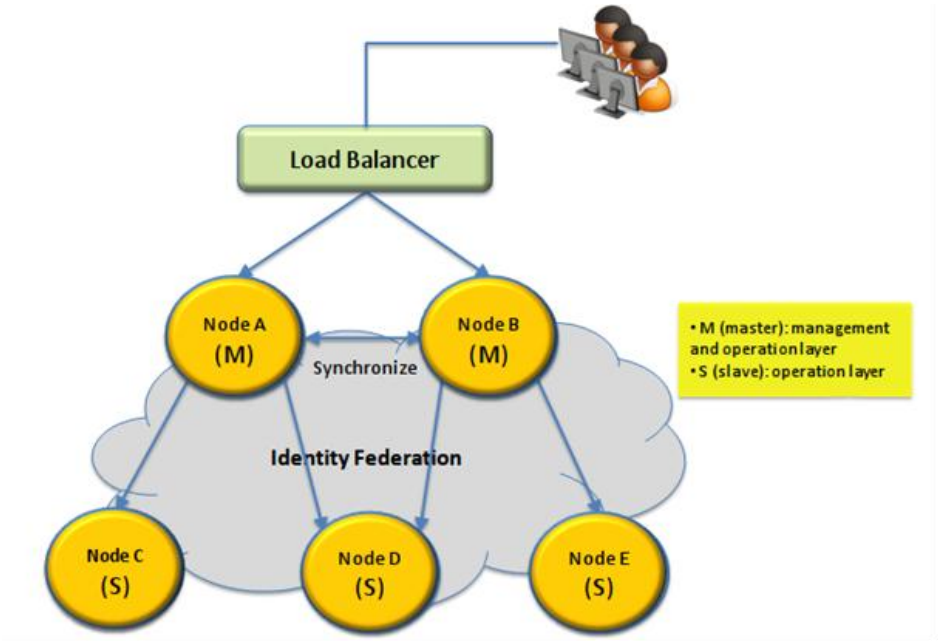


Figure 2: XIFI deployment architecture

Going into further detail, Figure 3 depicts the FMC compositional structure diagram which highlights the connectivity among the components in the XIFI architecture (see D1.1: XIFI Core Concepts, Requirements and Architecture Draft [66] for a deeper discussion). The lower area represents a generic node of the XIFI federation whilst the upper area contains the "federation services", i.e. the services offered to support the management and provisioning of recourses in a unified fashion. Such services are located only in the *master* nodes.

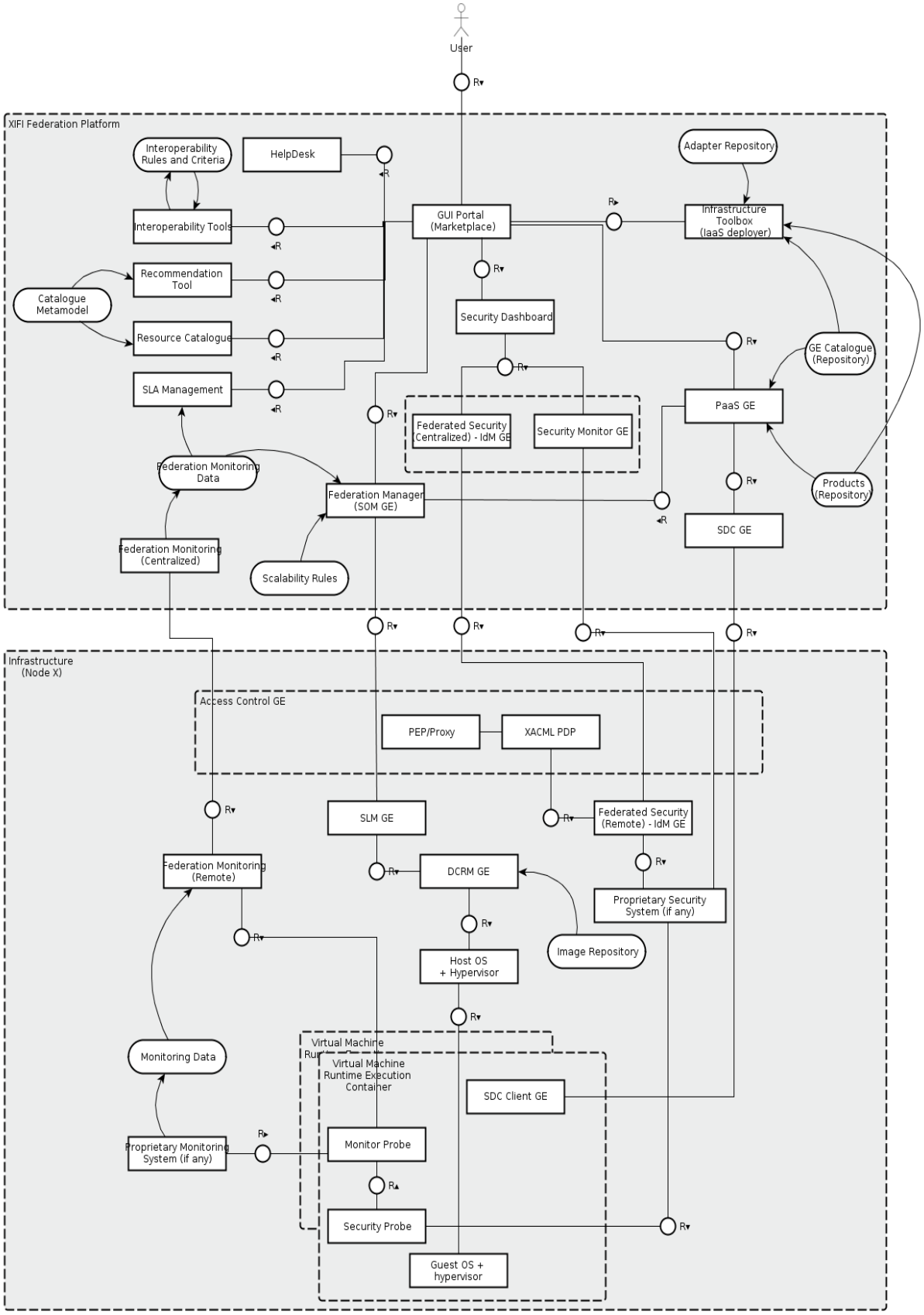


Figure 3: XIFI General FMC compositional structure diagram [67]

XIFI can be seen as a community cloud that provides service using the three traditional models of cloud platforms: SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service). Understanding this point is important for the design and development of infrastructure adapters and to elicit requirements on connectivity, monitoring and management services. Whilst some of the adapters addressed in this document have exclusive impact in the lower IaaS layer, others will be also related to the provisioning models of upper layers. The infrastructure monitoring and connectivity systems are deeply related to the IaaS layer, meanwhile management adapters are more oriented to the SaaS and PaaS layers. The discussion on provisioning models is better detailed in D1.1 [66].

2 CORE CONCEPTS

The following sections will provide a general introduction with regards to those main concepts that the reader should be aware in order to understand the context in which the designs and specifications of this document were developed.

2.1 XIFI Network-based Services Adapters

The goal of the XIFI federation and every other federation is to establish a sustainable marketplace for infrastructures and Future Internet services. In order to achieve this concept the integrating and federating of a multiplicity of heterogeneous environments needs to transpire. The interconnection of these environments brings its own constraints and degrees of freedom, both from an East-West data plane to share information, to a North-South managing interface that coordinates their decisions in a federation to allow external facing data connections to communicate. The coordination can be categorised in the following usage domains [32]:

- Business management: incorporates federation related qualities which primarily include reporting, service level agreements, analysis of infrastructure and business events, produce actionable information in user-specific environment, web-based views, etc.
- Service management: deals with services running on the network (discovery, operation, administration and maintenance services).
- Network management: management of the network from centralized visibility of resources, performance, services provision and network alarm management
- Element management: handles localized network controllers and network elements via protocols such as, OpenFlow, SNMP, etc. Functionality includes provision, discovery, and maintenance, with alarm management of the network devices.

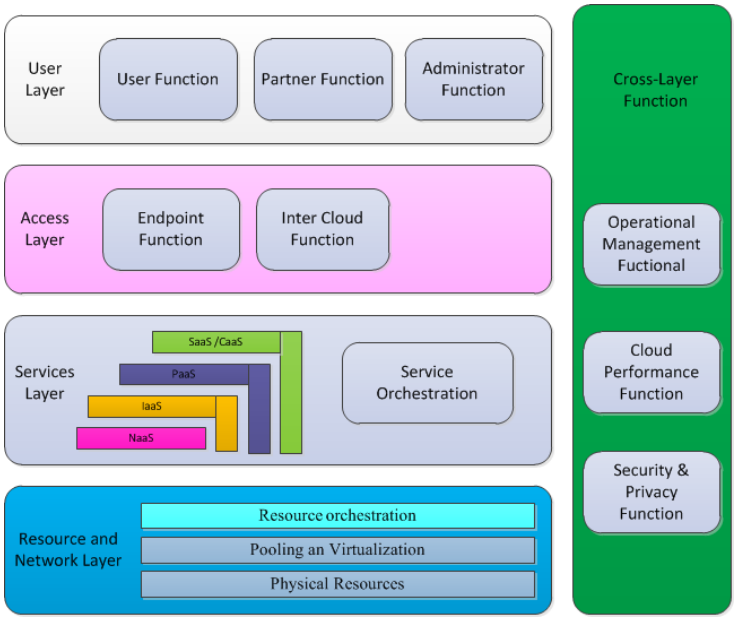


Figure 4: Cloud-layering framework [24]

Network-based Service Adapters aims to encompass all the previously mentioned management. This leads to the creation of a hybrid cloud where some of the resources are administered locally and will allow others to be provided externally, in this instance networking services. From the Cloud-layering framework reference model from Focus Group on Cloud Computing Technical Report [24] the network based service adapter is a concept that straddles the Access and Services layers shown in Figure 4.

A pure telecoms implementation is referred to as a northbound and southbound interface (NBI) often written in Java with Corba as middleware. It connects the network management platforms to the OSS/BSS deployment. Enabling network infrastructures evolve into a transparent platform while maintaining operations, administration and maintenance. When the NBI concept is transposed on to cloud it appears as a network adapter service that would act as a gateway for the Federation to interact with the hybrid cloud. Essentially a network-based services adaptor is software that allows external networking cloud services (Federation) to interoperate with underlining cloud instances.

Notably there are two standards bodies that are associated with cloud and cloud interoperability, Open Cloud Computing Interface (OCCI) from Open Grid Forum (OGF) [9] and Cloud Infrastructure Management Interface (CIMI) from Distributed Management Task Force (DTMF) [43]. As a result of the introduction of new cloud interconnections technologies, such as VXLAN, OpenFlow, etc, these federating standards are in still their first stages.

2.2 XIFI Infrastructure Monitoring Adapters

Carrying on with the introduction to the specific XIFI adapters, there is a set of them that will play a relevant role in the abstraction layer; the XIFI Infrastructure Monitoring Adapters. These adaptation components are in charge of providing an abstraction layer to the diverse set of monitoring systems of the infrastructure nodes, responsible for exposing and controlling relevant parameters which point their performance status. This will be an important milestone in order to assure the fulfilment of the required level of interoperability with the tools adopted by XIFI federation to ensure shared resource management.

Each infrastructure owner, independently from the intended service to XIFI project, requires monitoring the performance of the network and datacenter components present within the domain. This is the role assigned to a Network Operations Centre (NOC), for example. Therefore, the infrastructure shall be equipped with a monitoring framework providing metrics of status and performance. The XIFI monitoring adapters will aim to collect such measurement data, process it and feed the federation layer on top of the adaptation layer with added-value and standard data. This information will be exposed through a set of APIs.

To facilitate the inclusion of infrastructures that are willing to expose only minimal data, it can be defined different levels of compliance for the monitoring data to be provided.

- level 1) basic information about infrastructure availability (e.g. whether the infrastructure is currently accessible)
- level 2) information about availability of the single resources (e.g. whether there are computational resources free to use)
- level 3) information related to QoS of the single resources (e.g. the current load of the network, and its bandwidth usage)

XIFI monitoring adapters shall also provide the accessibility to check the status of those FI-WARE GEs deployed within the infrastructures, since the instances belong to the datacenters.

Section 6 of this document will go through a deeper assessment of this framework and address the specification of the **XIFI Infrastructure Monitoring Middleware (XIMM)**, the proposed solution to play the role of the monitoring adapters.

2.3 XIFI Generic Enablers Management Adapters

In order to boost the applicability of the concepts of FIWARE, the (Generic) Enablers should be seamlessly deployed, managed, monitored and configured on top of different, federated (XIFI) infrastructures. This is the main aim of the (Generic) Enablers adapters. Such adapters can support XIFI nodes (with the prerequisite of the existence of the cloud hosting FIWARE GEs or third party tools with equivalent functionality) and deal with the GE management in the following ways:

- Before and during the deployment phase: where homogenized way to package Enablers (including the solution of packaging as Virtual Machines and the usage of recipes within the context of FIWARE and the typical deployment tools). The adapter defines a higher layer to verify the availability of resources prior to the deployment of a GE and furthermore to simplify and homogenize the deployment on top of different nodes. This functionality is made available through the GE Deployment and Configuration API that also supports the sharing of information on the Enablers installed on independent nodes towards the overall XIFI federation.
- After the deployment: focusing on monitoring the operations of the Generic Enablers (GEs), directly from the Runtime Execution Container (REC). This allows the adapter to retrieve data that can support the infrastructure providers and the users (through the Marketplace) to select the usage of Enablers and balance the resources against their requirements by using the most suitable nodes and resources. Such data can also be transformed to SLA - related information.

2.4 Software-Defined Networking (SDN)

Cloud-based services allow end-users to deploy their applications and services in a distributed manner, even making use of computing resources allocated in separated datacenters. Those datacenters communicate traversing networks presenting different technologies and topologies, and typically being part of distinct administrative domains. Additionally, such resources can be dynamically re-assigned within and between datacenters changing on-the-fly both the overlay service topology and the corresponding traffic load.

The delivery of distributed cloud services implies the configuration of various capabilities across the involved networks such as forwarding and routing rules, quality of service guarantees, and security aspects. Some other aspects rely on the own network capabilities, like failure recovery or congestion control. This complex and heterogeneous environment can be managed in a more simple, scalable and flexible way by means of the Software Defined Networking (SDN) paradigm [5] [44].

A SDN is a programmable network in which the data plane of the traffic forwarding nodes is configured from a separated control element (SDN controller) according to specific requirements from distinct applications. SDN starts from the idea of decoupling the control and data plane in contrast to the tight integration observed in conventional network equipment. This approach allows the network infrastructure to match the dynamism of the cloud computing services.

SDN facilitates the change of the network behaviour by modifying the forwarding behaviour of the nodes. A programmatic interface is defined to interact with the data plane functions of the network nodes. OpenFlow is one realization of such interface.

The OpenFlow protocol permits to define rules for individual packet flows according to specific fields on the packet header at either layer-2, layer-3 or layer-4 levels. These rules are populated in the form of flow table entries that result on the forwarding configuration of the nodes. Sophisticated flow handling can be achieved by cascading a number of flow tables before the flows are delivered to the egress port. This fine grained control provides high flexibility.

OpenFlow provides a vendor-agnostic, abstract interface that helps to integrate heterogeneous multi-vendor networking environments. At the same time it facilitates the orchestration of network resources because the representation of the node behaviour becomes generic and independent of the particular hardware implementations and command syntax.

In large distributed environments the deployment of a single SDN controller to manage all the required connections does not scale. The existence of different administrative domains and a large number of devices complicates the operation and compromises the scalability of the controller.

A potential solution is the federation of non-overlapping SDN-based domains. However, the existence of multiple domains raises the problem of controller coordination for the provision of an end-to-end connectivity service. SDN controllers interconnection has been proposed in [61] as a way of exchanging information between them. Nevertheless, in such a federated environment it is yet required a logically centralized view of all the network resources available to set up the connectivity service, and an orchestrated action to provision it. These two requirements need to be addressed for a consistent delivery of a connectivity service between SDN domains.

The XIFI environment, as already described, is composed by a federation of distributed datacenters offering a shared infrastructure for cloud-based services. The inter-datacenter connectivity service to be provided by XIFI faces a number of requirements for network resources as described in [4], specifically the support policy based control on flow by flow basis in a fine-grained manner and the dynamic adaptation to the traffic generated by cloud services. Network programmability is the mean identified to accomplish them.

When deploying their applications and services across distributed datacenters, end-users do not need to assume any specific negotiation with XIFI for the provision of network connectivity. The XIFI architecture should enable a single operation point for both computing and network resources. Then an integrated provisioning system is required to automate the connectivity setup locally in the distinct datacenters. To achieve that goal a XIFI network controller is designed to orchestrate the complete service provisioning process. The XIFI network controller will intelligently interact with the SDN controllers deployed locally in each datacenter, as part of the federation of SDN-based domains. Furthermore, the XIFI network controller will complement the local SDN controllers by providing the needed coordination among multiple independent cloud systems residing in different domains and connected through different networks. The next sections describe in detail the XIFI network controller. An SDN controller, by its own, is not able of implementing the network management capabilities required for supporting the distributed connectivity of the XIFI federated framework, as well as dealing with the integration of the envisaged heterogeneous network environments.

2.5 Bandwidth on Demand (BoD)

Bandwidth-on-Demand (BoD) services have received much attention as a means of providing higher, guaranteed bandwidth with less jitter than traditional IP connections and avoiding the requirement or resources for a permanent circuit. If a dedicated circuit is chosen, the task of manually provisioning multi-domain circuits is an impractical and costly process for a one-off or short term circuit requirement, often taking weeks to set up the required end-to-end connection. BoD service concept aims to overcome such limitations.

Bandwidth on Demand (BoD) is a service for dynamic bandwidth provisioning, potentially across multiple domains and networks, enabling to instantly create dynamic point-to-point circuits.

Conventional BoD services enable users to specify service parameters such as the start and end points, the required bandwidth and the holding time. Additionally, some recent BoD implementations also provide with a tracking circuit option that enables users to monitor and check the status of their service (i.e. requested circuits). Some desirable features requirements of BoD tools may include:

- **Multi-domain service:** BoD services so far have only been available within a single domain. Despite the fact that ad-hoc multi-domain circuits have been created before, recent BoD services bet for multi-domain bandwidth-on-demand service.
- **Dynamic provisioning:** Traditionally, creating multi-domain point-to-point circuits has been a time and resource-intensive process. Modern BoD service's and advanced provisioning tools now enable circuit paths to be found and created dynamically with just a few clicks of the mouse.
- **Flexible connections:** BoD services offer dedicated point-to-point circuit, guaranteed capacity, no congestion, and deterministic performance, but are also incorporating the flexibility of an IP service.
- **Circuit Protection:** BoD systems should be able to provide with either no protection, partial protection or full protection, so as to survive failures, e.g. physical layer cuts.
- **Reduction of operational and administrative costs** for service providers. BoD process automation may enhance the service provisioning also on service providers´ side.
- **In-advance BoD Reservation:** Users may be capable to reserve bandwidth for planned events or experiments in advance.

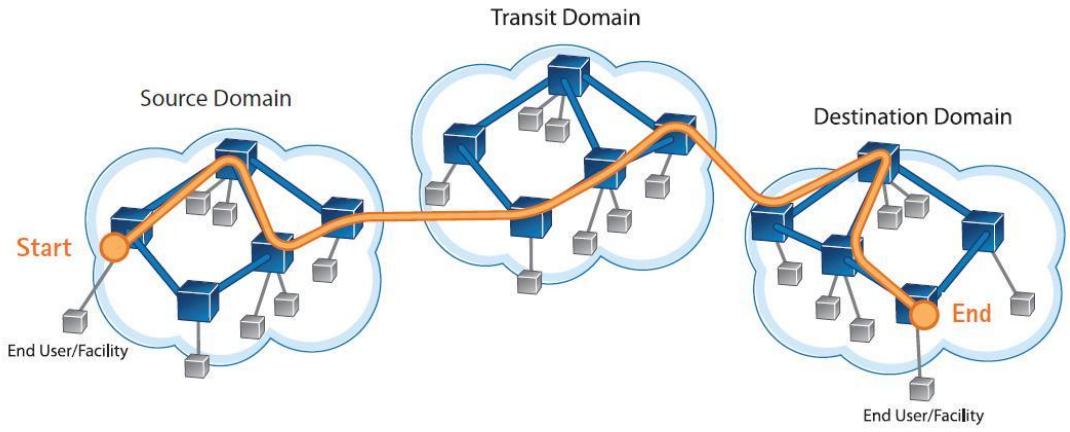


Figure 5: Generic E2E BoD circuit in a multi-domain infrastructure [26]

2.6 Network as a Service (NaaS)

NaaS is a business model related to network infrastructure servicing for delivering network services virtually over service providers´ infrastructures and the Internet with direct, dynamic, scalable, yet secure and isolated tenant access to a network. NaaS model abstracts the underlying network complexity and offers network functions and capabilities as a service and enabling the possibility to easily deploy and operate customized advanced network services and deliver reliable and predictable performance according to on-top application needs. Besides, it also brings up the possibility to delegate management permissions and control functions to users/customers so that they are capable to configure the service in a customized and flexible way. For these reasons, four top characteristics of NaaS have been identified:

- Decoupled network resource management and control from actual services.
- Abstracted operational model.
- Coordinated management and control functions along different strata.
- Policed resource and capabilities access, depending on different resource access rights and ownership patterns.

NaaS concept represents an interesting service model from which XIFI service providers (e.g. NRENs) and potential customers (experimenters, research institutions) can benefit: The formers have the opportunity to become NaaS providers and enable to customers with isolated frameworks, and delegated permissions. On the other hand, organizations should have the capability to manage any part of the infrastructure and resources they receive or to deploy any type of application or service on top.

NaaS capabilities also represent a challenge for the development of added-value virtualization services, as they can extend the IaaS model from Datacenters to access and transport networks. Besides, NaaS may enable the evolution of network management and operations to increase flexibility in operations and potentially reduce OPEX in the short/mid term.

2.7 Infrastructure as a Service (IaaS)

The term Infrastructure as a Service is defined as a Cloud Service Model by NIST [63]. The concept of Infrastructure as a Service (IaaS) [41] is characterised as a set of resources made available to cloud consumers by cloud providers via a set of interfaces. These are typically physical storage, computing, network and other resources which are subdivided between cloud consumers using various virtualisation techniques and provided using cloud software. The cloud consumer is provided with an interface to manage the requesting and releasing of the resources while not having low-level physical access to them. They can use these resources as they choose without having to worry about physical maintenance.

IaaS is divided into the resources and facilities that are managed and hypervisors that provide those facilities to cloud consumers. The cloud provider seeks to ensure all facilities support high level of automation so operation costs are kept to a minimum. This includes the use of integrated platform management interfaces (IMPI), Hypervisor interfaces, generic abstractions around differing hypervisor interfaces, storage system interfaces such as the Cloud Data Management Interface (CDMI) and network resources.

2.8 Platform as a Service (PaaS)

Platform as a Service (PaaS) refers to the provision of a computing platform and the provision and deployment of the associated set of software applications (called a solution stack) to an enterprise by a cloud provider. PaaS makes deployment of applications possible across your entire network while negating the need for you to buy or maintain hardware or software, basically because it is all owned and maintained by the cloud provider. The provider can customise, integrate or develop from scratch, applications needed by the enterprise, which the client can then access via the internet.

PaaS platforms also have functional differences from traditional development platforms. These include:

- **Multi-tenant development tool**: traditional development tools are single user - a cloud-based studio must support multiple users, each with multiple active projects.
- **Multi-tenant deployment architecture**: scalability is often not a concern of the initial development effort and is left instead for the sys administrators to deal with when the project deploys. In PaaS, scalability of the application and data tiers must be built-in (e.g., load balancing, failover need to be basic elements of the dev platform itself).
- **Integrated management**: traditional development solutions usually do not concern themselves with runtime monitoring, but in PaaS, the monitoring ability needs to be baked into the development platform.
- **Integrated billing**: PaaS offerings require mechanisms for billing based on usage that are unique to the SaaS world.

The benefits of PaaS lie in greatly increasing the number of people who can develop, maintain and deploy web applications. In short, PaaS offers to democratize development of web applications much the same way that Microsoft Access democratized development of client/server applications. It offers the potential for general developers to build web applications without needing specialized expertise.

3 REQUIREMENTS

3.1 Global Requirements

As stated in the D1.1 [66], XIFI should assure the setup of a community cloud, a federation of ICT infrastructures offering cloud services to the users interested to experiment innovative idea in the Future Internet field. XIFI bases its offer on the provision of the FI-WARE Generic Enablers in both a SaaS or PaaS model depending on the specific GE requested and on the needs of the experimenters. Since the XIFI federation is composed by different infrastructures located in different Nations (up to 5 in the first phase), we decided to create a two level architecture where two infrastructures will assume the role of master and the other three (in the first phase) the role of slaves. This means that the master nodes will be responsible of the management and operational services whilst the slave ones will be responsible only of the operational services. See Architecture Overview (section 1.4).

The previous architecture, together with the survey on the five XIFI nodes and on the UC projects, imposes some minimal requirements in terms of hardware and network connection among the nodes (see D1.1). These requirements are reported in the Infrastructure Requirements section hereunder.

These requirements come from the needs to allow the installation on all the XIFI nodes of the FI-WARE Cloud hosting stack based on the DCRM Generic Enablers (see DCRM description [14]).

Digging a bit more into the requirements imposed to the XIFI Monitoring System, and considering the deployment architecture aforementioned, it should be taken into account that many infrastructures can already have their own monitoring system and that some data collected by these systems can be "private" to the infrastructure and not shareable among other subjects. Moreover the monitoring data collected should be handled with care because in many cases they are not public but can be accessed only by the relevant users (multi-tenancy).

All this means that the XIFI Monitoring System should be able to support a large range of data collection protocols in order to be able to gather data not only from a variety of network devices, computer devices, services and virtual machines but also from different existing monitoring system (already installed on the infrastructure premises).

The estimated amount of monitoring data collected from each infrastructure could be very big and for this reason some data aggregation and filtering together with big data analysis techniques are needed.

In order to accomplish the previous mentioned requirements, a distributed architecture of the XIFI Monitoring System is needed:

- The data is collected locally for each infrastructure directly from the devices/services or indirectly from the existing monitoring systems,
- Then the data is filtered, normalized, stored (so to keep the detailed data persistent locally) and sent to a central monitoring systems (located only on the master nodes)
- On the master nodes the data is finally aggregated, stored (so to keep the aggregated data persistent at "federation" level) and is ready to be used by the other XIFI services like for example the XIFI portal or the XIFI SLA Management.

Many of these functionalities can be satisfied by the FI-WARE Monitoring GE (see [16]) and for this reason the implementation of the Monitoring System in XIFI will trust on the usable of this GE.

3.2 Infrastructure Requirements

XIFI infrastructure can be categorised in these ways: server hosting, inter-node connectivity, internet access, NREN (National Research & Education Network), testbed, experimentation and research lab.

The users could be: research centres and SME (Small/Medium Enterprises), academic users, educational institutions, or ICT companies interested in FI software development in EU.

Its main technical features can span over data centres, backbone networks, mobile access networks and IoT communications.

The capacity of the infrastructure in terms of connectivity, hardware and software must comply with what is defined by the deliverable D1.1 [66], in particular the sections "Minimal network and capacity requirements for each node" and "FI-WARE Cloud Hosting Main Constraints". These sections are summarised here for clarity:

- **Connectivity (for each node)**:
  - Inter node connectivity will be done via classical IP routing. P2P and other solutions will be evaluated later on.
  - The backbone will be implemented using IPv6.
  - Service to end users will be provided using dual stack IPv4/IPv6.
  - Bandwidth between any two master nodes should be at least 1Gbit/s.
  - Bandwidth among the other nodes should preferably be of the order of 1Gbit/s.
  - Connectivity to the general internet should be at least 100 Mbps.
- **Hardware (for each node)**:
  - 100 CPU cores
  - Core types: Intel VT-x or AMD AMD-v
  - RAM 2 GB x Core
  - HD 20 GB x Core
  - SEC Firewall
- **Software requirements (FI-WARE DCRM Cloud Hosting):**
  - Operating system: Ubuntu 12.04
  - Hypervisor: KVM
  - IaaS Manager: OpenStackGrizzly version (all basic components plus Quantum and Swift)
- **Hardware requirements (FI-WARE DCRM Cloud Hosting)**:
  - CPU: 8 cores (>=2.4 GHz, VT-x enabled)
  - RAM: 16GB
  - Disk Space: 1TB

Each infrastructure should provide:

- An assurance management system to detect and signal failures, monitor basic performance parameters like bandwidth, delay, packet loss (e.g. PerfSONAR) according to the selected tools.
- Security service starting from a firewall with VPN capabilities for remote access, to optionally cover AAA system like EduGAIN.
- An internal monitoring system that will collect the following main measurements/parameters for the devices:
    - Uptime for devices and services provided
    - Status of hardware interfaces
    - Traffic for interfaces
    - Device specific information (SNMP enterprise MIBs)
    - Alarm and faults traps
    - CPU load
    - Memory usage
    - Disk usage
    - Temperature (suggested)
    - Power consumption (suggested)
- The possibility to share the collected monitoring data if needed, in accordance with tools and procedures and with the internal policies of each infrastructure owner and non-disclosure agreements with third parties.

Note that each location is different and they have their own requirements or restrictions. For example, HEAnet and GÉANT are network providers and do not provide compute resources. Partners may also be using existing monitoring systems or other systems in production and may not be able to replace these specifically for the XIFI project. Any integration that is needed must be pursued with the infrastructure providers in detail.

3.3 FI-WARE Platform Requirements

Appendix A presents a table where can be found software and hardware requirements for the deployment of the Generic Enablers provided by the FI-WARE Catalogue [11]. This effort falls within the interest to ensure that Generic Enablers can be seamlessly deployed, managed, monitored and configured on top of different infrastructures, including the five XIFI nodes during the first year of the project lifetime.

Some general remarks:

- All GEs require 2 to 4 CPU cores, apart from DB Anonymizer (Security chapter) and DCRM (Cloud chapter) requiring 8 cores.
- All GEs require 2-4GB of RAM, apart from DB Anonymizer (Security chapter) and DCRM (Cloud chapter) requiring 16GB of RAM.
- All GEs require 5-50GB HDD, apart from DB Anonymizer (Security chapter) and DCRM

(Cloud chapter) requiring 0.5T and 1T of HDD, respectively.

- A number of OSes is used (Ubuntu, Windows, CentOS) with different versions (for example: CentOS v5 and v6.3 is used).
- DB requirements are mainly limited to MySQL and MongoDB.
- Application and Web servers used are mainly Tomcat and JBoss.
- Of particular interest is the variety of other software components (see Table below for details).
- Regarding the GE requirements on the number of physical nodes (servers), DCRM requires 5 nodes with specific requirements on software (see Table in Appendix A for details).

4 BACKGROUND AND PRIOR ART

This section provides insight information on the state-of-the-art of some remarkable solutions that shall be relevant in our specific context. The general purpose of this document is to provide details regarding the specification of adapters; development details are considered out of scope at this stage. Nevertheless, the tools and systems which are addressed within this section will be used or will play a significant role to face the future implementation of the tackled specifications.

- Starting with **Network-based services management systems**, traditional and open access management approaches, dealing with the operation, administration, maintenance and provisioning (OAMP) procedures in any phase of a network life-cycle, are presented. Moreover, SDN concept is described which can be applied to network management problems considering that network configuration consists in processing events in domains such as time, history, user or traffic flows.
- Also, **measurement tools and systems** state-of-the-art is discussed, focusing on specific tools considered as viable solutions for the XIFI federated platform, including PerfSONAR, Nagios, etc.
- Finally, **GEs management tools** are described mostly based on FI-WARE developments so far, such as PaaS and SDC.

4.1 Network-based Services Management Systems

Proper management is essential for the operation, administration, maintenance and provisioning (OAMP) procedures in any phase of a network lifecycle. The set of functions that a management system supports over network resources such as planning, deployment, configuration or monitoring, can be implemented following different architectures and levels of automation involving less or more complex software systems. The evolution of management systems has been driven to solve problems as scalability, security, interoperability, and service management challenges. Besides, management technologies have a significant impact on the management applications and network design.

4.1.1 Traditional Management

According to [2] there are several traditional management approaches that may be classified into different categories:

- **Centralized:** The centralized approach assumes the existence of a single system that controls the whole network of elements, each of them running a local management agent. The centralized manager coordinates a potentially very large number of agents. This centralized manager, typically known as NMS (Network Management System) monitors and administers one or more networks, composed of (NEs) Network Elements, known as managed entities.

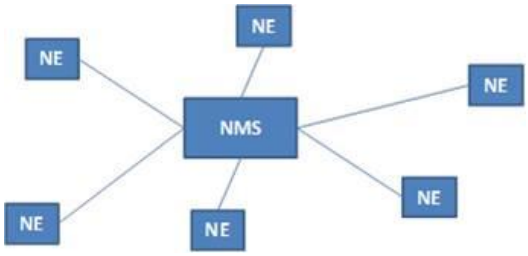


Figure 6: NMS Centralized Approach

- **Weakly and strongly distributed hierarchical:** The distributed management approaches introduce the concept of a management hierarchy, in a way similar to the human organizations. With these approaches the Central Manager (NMS), delegates part of the management load among Agent Managers, each responsible for a part of the network. The Agent Managers typically do not exchange information between them, being the Central Manager the single responsible for the coordination of the whole network management. Other approaches define a more intense cooperation between the different Agent Managers. These approaches are known as Strongly Distributed approaches. Weakly distributed approaches have the advantage of lowering the processing needs of the NMS, but basically they suffer from the same constraints as the full centralized model. The NMS is still a single point of failure, and the Agent Manager acts as a "mini"-NMS, responsible for a part of the network. On strongly distributed approaches there is a relation between the Agent Managers, making them able to delegate decisions. However, the NMS is still the single full intelligent entity.

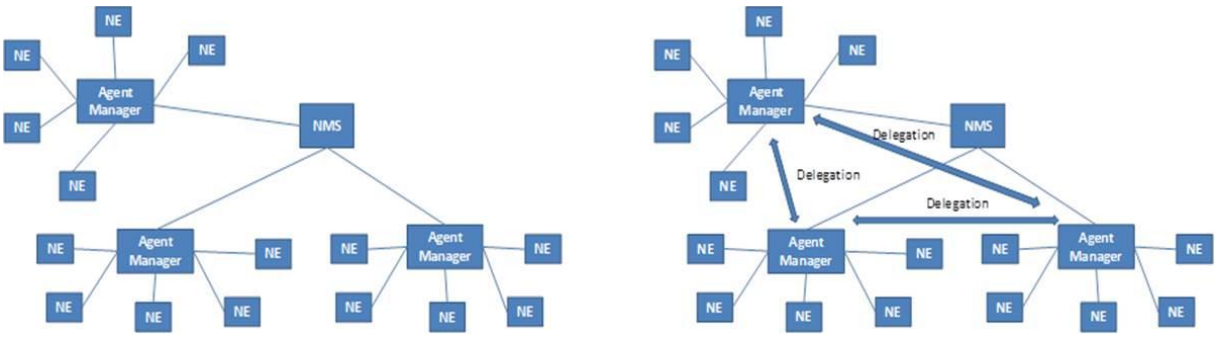


Figure 7: NMS Weakly and Strongly Distributed Approaches

- **Cooperative Paradigms:** In cooperative approaches, as the name indicates, the main characteristic is that the Agent Managers cooperate between them and there is also cooperation between an Agent Manager and a NE belonging to another Agent Manager area. These entities act in both managers and agents, and work together in a collaborative way fulfilling management tasks.

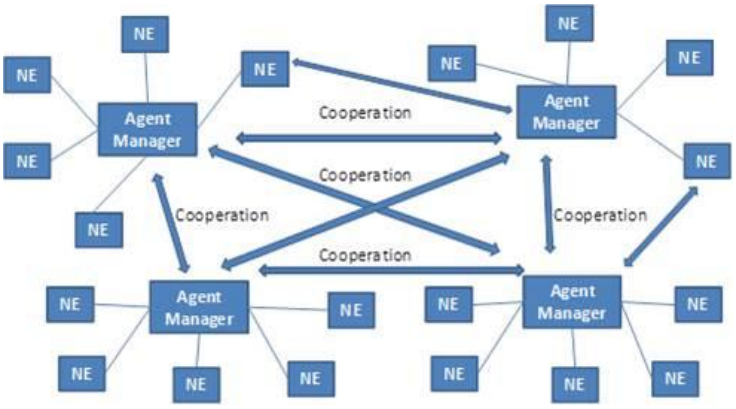


Figure 8: NMS Cooperative Management Approach

4.1.2 Open Access Management

To each of these approaches, another dimension may be applied: the Open Access concept, in which the roles of network and service providers are separated and the network provider offers tools and/or interfaces to partition its network into a number of self-contained and virtualized smaller networks, able to be managed by different service providers. The traditional telecom model is based on vertical integration, in which one entity delivers the service, operates the network, and owns the network infrastructure. The open network model, in which services are provided on a fair and non-discriminatory basis to the network users, is enabled by conceptually separating the roles of the service provider and the network and communication operator. Due to the different technical and economic nature of the different parts of the network, different roles and actors can be identified.

4.1.3 SDN-based Network Management

Nowadays, management is based mainly on software systems allowing different levels of automation of the different management processes. In this sense, Software Defined Networking (SDN) is quickly becoming an important strategy in addressing the network needs that companies have right now. From a functional perspective, SDN is the physical separation of the network control plane from the forwarding plane, where a control plane controls several devices [62].

The SDN concept can be applied to network management problems considering that network configuration consists in processing events in domains such as time, history, user or traffic flows. Therefore, novel management systems could benefits from applying SDN architectures in a similar way that OpenFlow currently allows controlling flows. Figure 9 depicts the different SDN layers as proposed by the Open Networking Foundation.

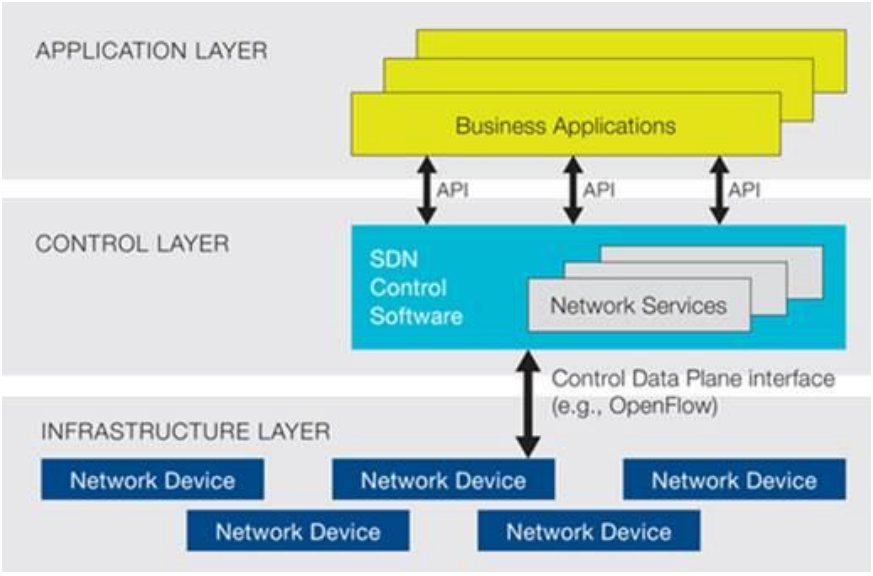


Figure 9: SDN Architecture Layers [62]

The same ONF [46] is promoting the OpenFlow standard [47] as the communication protocol to access, manage, and configure the data plane from the control plane. Although typically SDN started using OpenFlow, which lead to misunderstand OpenFlow to be equivalent to be SDN, there is no requirement for the use of OpenFlow within an SDN environment.

The SDN concept can be applied to network management problems considering that network configuration consists in processing events in domains such as time, history, user or traffic flows. Therefore, novel management systems could benefit from applying SDN architectures in a similar way that OpenFlow currently allows controlling flows.

4.1.4 OpenNaaS

The NaaS model has been brought forward with the OpenNaaS framework [48] for easy prototyping and proof-casing of XIFI network connectivity service. OpenNaaS is an open-source framework, developed under the FP7 MANTYCHORE project [33], which provides tools for managing the different resources present in any network infrastructure.

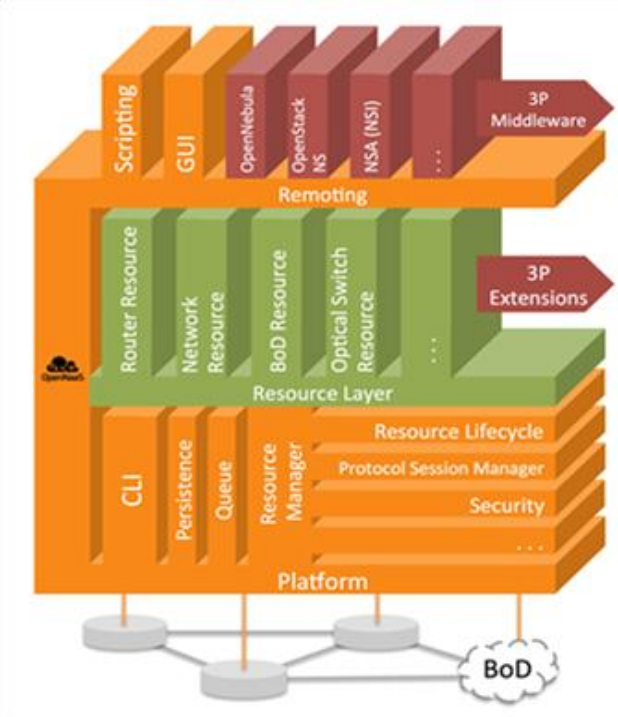


Figure 10: OpenNaaS Architecture [48]

In the specific case of XIFI connectivity service, OpenNaaS framework is part of the XIFI network controller, enabling users to access to the isolated control and management of the infrastructure provided to connect the XIFI nodes. Besides OpenNaaS interoperates with other network controller modules to achieve the E2E inter-DC XIFI connectivity service.

4.2 Network Monitoring Tools

4.2.1 PerfSONAR

With the increase of distributed computing over multiple administrative domains came the requirement to measure network performance characteristics and share this information between the users and service providers. This was addressed by the Open Grid Forum Network Measurement and Network Measurement and Control working groups [45], which defines a set of protocols standards for sharing data between measurement and monitoring systems, often called the NMWG protocol. PerfSONAR (PERFormance focused Service Oriented Network monitoring Architecture) [56] is a framework that implements these protocols for both regular periodic observations, useful for forming historical records, and for making on-demand measurements to aid problem solving and resolution. PerfSONAR enables multi-domain network monitoring of the following performance characteristics:

- Packet loss
- One-way delay and jitter
- Achievable UDP and TCP throughput
- Network traffic utilisation

- Switch and router statistics such as receive errors and buffer overruns
- Traceroute

The web service-based infrastructure was developed by several international partners from Internet2, ESNet and the GÉANT2/GÉANT3 projects with the purpose of providing network administrators and research users with an easy access to cross-domain performance information and facilitating the management of advanced networks. The framework is made up of several components or services including:

- The Measurement Point (MP) services that provide measurement data. This may be done either by initiating active measurement tests or querying passive measurement devices or existing database. A MP provides a standard interface "wrapper" around one or more measurement tools that make the measurement observations of the network characteristics. Often the MP stores the observation just made using an archive service.
- Measurement Archive (MA) services are used to record and publish historical monitoring data which are stored in an archive. Any form of database may be used, for example a Round Robin Database (RRD MA), a relational database (SQL MA) or a proprietary database of a Network Management System.
- The Lookup Service (LS) enables users to discover other services (MP, MA, Authentication service, etc.) and, in particular, the tools, capabilities, or data offered by those services.
- User Interface (UI) provides several methods of visualizing the measured network characteristics in tabular and graphical forms as well as providing the interface for making on-demand measurements.

At each MP PerfSONAR uses specific network measurement tools to perform the measurement of the network characteristic between the selected PM end points. For example, Iperf [31] is uses for making TCP or UDP achievable throughput measurements by the BWCTL service; and the owping tool is used to make one-way delay, jitter and packet loss measurements by the OWAMP service. PerfSONAR is designed to allow new tools to be added as required for future services. The blue lines in Figure 11 represent the measurement observations made between MPs by Iperf or owping tools. The MPs and MAs are "controlled" by a web service layer using the OGF NMWG protocols discussed above. The interactions between the MP, MA and UI PerfSONAR services are indicated by the red lines in Figure 11.

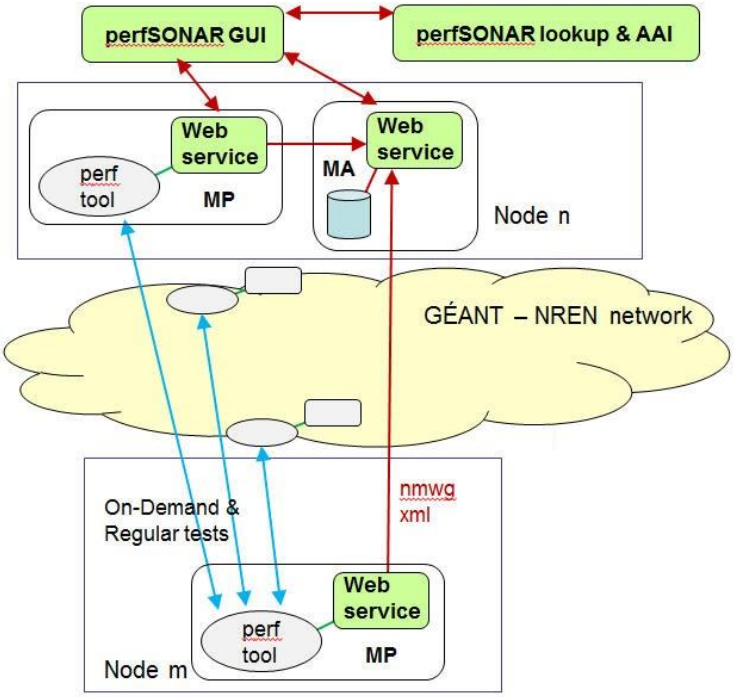


Figure 11: Demonstration of how the PerfSONAR services may be linked together

The recording of traffic utilisation and switch/ router statistics is a little different as indicated in Figure 12 which shows a typical PerfSONAR RRD MP/MA. It uses a standard tool such as MRTG or Cacti to access the router or switch with SNMP and place the results into the RRD database. Depending on the configuration of the tool, the router or switch may be local or remote to the RRD database.

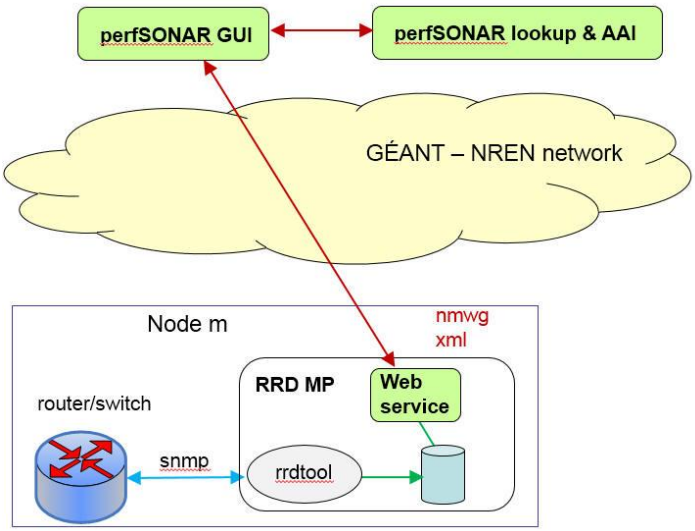


Figure 12: Illustration of how PerfSONAR provides network utilisation statistics

Currently there are available two main open-source implementations which may be deployed; PerfSONAR MDM and PerfSONAR-PS are respectively the GÉANT and the Internet2/ESnet implementation of the PerfSONAR monitoring concept and infrastructure. They both use the PerfSONAR protocol (specified by OGF NM-WG) to exchange data, and share the same overall design goals which include flexibility, extensibility, openness, and decentralization. At the time of writing, ESnet and GÉANT are discussing the optimum way to integrate the implementations, hence improving the services and reducing the resources required for on-going maintenance.

Even if they share a common view, the approach is taken in a different manner (software development, product life cycles, interaction with the users, the actual implementation and deployment). The differences are mainly due to the fact that PerfSONAR MDM is designed to provide a service, with an official support and a federated deployment, centrally monitored and coordinated, while PerfSONAR-PS is intended o be a collection of monitoring and troubleshooting tools to be downloaded when needed. Document [54] assesses a fine-grained comparison of them.

4.2.1.1 PerfSONAR MDM

PerfSONAR Multi-Domain Monitoring (MDM) has been developed within the umbrella of GÉANT to enable an easy and troubleshoot of problems on communications that flows between several administrative domains. The scope of this implementation is the ensemble of European research network interconnected.

Some specific features of this implementation are listed below.

- Written in Java
- Available in Debian or RPM packages
- Bootable USB stick
- Virtual Machine image
- Support from GÉANT Multi-Domain Service Desk
- PerfSONAR-MDM 3.3 Components are:
    - RRD Measurement Archive
    - SQL Measurement Archive
    - Lookup Service
    - BWCTL Measurement Point)
    - Easy to use web based visualisation tool

For a deeper analysis, it is recommended to review the PerfSONAR MDM Documentation [53].

4.2.1.2 PerfSONAR-PS

As the previous case, some key features of this implementation are provided to have a general idea. To find further details, check their website [58].

- Collaboratively developed by: ESnet, Fermilab, Georgia Tech, Indiana University, Internet2, SLAC and The University of Delaware
- Written in Perl. Available for independent deployment or through the Performance Toolkit (pS-PT)

The PerfSONAR Performance Toolkit (pS-PT) [57]:

- Current version: 3.3
- LiveCD or Net-Install based on CentOS 5.5
- Boot with the installation CD
- Is easy to deploy
- User-friendly Web Interface

4.3 Datacenter Monitoring Tools

4.3.1 Collectd

Collectd [8] is a daemon which collects system performance statistics periodically and provides mechanisms to store the values in a variety of ways, for example in RRD files. Those statistics can then be used to find current performance bottlenecks (i.e. performance analysis) and predict future system load (i.e. capacity planning).

Everything in Collectd is done by plugins, except parsing the *configfile*. Collectd is able to handle any number of hosts, from one to several hundred (or possibly thousand, but no one has reported that yet). This is achieved by utilizing the resources as efficient as possible, e. g. by merging multiple RRD-updates into one update operation, merging the biggest possible number of values into each one network packet and so on. The multithreaded layout allows for multiple plugins to be queried simultaneously – without running into problems due to IO-latencies.

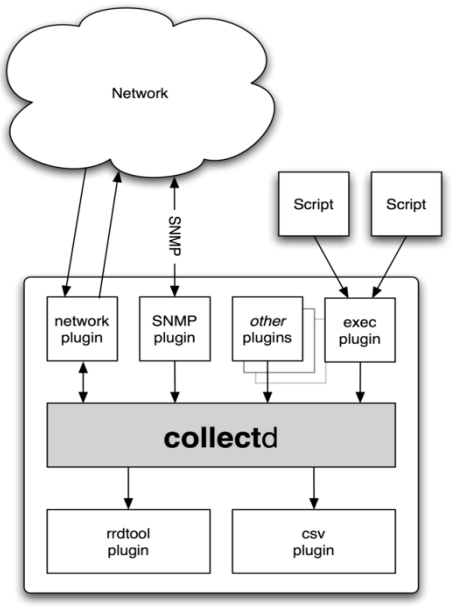


Figure 13: Collectd Architecture overview [8]

Some public and relevant information can be:

- Collectd Documentation [7]
- Collectd Canonical repository [6]

4.3.2 Nagios

Nagios [39] is the de facto industry standard for monitoring IT infrastructures that empowers organizations to identify and resolve IT infrastructure problems before they affect critical business processes. Nagios offers more than 3,000 plugins, spanning from system metrics, network protocols, applications, databases, services and servers. Moreover, Nagios provides means for reporting and alerting. It is based on an open source software license under the terms of the GNU General Public License version 2, as published by the Free Software Foundation. A detailed list of available plugins can be found in [38]. Although Nagios was originally designed to run under Linux, it shall work under most other machines.

By using plugins and add-ons, Nagios offers the capability to monitor host resources and processes (CPU load, memory usage, disk utilization, etc) as well as network services (SMTP, POP3, HTTP, etc.) on Windows and Linux/Unix machines. As a high-level view example, monitoring Windows-based host resources requires installation of the *NSClient++* add-on on the host machine and *check_nt* plugin on the Nagios monitoring server, as depicted in Figure 14.

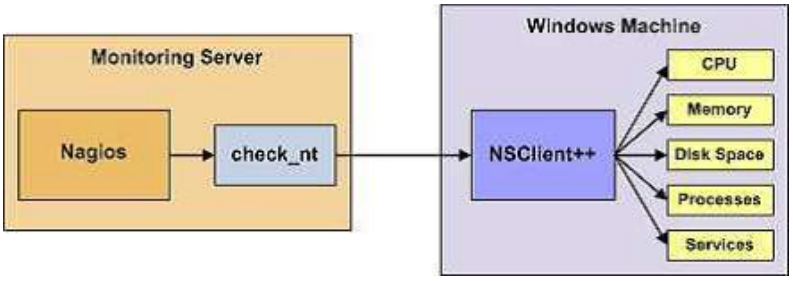


Figure 14: Nagios monitoring Windows-based host resources [39]

Similarly, Linux/Unix-based host resources can be monitored by installing the *NRPE* add-on, allowing to execute plugins on remote Linux/Unix hosts (see Figure 15).

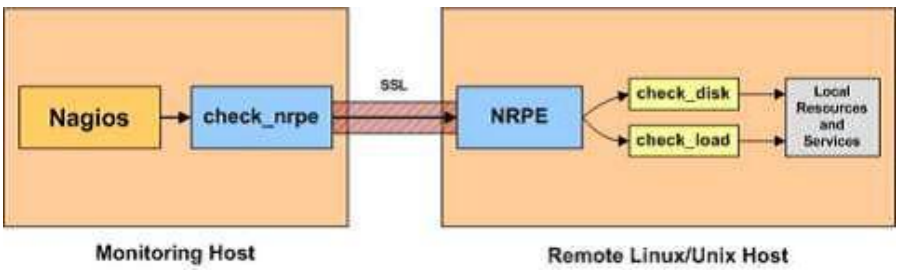


Figure 15: Nagios monitoring Unix-based host resources [39]

Reviewing the information obtained from the surveys contained in D1.1 [66], the results revealed that the majority of the infrastructures which initially will take part in the XIFI federation already deployed Nagios in their IT infrastructures. Hence they are aware of the services provided by this enterprise-class monitoring solution. That is a strong argument to consider this tool as a relevant component in the XIFI adaptation mechanism.

Unlike many other monitoring systems, Nagios does not include any internal mechanisms for monitoring hosts and services on the network. Instead, it relies on external programs (called plugins) to perform the measurements. Plugins are compiled executables or scripts (Perl scripts, shell scripts, etc.) that can be run from a command line to check the status or a host or service, while the results are used by Nagios to determine the current status of hosts and services on the network. The advantage of this approach is that one is free to monitor anything given that the process can be automated. On the other hand, the disadvantage relies on the fact that Nagios has absolutely no idea about what it is monitored; it just tracks changes in the state of the monitored resources or services.

Nagios supports add-ons that allow the environment to monitor a large infrastructure. More specifically, it provides two different options for setting up a distributed monitoring environment:

- In case of relatively large installations that require off-loading checks of the monitoring server to other machines, Nagios provides an add-on that faces the problem of scalability and complexity of distributed Nagios setups, following a master/slave configuration approach. In this respect, the Nagios server hands out jobs to slave nodes, while the master server contains all the configuration and check definitions [10].
- A different approach is followed in [37] and [36], where several Nagios servers that monitor a portion of the entire infrastructure are set-up, managed by a central dashboard that allows checking quickly the status of resources and services belonging to different portions of the infrastructure from a single server. In contrast to the previous distribution approach, within these add-ons the configuration is handled on the distributed (child) servers.

4.3.3 OpenNMS

OpenNMS (Open Network Management System) [50] is the first enterprise-grade network management and monitoring platform developed under the open source model. It was designed to manage tens of thousands of devices from a single server as well as manage unlimited devices using a cluster of servers. OpenNMS includes a discovery engine to automatically configure and manage network devices without operator intervention. It is published under the GNU General Public License version 3. OpenNMS is known for its scalability, which allows to be used in place of large enterprise environments.

This system is written in Java, and thus can run on any platform with support for a Java SDK version 1.6 or higher. Precompiled binaries are available for most Linux distributions, Windows, Solaris and OS X. In addition to Java, it requires the PostgreSQL database, although work is being done to make the application database independent.

OpenNMS has four main functional areas as described below.

- **Event Notifications and management**: it can receive events in the form of SNMP Traps, Syslog messages, TL/1 events or custom messages sent as XML to port 5817. Events can be configured to generate alarms and other triggered events, such as when an alarm is escalated in severity. Events can generate notifications via e-mail, SMS, XMPP and custom notification methods. From the literature, OpenNMS has been shown to be able to process 125,000 syslog messages per minute continuously. OpenNMS has a trouble ticketing API that allows it to

have bidirectional communication with many popular trouble ticketing systems.

- **Discovery and Provisioning**: it contains an advanced provisioning system for adding devices to the management system. This process can occur automatically by submitting a list or range of IP addresses to the system (both IPv4 and IPv6). It is asynchronous for scalability, and has been shown to provision networks of more than 50,000 discrete devices as well as networks of single devices with over 200,000 virtual interfaces.
- **Service Monitoring**: the types of monitors span from the very simple (ICMP pings, HTTP and DNS req., TCP port checks) to the complex (Page Sequence Monitoring, Mail Transport Monitor). For a complete view of the base monitoring modules see [49]. Outage information is stored in the database and can be used to generate availability reports. In addition to being able to monitor network services from the point of view of the OpenNMS server, remote pollers can be deployed to measure availability from distant locations.
- **Data Collection**: it perform data collection by using a number of network protocols including SNMP, HTTP, JMX, WMI, XMP, XML, NSClient, and JDBC. Data can be collected, stored, graphed as well as checked against thresholds.

4.4 FI-WARE GEs Management Tools

By the term "management" of the Generic Enablers, in the context of this document, we mean (a) the deployment and configuration of the GEs and (b) their monitoring. In this Section we deal with the first part, which is based on the PaaS (Platform as a Service) Manager and the SDC (Software Deployment and Configuration) GEs of FI-WARE.

The PaaS Generic Enabler is based on the outcomes of the 4CaaSt (FP7-257928) project [35], a European Commission FP7 funded Integrated Project started on June 1st 2010. Some of the baseline assets are the following:

- The Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies.
- The Spring Framework is an open source application framework and Inversion of Control container for the Java platform.
- Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database.
- The PostgreSQL is open source object-relational database system.
- The Opscode Chef is an open-source systems integration framework built specifically for automating the cloud.
- PaaS Management Platform is an advanced PaaS Management toolkit that allows to the users the facility to manage their applications without worrying about the underlying infrastructure of virtual resources (VMs, virtual networks and virtual storage) required for the execution of the application components.

The main functionalities that the PaaS Manager of FI-WARE provides are:

- Management of Application Environments, which involve the provisioning and configuration of IaaS resources (eventually including NaaS features), and installation, configuration and management of the Products Instances (PIs) required for the application components to be deployed.

- Management of Application Components (ACs) (lifecycle and configuration) with the help of SDC GE for the installation and configuration of ACs.

The picture below (Figure 16), provided by the architecture description of FI-WARE, represents the architecture of the PaaS Manager [19]. The PaaS consists of three layers and a persistence mechanism: first: the implementation of the PaaS interface, second the set of managers in charge of executing the tasks, and the clients to interact with other GEs (SM GE and SDC GE). By using these GEs, the PaaS Manager GE orchestrates the creation of different VDCs for different customers, and the provisioning of VMs or servers and deployment of the corresponding software products over which the applications can be deployed.

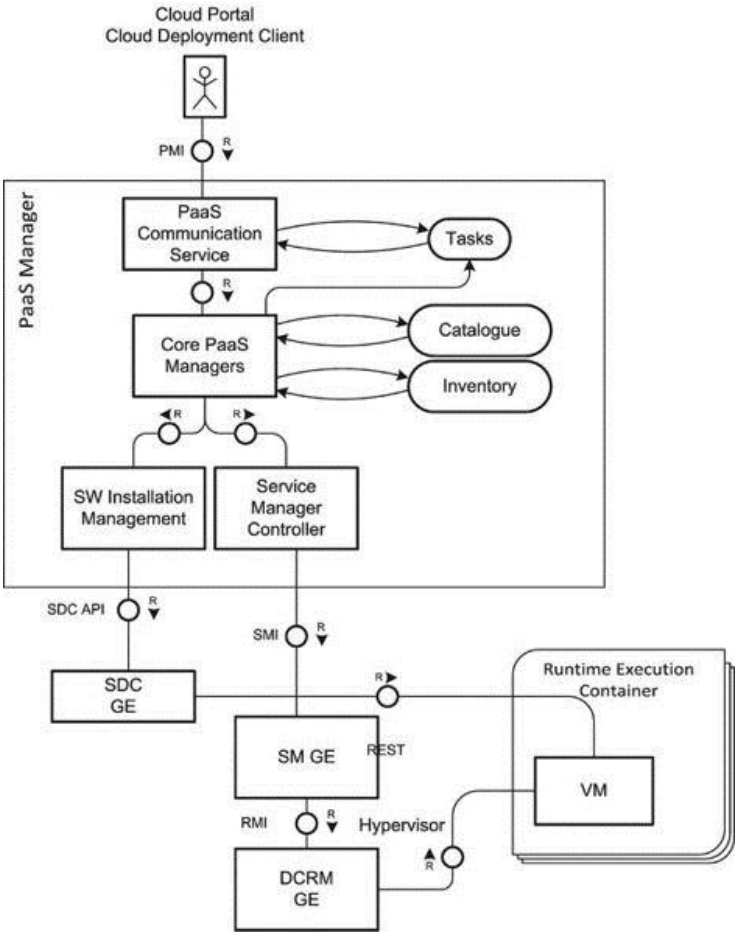


Figure 16: PaaS Manager Architecture [19]

The Software Deployment and Configuration (SDC) GE [20] is the key enabler used to support automated deployment (installation and configuration) of software on running virtual machines. As part of the complete process of deployment of applications, the aim of SDC GE is to deploy software product instances upon request of the user using the SDC API or through the Cloud Portal, which in turn uses the PaaS Manager GE. After that, users will be able to deploy artefacts that are part of the application, on top of the deployed product instances.

Figure 17 shows the main components of the Service Deployment and Configuration GE (SDC GE).

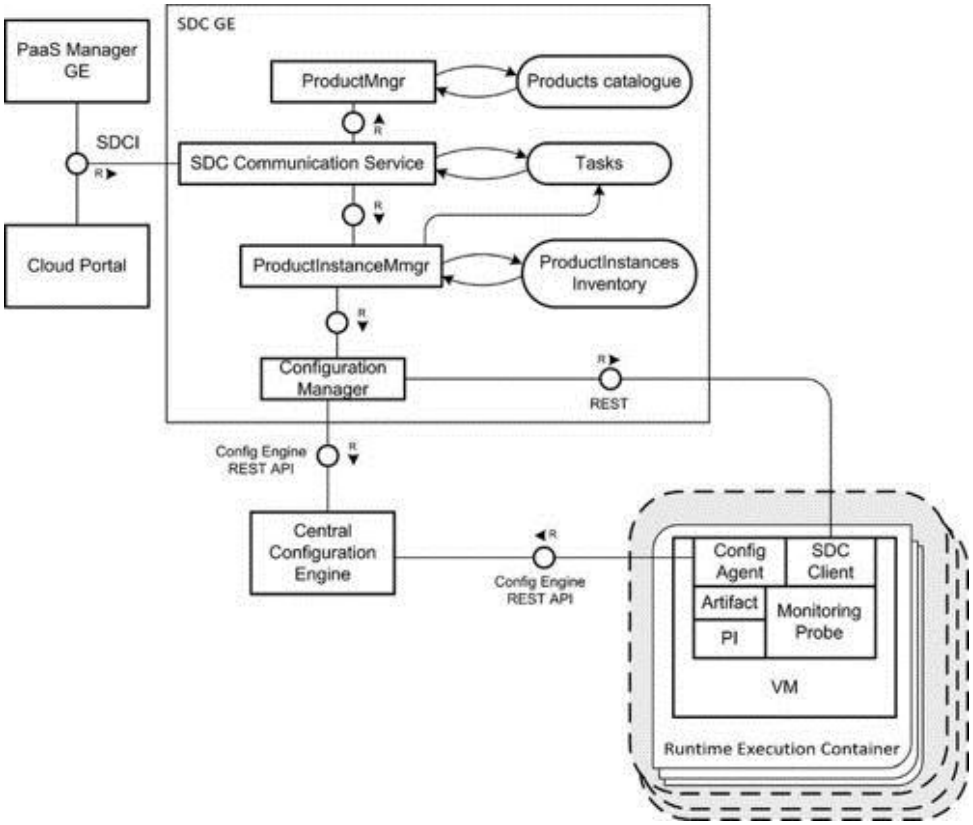


Figure 17: SDC Architecture [20]

The PaaS Manager API is a RESTful, resource-oriented API accessed via HTTP/HTTPS that uses XML-based and/or JSON-based representations for information interchange that provide management of software in the Cloud. The PaaS Manager offers some mechanisms for creating environment for deploying application. The environment involves both the virtual machines and the software required.

The API consumer must indicate the resource identifier while invoking a GET, PUT, POST or DELETE operation. PaaS Manager API combines both identification and location by terms of URL. Each invocation provides the URL of the target resource along the verb and any required input data. That URL is used to identify unambiguously the resource. For HTTP transport, this is made using the mechanisms described by HTTP protocol specification as defined by IETF RFC-2616.

PaaS Manager API entities provide an instance identifier property (instance ID). This property is used to identify unambiguously the entity but not the REST resource used to manage it, which is identified by its URL as described above (although this URL can contain the instance ID). It is common that most implementations make use of instance ID to compose the URL (e.g., the book with instance ID 1492 could be represented by resource *http://.../book/1492*), but such an assumption should not be taken by API consumer to obtain the resource URL from its instance ID.

For further analysis, check FI-WARE Open API Specification of PaaS [19].

4.4.1 Cloud IaaS Deployment tools

FI-WARE developed the SDC tool to support GE installation. So far, nevertheless, there is no plan to support the deployment of IaaS DCRM and related tools in a similar manner. This is of primary importance to build the so called infrastructure toolbox. A number of tools exist, to support deployment of IaaS solutions, some are OpenStack [51] focused and other are IaaS solution independent.

In order to provide a manageable and flexible toolbox which gives the ability of managing and controlling the deployment of services across the datacenter, we have analyzed the following products:

- Fuel: a tool focused on OpenStack provided by Mirantis [34].
- Tuskar: an open source project tailored for OpenStack deployments [65].
- Foreman: an independent solution that covers the complete lifecycle management tool for physical and virtual servers [23].
- Rackspace Private Cloud: a tool that enables users to deploy a private cloud OpenStack cluster configured according to the recommendations of Rackspace OpenStack specialists [60].

An OpenStack cloud installation consists of many packages from different projects, each with its own requirements, installation procedures and configuration management.

The scope of these tools is to help the cloud environments administrators to organize and manage a large variety of infrastructures and services, thus to have a systematic deployment and maintenance process.

4.4.1.1 Fuel for OpenStack

Fuel works using Puppet [59], an IT automation software (it provides, through appropriate scripts, an easy way to automate repetitive tasks). It automates provisioning and deployment of all core OpenStack components including Nova, Glance, Horizon, Swift, Keystone, Neutron/Quantum and Cinder.

In last release 3.1, Mirantis provides an OpenStack distribution onto Red Hat Enterprise Linux or CentOS powered nodes. Ubuntu distribution support is not currently available in the latest version of Fuel, but in the release plan of Mirantis, the support for Ubuntu will be reintroduced by the end of the year 2013.

Fuel it is composed by the Fuel Master node (where are installed all necessary modules as the provisioning agent, the web user interface, the nodes discovery agent), the Puppet scripts and the OpenStack packages.

In practice, after the setup of the Fuel Master Node, the cloud infrastructure administrator can discover his virtual or physical nodes and configure his OpenStack cluster using the Fuel UI. Finally, he deploys his OpenStack cluster on discovered nodes. Fuel will perform all deployment process by applying pre-configured and pre-integrated Puppet manifests.

One of the advantages of Fuel is that it comes with a number of pre-built deployment configurations that can be used by users to build their OpenStack cloud infrastructure. These are well-specified configurations of OpenStack in according to the best practices recommended by OpenStack specialists.

4.4.1.2 Tuskar

As Fuel by Mirantis, also Tuskar provides, to cloud environments administrators, the ability to control the OpenStack deployment across their datacenters.

With Tuskar, administrators model their cluster of nodes into "resource classes". Every resource class is composed by a service type, more rack elements (where a rack models is a set of nodes, capacities and network properties) and one or more flavours (where a flavour is a virtual machine template).

Tuskar services are available via a RESTful API or via management console, through which administrators are able to classify their hardware and define their datacenters. In addition, Tuskar provides a set of functionalities for performance monitoring, health statistics, and usage metrics. It aims to support and optimize the architecture designer and administrators of cloud environments in them decisions.

The lack of Tuskar is that it is in an early development phase and it does not provide a stable version.

4.4.1.3 Foreman

Foreman, is a tool, independent from OpenStack installation, which manages every phase of the lifecycle management of physical and virtual servers. It contains a collection of Puppet modules and it uses native OS packaging (e.g. RPM and .deb packages).

A Foreman installation, similarly to Fuel, contains a central Foreman instance which is responsible for providing the Web based GUI, node configuration and initial host configuration files.

Foreman is able to manage multiple subnets, DNS zones, Puppet masters from a single instance using the Smart Proxy mechanism. A smarty proxy is a software module which offers through APIs the following services: DHCP, DNS, TFTP, Puppet and Puppet agent.

4.4.1.4 Rackspace Private Cloud

Rackspace offers a tool set that enables users to quickly deploy a private cloud OpenStack cluster configured according to the recommendations of Rackspace OpenStack specialists. Rackspace Private Cloud enables users to create an OpenStack cluster on Ubuntu, CentOS, or RHEL, using a set of installation scripts.

This tool supports integration with some OpenStack components, as such as floating IP address management, security groups, availability zones, and the python-novaclient command line client, but it does not support some functionality as such as Nova Object Storage, Nova Volumes and centralized metadata servers.

5 NETWORK-BASED SERVICES ADAPTERS

5.1 Architecture Description

XIFI proposes an SDN network service provisioning design following the ABNO (Application Based Network Orchestration) [1] and OpenNaaS [48] architectures. The coordination between these two entities results crucial for a suitable implementation of the overall XIFI connectivity service. Figure 18 represents the modular architecture of the XIFI Network Controller including OpenNaaS (green color) and the specific components of the ABNO framework (red color) considered in XIFI.

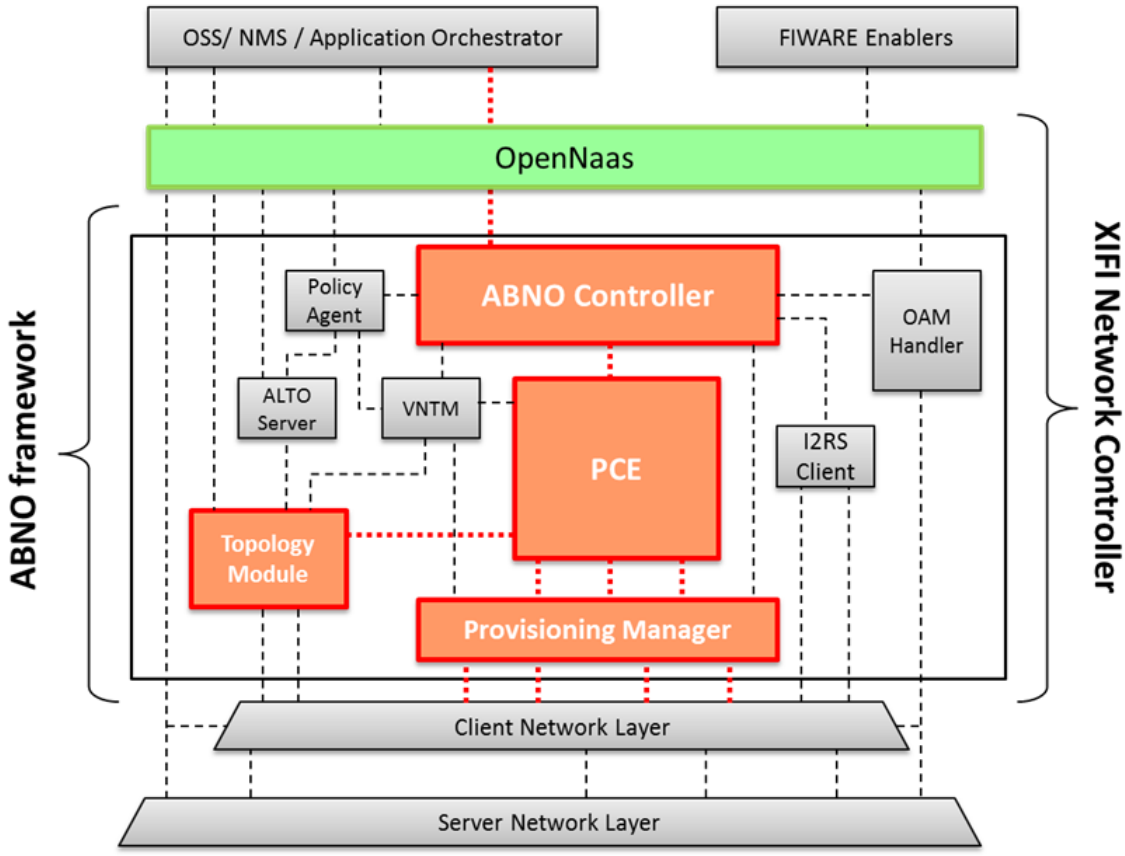


Figure 18: XIFI Network Controller with OpenNaaS and ABNO components

OpenNaaS is an open-source framework, developed under the FP7 MANTYCHORE project [33] which provides with a suitable set of tools for the management and operation of network services, such us the XIFI network connectivity service.

In XIFI, the OpenNaaS framework provides the overall management of the connectivity service while keeping a logical view of the network thanks to the abstracted full-network topology orchestration provided by ABNO. Thus, OpenNaaS is flexible enough to be applied to different underlying control systems such as OpenFlow-based or in the XIFI case ABNO.

The OpenNaaS framework is characterized by the implementation of virtual resources that represent manageable units within the network. A resource can be (among other things) a switch, a router, a link, a logical router, or a network, and each resource is composed of one or more capabilities. A capability is an interface to a given resource functionality (e.g. OSPF configuration in router resource). OpenNaaS can be divided in three layers, as can be seen in Figure 10: Platform, NaaS layer and Network intelligence. The platform layer implements the common and reusable components; it provides the application context and defines its technological framework, including communication channels to access OpenNaaS.

The NaaS layer reuses platform defined components to expose user manageable resources. This layer offers support for concrete resource types, functionalities (called capabilities) and device types. Resources and capabilities are exposed to the user, abstracted from implementation details and from hardware specifications. Hence, this layer acts as a Hardware Abstraction Layer.

The Network Intelligence Layer is built on top of the abstraction provided by the NaaS layer, to enable the management of the virtual resources. Here, management policies may be defined and actions based on them are applied to the network using OpenNaaS. This layer also implements plugins for accessing OpenNaaS externally.

To access the services that OpenNaaS provides, RESTful web service interfaces expose the resources under particular users' control so that network service management capabilities can be delegated to users/experimenters using the XIFI connectivity service. This way OpenNaaS performs user access management, keeping a restricted view of the provided network services based on credentials.

Additionally, OpenNaaS implements a Neutron (Quantum) plugin to provide the associated network service and allow its request directly through OpenStack [51]. This will enable the coordination of the required inter-DC network service and its management so that an automatic complete inter-DC connectivity solution can be achieved. Figure 19 shows the detail of the interconnection between OpenStack and OpenNaaS and the different modules and capabilities that shape the system.

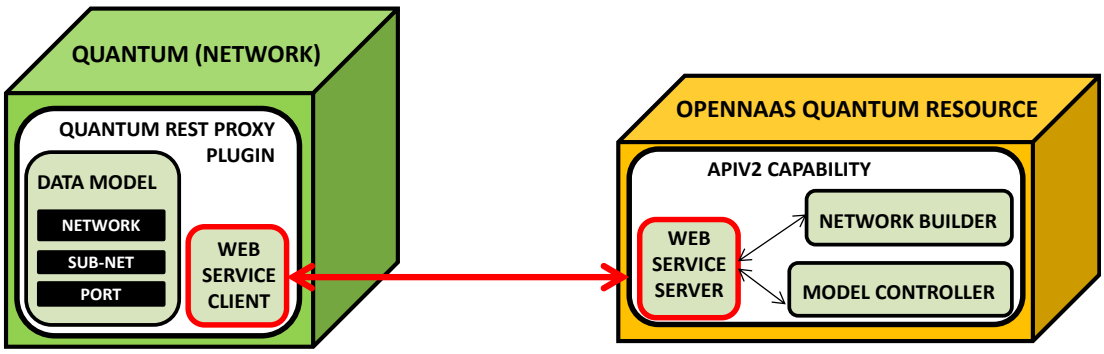


Figure 19: Detail of the Quantum plugin and its system modules

Finally, OpenNaaS will implement the required interface to request the network connectivity through ABNO. This interface will specify the endpoints to be connected and the characteristics of the required connectivity.

The ABNO architecture [1] has been proposed in IETF as an SDN framework based on standard building blocks. The set of standard functional blocks includes elements such as the Patch Computation Element (PCE). The components of the ABNO framework permit a number of functions to act in an integrated way, including policy control, topology information, and network resource provisioning.

The ABNO architecture is currently being defined in the IETF. A number of interfaces between the distinct modules are standard. However, there are some modules whose interfaces are not already defined. Generically, HTTP/JSON interfaces will be used in these interfaces. The main reason for this is to facilitate an easy development and to provide flexibility for the workflows definition. These interfaces will help to have a modular design, which can be adapted to the future requirements that may come during the project.

The ABNO Controller is responsible of orchestrating the connectivity service requests by invoking the necessary components in the right order according to specific workflows. It is connected to the OpenNaaS component from where the connectivity request is progressed. The ABNO Controller will offer an interface to OpenNaaS for receiving connectivity requests or to trigger any other workflow using HTTP/JSON. This interface is not yet defined in IETF. The parameters of the request could vary depending on the workflow, but the operation type is always mandatory.

This would be an example of an IP Link provisioning request:

```
{
    "ID_Operation":"15697"
    "Operation Type": "IP_LINK_PROVISIONING",
    "Source Node": "10.95.73.72",
    "Destination Node": "10.95.73.74",
    "Interfaces ID": ["70.70.70.1","70.70.70.2",]
    "IGP ID": "100"
}
```

Once the path has been configured the ABNO controller sends a notification response to the NMS:

```
{
    "ID_Operation":"15697"
    "Operation Type":"IP_LINK_PROVISIONING",
    "Source Node": "10.95.73.72",
    "Destination Node": "10.95.73.74",
    "Result": "IP_LINK_CONFIGURED"
    "Error Code": "NO_ERROR"
}
```

The Path Computation Element (PCE) is in charge of the necessary computations to find connectivity between network elements based on the topological and traffic engineering information, applying a wide set of constraints, including e.g. bandwidth, QoS, exclusions. The PCE architecture supports coordination among several PCEs, with different responsibilities is envisioned and is especially useful for multi-domain environments, in which each domain is independently administrated, or multi-technology networks, in which switching can be performed using different technologies, such as Ethernet or optical. As mentioned, the PCE relies on topological and traffic engineering information. In that sense, the Topology Module is in charge of discovering the network resources and their availability. The topology module is able to interface with third party tools, SNMP, NetConf, IGP (OSPF-TE, ISIS-TE), BGP, OpenFlow or REST/JSON.

There is an interface between the ABNO controller and the PCE. The ABNO controller queries the PCE using PCEP to determine what services can be provisioned. As there are stateless and stateful PCEs, this interface will support requests for both PCEs.

The Provisioning Manager is an element in charge of controlling the devices of the network to create the desired connectivity. There are two main approaches that can be followed. On the one hand, there can be a trigger to the control plane (e.g. GMPLS) and rely on it for the necessary provisioning. On the other hand, there can be a programming of each individual network element involved in the connection. The provisioning manager receives the requests with a standard format that is able to cover the different network technologies. I order to control the devices (either relying on a control plane or configuring the devices themselves), standard interfaces are used as a preferred option. Among the standard interfaces, Netconf and PCEP are candidate for option like OpenFlow, ForCES, GSMP and, again, Netconf suit for the second option. In XIFI, the implementation is targeted at using OpenFlow as the configuration interface.

There will be an interface among the Provisioning Manager and the ABNO Controller. The requests will be channelled through PCEP. There will be also a logic interface among the Provisioning Manager and the PCE. In case the PCE is stateless, this interface does not apply, actually. If the PCE is stateful and active, the interface is used to act on existing connections and if the PCE is stateful, active PCE with instantiation capabilities, the PCE can request the establishment of new connections using it.

5.1.1 Multisite Orchestration

This use case addresses a scenario where servers (or virtual machines running on those servers) which are hosted in different separated datacenters require to be logically located on the same Layer2 network. While the target for this scenario is to focus on an easier way provisioning of services, some other features like SLAs enforcement, QoS guarantees, or security could be also explored.

The conventional approach for connecting computing resources to the network uses network segmentation based on VLANs to separate end-user or tenant services. This way of segmenting the network provides a limited number of configurable networks per DC (determined by the theoretical 4096 available different VLAN tags), leading to a careful planning of resources either locally or remotely, when more than one DC is involved in a service. This approach has the burden of having to manually reconfigure multiple switches and routers every time a new service has to be deployed, creating inefficiencies and costs.

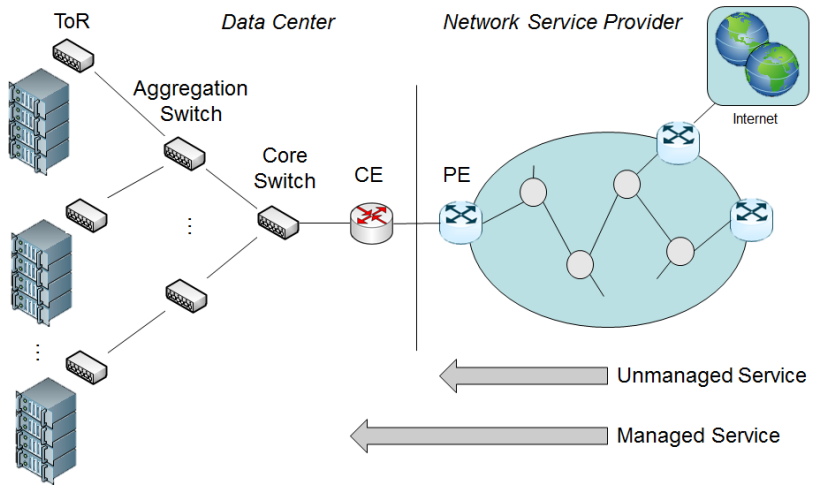


Figure 20: Current ways of connecting DCs

There is a lack of an easy translation of business-level requirements (in terms of service connectivity and associated KPIs) into the provision of the necessary network resources to provide them. The proposal here is to use a Carrier SDN framework to arrange, coordinate and manage the automated deployment of the traffic forwarding rules and the configuration of additional network resources for the service composition. To that end, some abstraction capabilities are needed to facilitate the programmability of the network in an agnostic way regarding the specific underlying infrastructure, allowing a rapid and dynamic response to application needs, changes in network conditions, and business policy enforcement. Multisite L2 services can be built under the concept of virtual patch-panels. In each location, ports on multiple switches (spread across the network) can be programmatically connected among them to set up extended point-to-point connections, in a dynamic and automated way. It is also important to implement mechanisms capable of providing a dynamic on-demand scaling (up and down) of the resources offered to those services, and capable of automatically propagating any network change that could affect those services.

The mentioned programmable switches (or set of switches) in each DC will act as service demarcation points. These switches will offer OpenFlow-based interfaces to make possible their programmability.

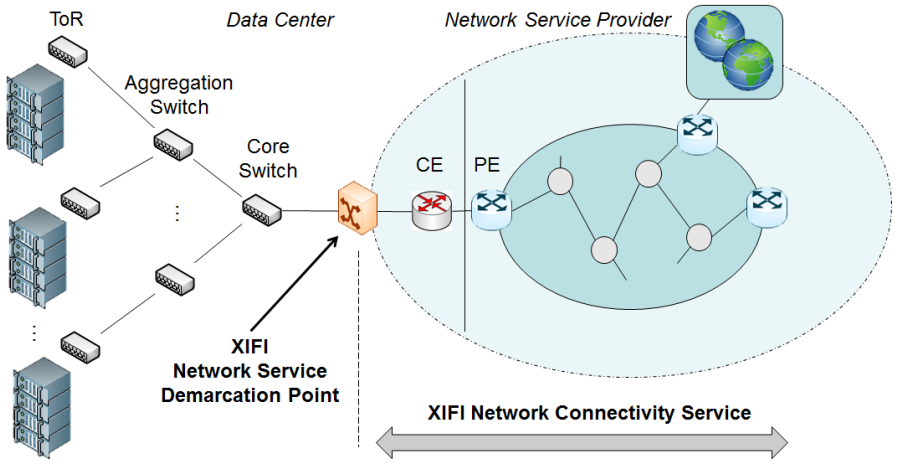


Figure 21: Proposed demarcation point for connecting DCs in a Carrier SDN framework

This logically centralized Carrier SDN framework suites well for the control of autonomous network environments, like the one formed by the border nodes acting as service demarcation points for each DC. By offering a connectivity service API on top of the ABNO controller, it is possible to enable the construction of such L2 overlay networks to interconnect the Datacenters independently of the virtualization technology in use by the cloud provider internally to the DCs (e.g. OpenStack or manual configuration). Furthermore, the adoption of standardized elements and protocols helps to achieve solution robustness and future-proof multi-vendor interoperability.

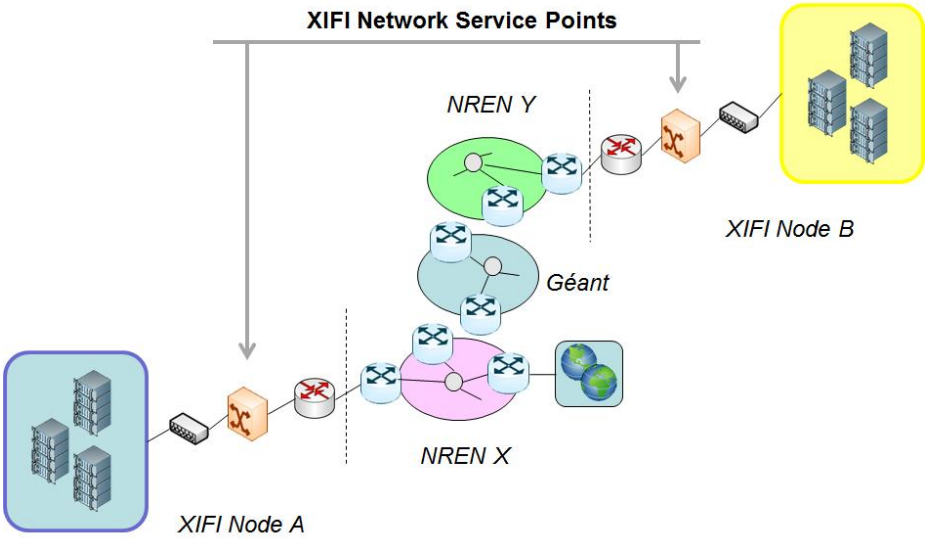


Figure 22: End-to-end service view of a point-to-point multisite L2 connection

The virtual patch panel function will consist on a stitching of the per-user generated VLAN in a DC with the corresponding pre-provisioned connection that connects with the remote DC. The same is done in the other end to build the end-to-end point-to-point layer2 connection. The interconnection between DCs will consist on a (full or not) mesh of tunnels among the DC locations, so different paths can be selected from a centralized controller to accommodate the end-user flows according to their needs, in the most efficient way (e.g., avoiding extra costs due to the use of a more expensive direct link, or using premium/gold, silver/bronze paths depending on the application's service). Some pre-provision and some pre-engineering will be required for having the (full or not) mesh of interconnections between the sites where the DCs are located, emulating a controlled point-to-point pipe passing through the border routers on each DC. For instance, if it is possible to route NVGRE ((Network Virtualization using Generic Routing Encapsulation) packets exiting the OpenFlow switch, then a pure routing connection to the remote DC would be enough. The same applies for Q-in-Q if the service provider is capable of providing a L2 transport connection between the sites end-to-end.

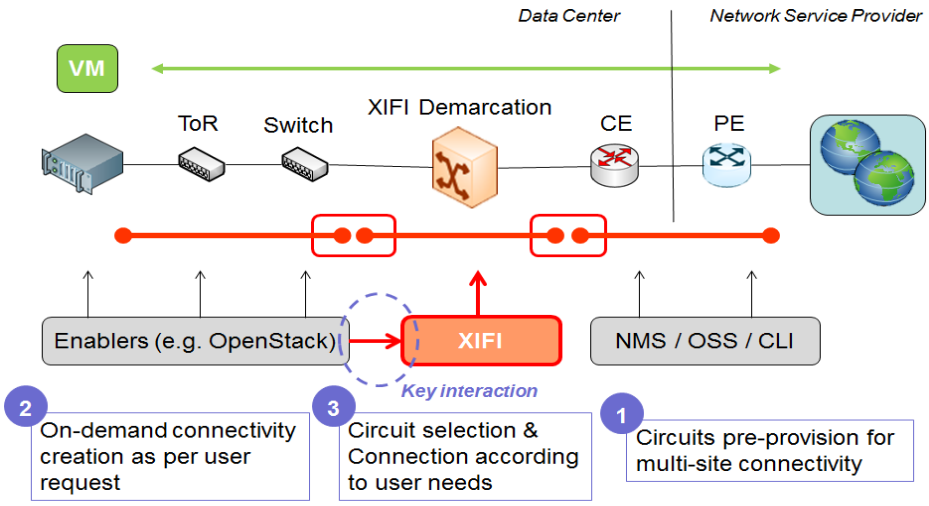


Figure 23: Service stitching in a virtual patch panel

From the data plane point of view, the OpenFlow switch acting as demarcation point will forward the tenant traffic to the network service provider router. The capabilities for handling the traffic by the demarcation points are determined by the capabilities that such switch can provide. According to the most recent OpenFlow specification (v1.3.2, from April 2013), the following actions could be allowed: Push / PoP of VLAN header; Push / PoP of MPLS header; and Push / PoP of PBB header. Additionally, the OF-Config protocol (last spec v1.1.1 is from March 2013) allows some tunnelling features that according to the documentation are IP-in-GRE, NVGRE and VxLAN. Some additional or more complex tunnelling scenarios could be worked out if combined with some engineering work in the border router of the network service provider.

The ABNO controller will dynamically instruct the OpenFlow switch by modifying the traffic flow tables to take the actions needed to prepare the frames before delivering them to the border router. The rules for preparing the frames are known in advanced as the border routers have been previously pre-configured to process the traffic in a certain way (e.g., in Q-in-Q case, the required tags for each traffic have been configured in the border router for dealing with that traffic).

5.1.2 XIFI Overlay tunnelling

5.1.2.1 Layer-2 Connectivity between DCs

Figure 24 schematically represents the different frame formats that could be in place in a layer-2 inter-datacenter connectivity service.

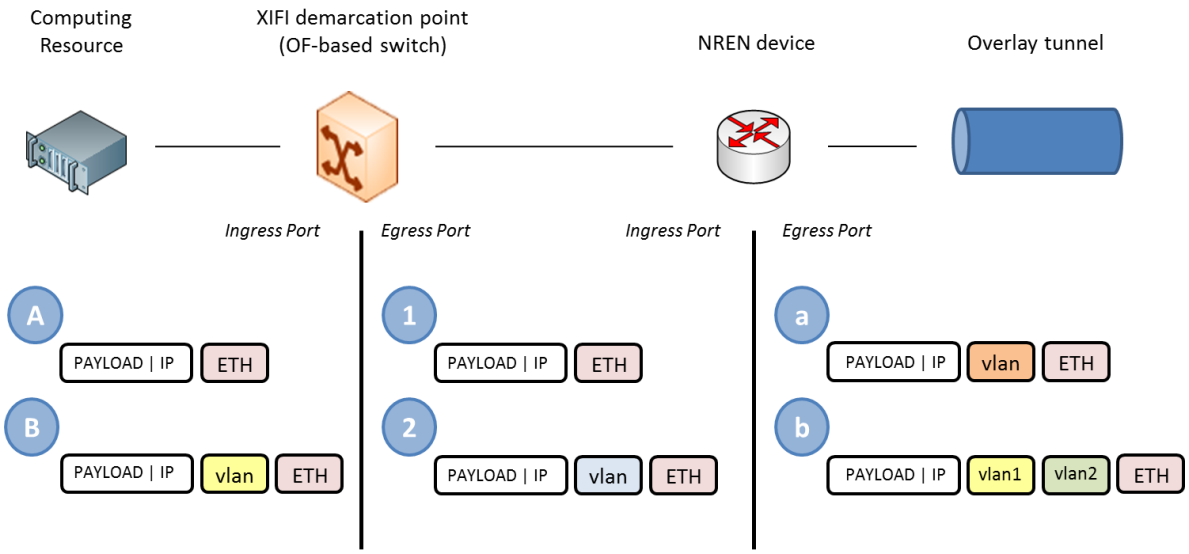


Figure 24: L2-connectivity scenario

The requirements for each of the possible scenario are summarized in the table below.

Transport Scenario	Requirements on XIFI device	Requirements on NREN device	Overlay Tunnel
A.1.a	Egress port assignment based on MAC destination addresses (separated egress port per destination)	Separated ingress port per XIFI DC destination. Vlan assignment based on ingress port.	L2
A.1.b	Egress port assignment based on MAC destination addresses (separated egress port per destination)	Separated ingress port per XIFI DC destination. Q-in-Q assignment based on ingress port.	Q-in-Q
A.2.a	Vlan assignment based on MAC destination address.	Vlan assignment based on vlan at ingress port.	L2
A.2.b	Vlan assignment based on MAC destination address.	Q-in-Q assignment based on vlan at ingress port.	Q-in-Q

B.1.a	Egress port assignment based on vlan tag (separated egress port per destination)	Separated ingress port per XIFI DC destination. Vlan assignment based on ingress port.	L2
B.1.b	Egress port assignment based on vlan tag (separated egress port per destination)	Separated ingress port per XIFI DC destination. Q-in-Q assignment based on ingress port.	Q-in-Q
B.2.a	Vlan assignment based on MAC destination address.	Vlan assignment based on vlan at ingress port.	L2
B.2.b	Vlan assignment based on MAC destination address.	Q-in-Q assignment based on vlan at ingress port.	Q-in-Q

Table 1: Layer-2 Connectivity requirements

5.1.2.2 Layer-3 Connectivity between DCs

Figure 25 schematically represents the different frame formats that could be in place in a layer-3 inter-datacenter connectivity service.

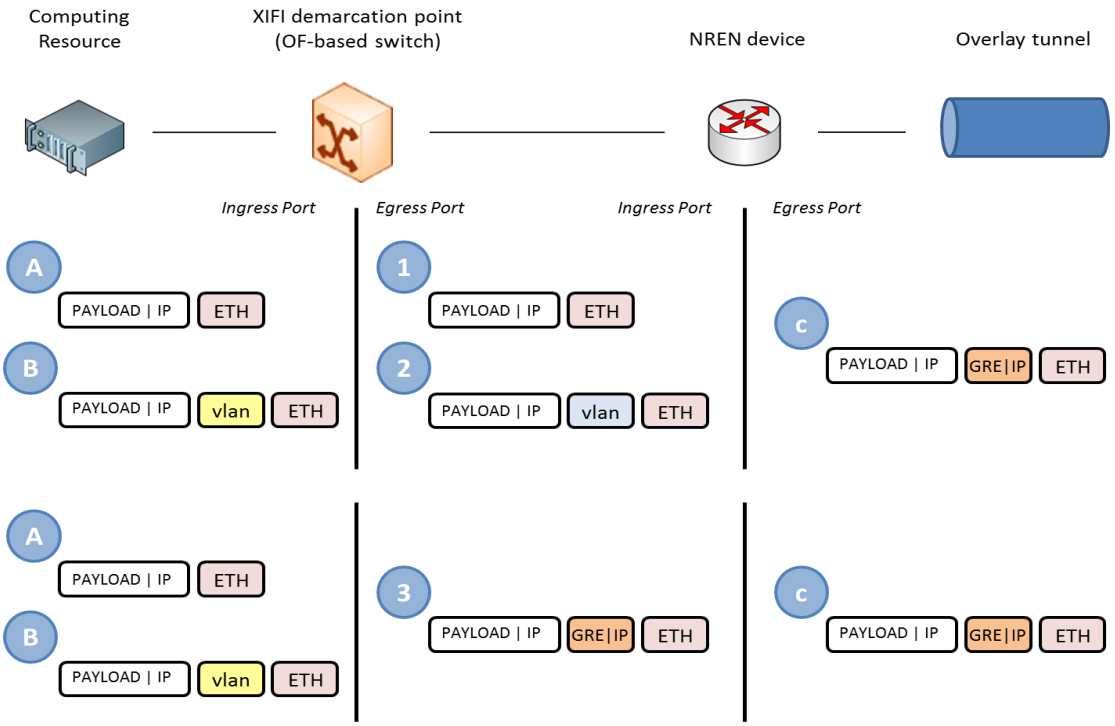


Figure 25: L3-connectivity scenario

The requirements for each of the possible scenario are summarized in the table below.

Transport Scenario	Requirements on XIFI device	Requirements on NREN device	Overlay Tunnel
A.1.c	Egress port assignment based on MAC destination addresses (separated egress port per destination)	Separated ingress port per XIFI DC destination. GRE/IP address based on ingress port.	L3
A.2.c	Vlan assignment based on MAC destination address.	GRE/IP address based on vlan at ingress port.	L3
A.3.c	GRE/IP address based on MAC destination address (it implies OF-config configuration in separated egress ports)	Routing based on outer tunnel destination IP address.	L3
B.1.c	Egress port assignment based on vlan tag (separated egress port per destination)	Separated ingress port per XIFI DC destination. GRE/IP address based on ingress port.	L3
B.2.c	Vlan assignment based on MAC destination address.	GRE/IP address based on vlan at ingress port.	L3
B.3.c	GRE/IP address based on MAC destination address (it implies OF-config configuration in separated egress ports)	Routing based on outer tunnel destination IP address.	L3

Table 2: Layer-3 Connectivity requirements

5.1.3 Target Connectivity Scenarios

Distinct tenant's VM connectivity scenarios are envisaged as potential connectivity scenarios demanded to XIFI. The scope of XIFI is to progressively provide these connection capabilities starting from the more simplistic scenario of connecting two VMs residing in two separated DCs. The possibility of providing different scenarios will depend on the capabilities in place either in the specific XIFI devices in each DC or the NREN infrastructure providing the external connectivity.

The foreseen connectivity scenarios are:

- Private network spanning multiple nodes between two DCs

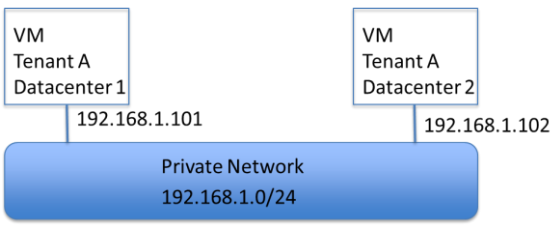


Figure 26: Connectivity scenario 1

- Private sub-networks in two DCs linked with dedicated router

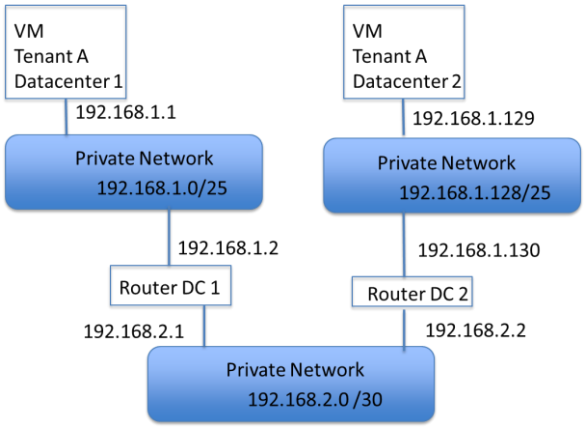


Figure 27: Connectivity scenario 2

- Private network spanning multiple nodes between two DCs, and shared network per DC

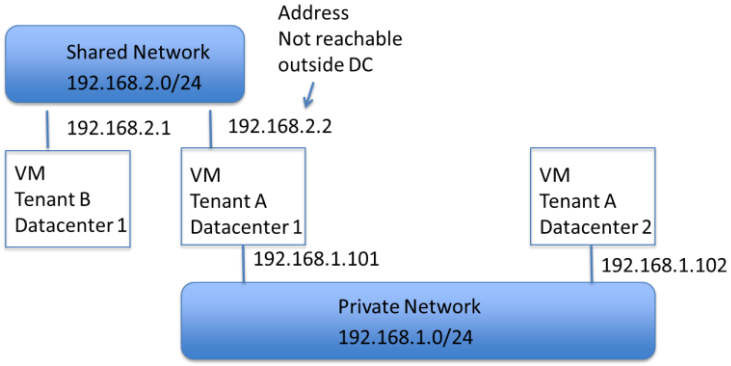


Figure 28: Connectivity scenario 3

- Private network spanning multiple nodes between two DCs, and shared network with public IPs per DC

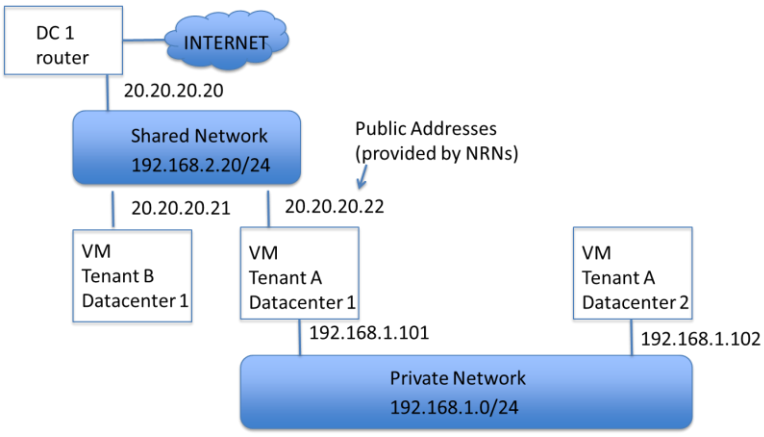


Figure 29: Connectivity scenario 4

5.1.4 XIFI SDN Controller Deployment Model

As previously stated, the XIFI SDN Network Controller will be based on the OpenNaaS and ABNO frameworks.

The main components of the XIFI controller require running in an infrastructure node with a *master* role, and they shall be accessible through the OpenStack Networking API in order to build virtual networks between hosts (virtual machines). Also, a Topology API is implemented in order to expose details of the XIFI Network Infrastructure.

Within each of the XIFI nodes there will be several processes running, which include a local Topology Module, in charge of getting the topological information from the node, and a local provisioning manager, which transforms the PCEP messages from the controller to OpenFlow Rules. A first version of the provisioning manager has been implemented extending the Flow Pusher of the open source Floodlight controller.

5.2 Scenarios of Use

5.2.1 Scenario 1: Joining/Attaching to the XIFI Federation

Section	Description
Title & Description	Joining/Attaching to the XIFI Federation scenario. This use case describes the scenario of an Infrastructure Joining the XIFI federation.
Author(s)	Ian Mills (WIT)
Actors	• XIFI Infrastructure • Infrastructure Owner
Objective	The main objective for this scenario is to showcase the use of a network based service adapter in the facilitating of the joining/attaching of an Infrastructure into the XIFI federation.
Pre-Conditions	The prerequisites of this scenario are: • The XIFI federation base is deployed and operational. • The Infrastructure is compliant with the operational, legal and technical terms of the XIFI Federation. • The infrastructure joining is compliant with the requirements for Master/Slave Node.
Process Dialog	1. Infrastructure owner connects to the XIFI portal in order to join the federation. 2. Discover attached network resources in the joining infrastructure. 3. Federate discovered networked resources. 4. Add cloud resources into resource catalogue/resource pool. 5. Add new infrastructure owner to Identity federation provided by XIFI federation.

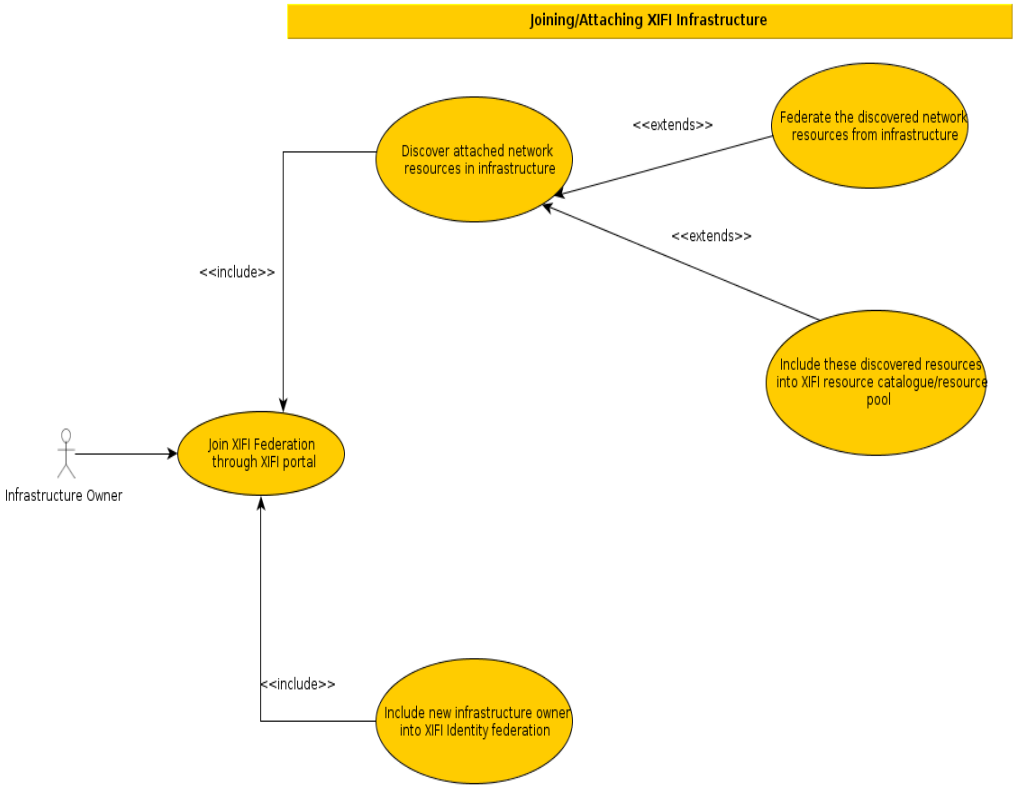
Post-Conditions	Infrastructure has now successfully been connected to the XIFI federation.
Diagrams	**Use Case Diagram** 

Table 3: Network-based Services Scenario 1

5.2.2 Scenario 2: Joining/Attaching a Hybrid Cloud

Section	Description
Title & Description	Joining/Attaching a Hybrid Cloud scenario.
Author(s)	Ian Mills (WIT)
Actors	• XIFI Infrastructure • Infrastructure Owner
Objective	The main objective for this scenario is to showcase the use of a network based service adapter in the facilitating of the joining/attaching of a Hybrid Cloud.

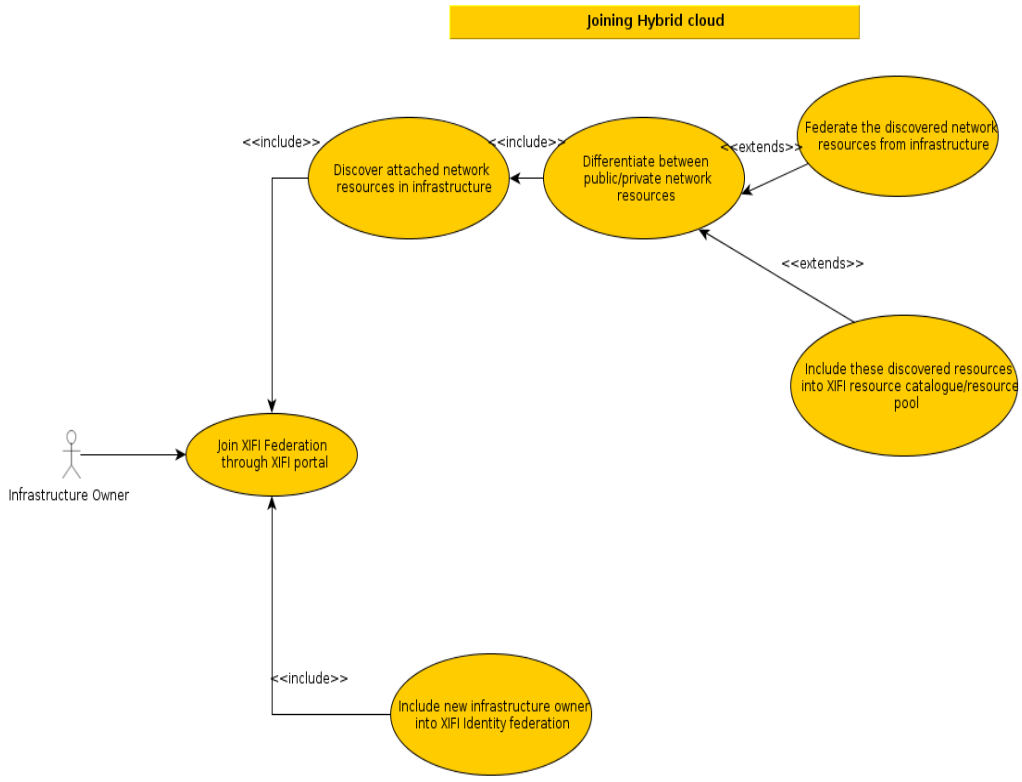
Pre-Conditions	The prerequisites of this scenario are: • The XIFI federation base is deployed and operational. • The Infrastructure is compliant with the operational, legal and technical terms of the XIFI Federation. • The infrastructure joining is compliant with the requirements for Master/Slave Node.
Process Dialog	1. Infrastructure owner connects to the XIFI portal in order to join the federation. 2. Differentiate between public/private network resources. 3. Discover attached network resources in the joining infrastructure. 4. Federate discovered networked resources. 5. Add cloud resources into resource catalogue/resource pool. 6. Add new infrastructure owner to Identity federation provided by XIFI federation.
Post-Conditions	Infrastructure has now successfully been connected to the XIFI federation.
Diagrams	**Use Case Diagram** 
Issues & Notes	Security in public/private cloud

Table 4: Network-based Services Scenario 2

5.2.3 Scenario 3: Network Discovery/Network services on offer

Section	Description
Title & Description	Network Discovery/Network services on offer. This use case describes the scenario of a network based service adapter being used to discover the network services/resources in an infrastructure.
Author(s)	Ian Mills (WIT)
Actors	• XIFI Infrastructure • Infrastructure Owner
Objective	The main objective for this scenario is to showcase the use of a network based service adapter in the facilitating of the discovery of networks and networked services on offer.
Pre-Conditions	The prerequisites of this scenario are: • The XIFI federation base is deployed and operational. • The Infrastructure is compliant with the operational, legal and technical terms of the XIFI Federation. • The infrastructure joining is compliant with the requirements for Master/Slave Node.
Process Dialog	1. Scan network for available resources in the infrastructure. 2. Gather network information metrics from attached infrastructure. 3. Add discovered network resources/network services into XIFI catalogue.
Post-Conditions	A full list of network resources/ services in the XIFI federation entered into XIFI catalogue.

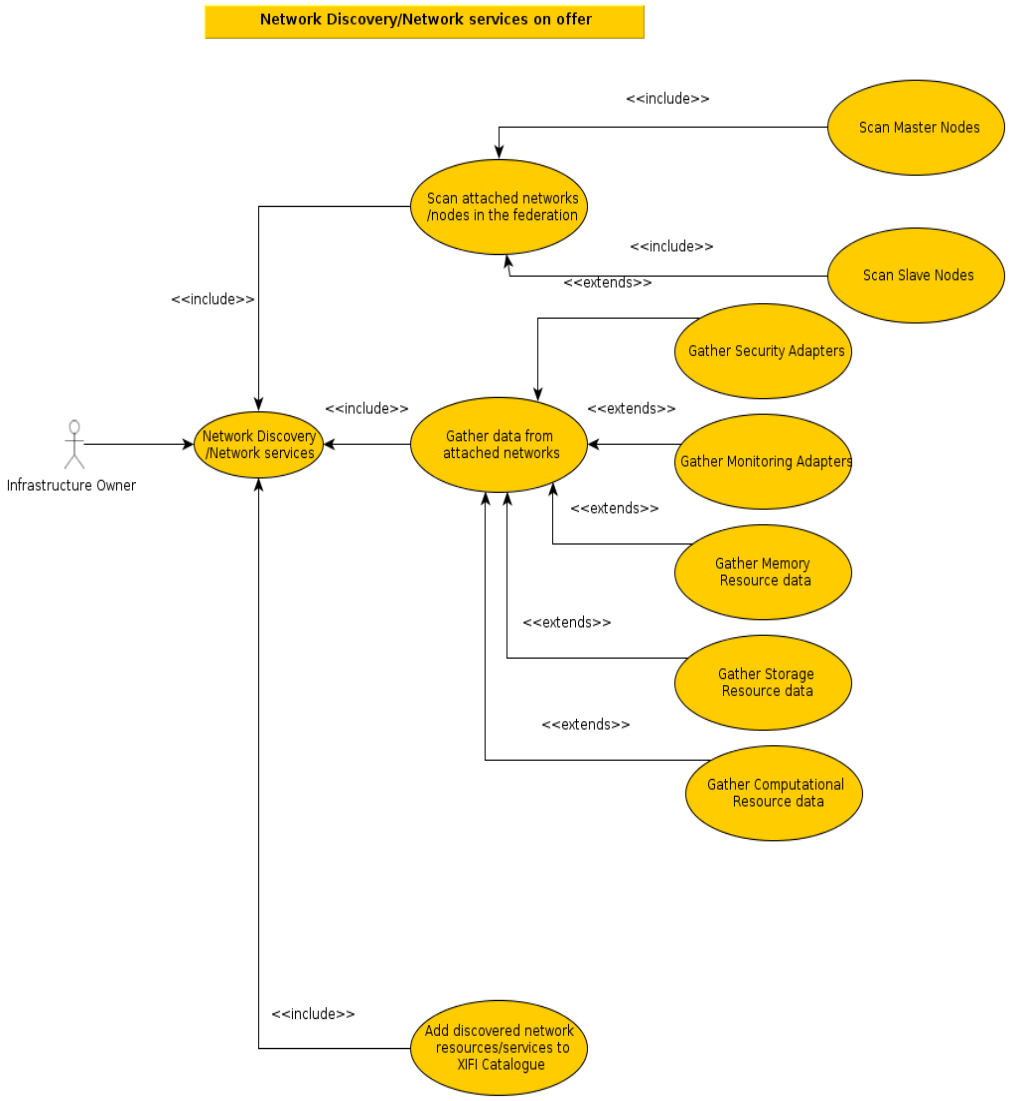
Diagrams	**Use Case Diagram** 
Issues & Notes	Due to security constraints the granularity of the condition of these networks may not be available.

Table 5: Network-based Services Scenario 3

5.2.4 Scenario 4: Network Provisioning

Section	Description
Title & Description	Network Provisioning. This use case describes the scenario of provisioning network resources to a XIFI Infrastructure user/developer
Author(s)	Ian Mills (WIT)

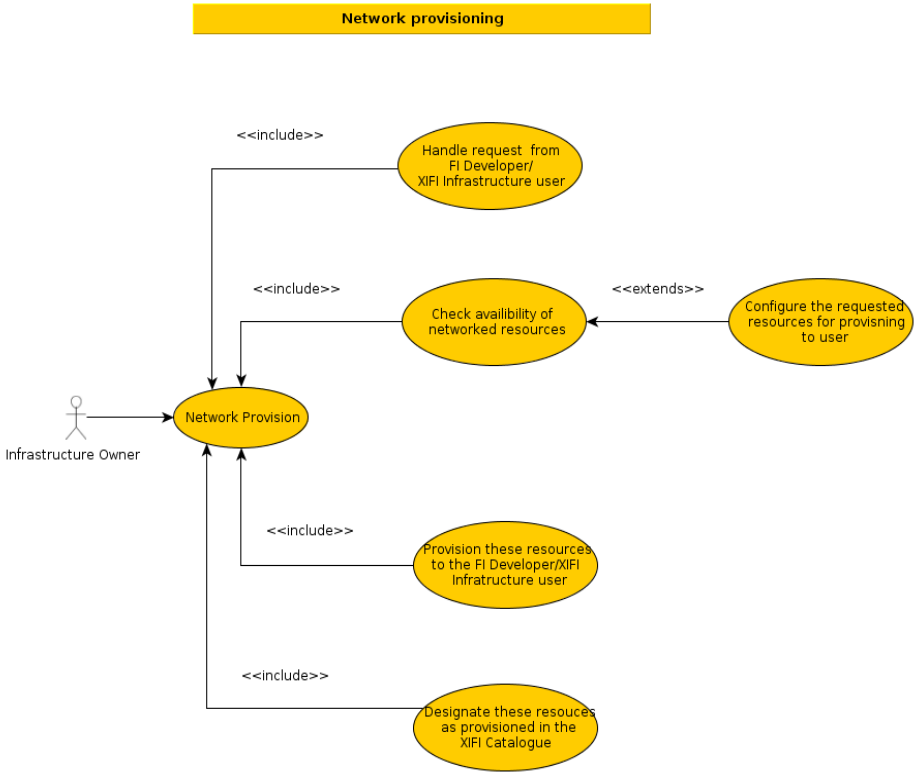
Actors	• XIFI Infrastructure • Infrastructure Owner
Objective	The main objective for this scenario is to showcase the use of a network based service adapter in the facilitating of available network resources and services to a XIFI infrastructure developer.
Pre-Conditions	The prerequisites of this scenario are: • The XIFI federation base is deployed and operational. • The Infrastructure is compliant with the operational, legal and technical terms of the XIFI Federation. • The infrastructure joining is compliant with the requirements for Master/Slave Node.
Process Dialog	1. Handle request from FI Developer/XIFI federation user for resources from XIFI catalogue. 2. Check availability of networked resources in the XIFI catalogue. 3. Configure the requested resources for provisioning to the user/developer. 4. Allocate these resources to FI Developer/XIFI federation user. 5. Designate these provisioned resources as issued in the XIFI catalogue.
Post-Conditions	Network services are provisioned to the FI developer/XIFI federation user.
Diagrams	**Use Case Diagram** 

Table 6: Network-based Services Scenario 4

5.2.5 Scenario 5: Network Monitoring (Trap and event handling)

Section	Description
Title & Description	Network Monitoring (Trap and event handling). This use case describes the scenario of Network Monitoring on the XIFI infrastructure. Specifically in terms of trapping and event handling.
Author(s)	Ian Mills (WIT)
Actors	• XIFI Infrastructure • Infrastructure Owner
Objective	The main objective for this scenario is to showcase the use of a network based service adapter in terms of monitoring the XIFI infrastructure and handling any traps caught and event handling.
Pre-Conditions	The prerequisites of this scenario are: • The XIFI federation base is deployed and operational. • The Infrastructure is compliant with the operational, legal and technical terms of the XIFI Federation. • The infrastructure joining is compliant with the requirements for Master/Slave Node.
Process Dialog	1. Network monitoring in the event of a trap or event. 2. XIMM throws a exception to a trap or event handler alert. 3. Designate the networked resources as unavailable. 4. Enable the graceful degradation of resources and isolate affected regions. 5. Alert FI Developer/XIFI Infrastructure user of issue.
Post-Conditions	In the case of a trap or event thrown.

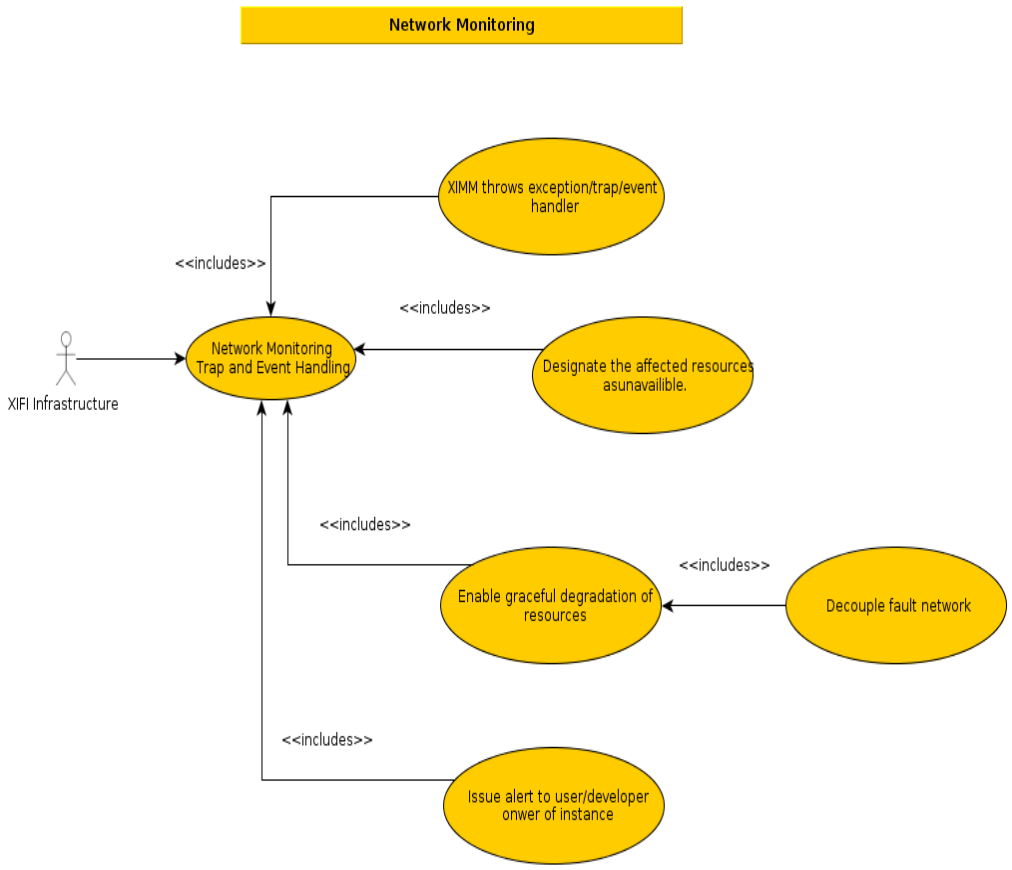
Diagrams	**Use Case Diagram** 

Table 7: Network-based Services Scenario 5

5.2.6 Scenario 6: Network User authorisation

Section	Description
Title & Description	Network User authorisation. This use case describes the scenario of a XIFI Infrastructure user authenticating through the XIFI login portal.
Author(s)	Ian Mills (WIT)
Actors	• XIFI Infrastructure • Infrastructure Owner
Objective	The main objective for this scenario is to showcase the use of a network based service adapter in the facilitating of XIFI Infrastructure user authentication across the network

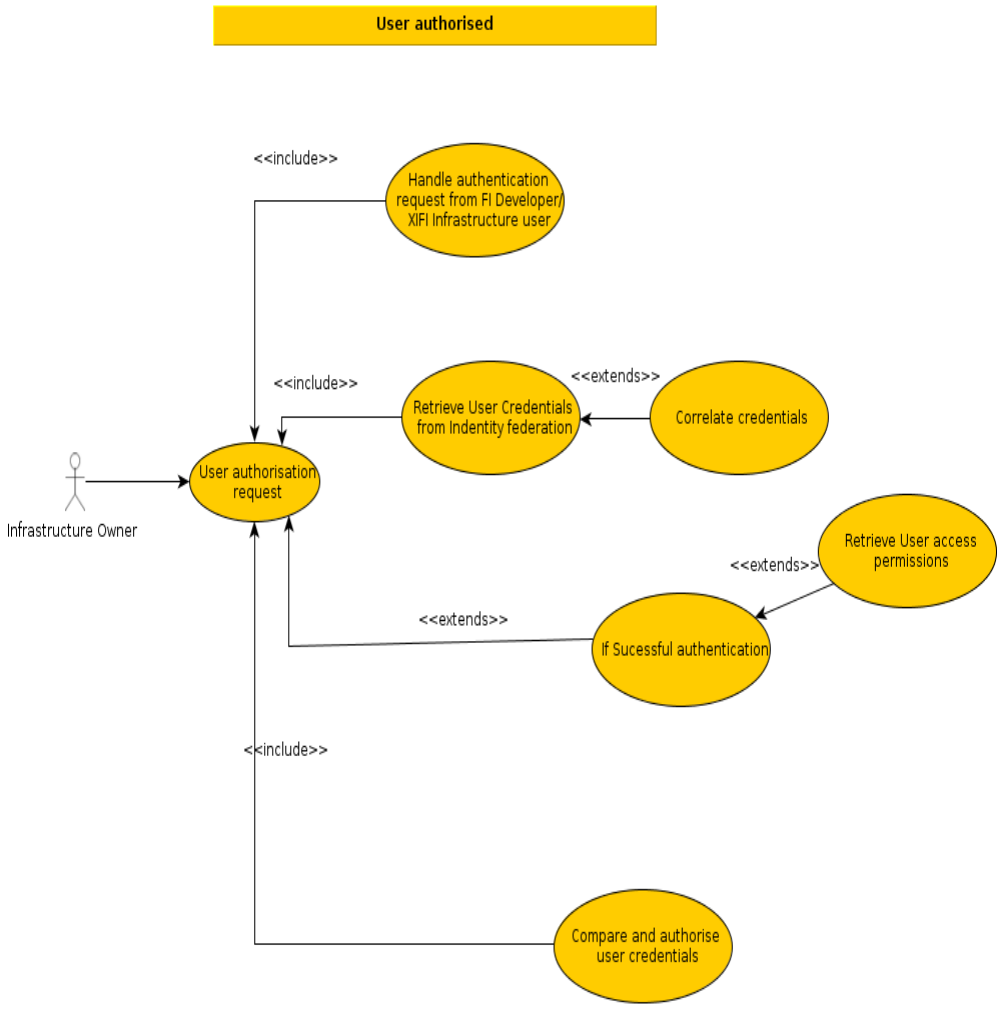
Pre-Conditions	The prerequisites of this scenario are: • The XIFI federation base is deployed and operational. • The Infrastructure is compliant with the operational, legal and technical terms of the XIFI Federation. • The infrastructure joining is compliant with the requirements for Master/Slave Node.
Process Dialog	1. FI Developer/ XIFI Infrastructure user requests authentication onto the infrastructure. 2. Adapter checks user credentials provided by the user and correlates them with the information available in the Identity federation. 3. Users authenticate successfully onto network. 4. Allow access to network infrastructure.
Diagrams	**Use Case Diagram** 

Table 8: Network-based Services Scenario 6

5.2.7 Scenario 7: Network bootstrapping of infrastructure resources

Section	Description
Title & Description	Network bootstrapping of infrastructure resources.
Author(s)	Adel Al-Hezmi (FhG)
Objective	Configure the end node (resource) with the required information to connect with the infrastructure (e.g. access network or IoT GEs).
Pre-Conditions	The prerequisites of this scenario are: • In case of using Universal Integrated Circuit Card (UICC), the associated SIM card should be configured with the required information • The XIFI federation base is deployed and operational, in particular the IoT management GE • Interface between IoT GEs or Monitoring GE should be implemented with associated network nodes (e.g. HSS in the 3GPP).
Process Dialog	1. If the resource is equipped with a UICC (e.g. SIM card), then UICC is configured with all the necessary information for performing Access Network registration. 2. Bootstrap **OTA (Over the Air):** The access credentials including the key material needed for registration operations is provisioned via an over-the-air mechanism. 3. Involve registration of the resource with the Access Network, based on the corresponding access network standards. 4. Register the infrastructure resource with the IoT GEs.
Post-Conditions	Infrastructure resource has successful register with the IoT GEs.

Table 9: Network-based Services Scenario 7

6 XIFI INFRASTRUCTURE MONITORING MIDDLEWARE

The **XIFI Infrastructure Monitoring Middleware (XIMM)** is an adaptation mechanism for a multi-domain measurement framework, providing a unified control and access to performance's metrics of the infrastructures, also known as nodes, belonging to the XIFI federation. In other words, considering the variety of different monitoring solutions deployed by the infrastructures participating in the XIFI project, XIMM will be the abstraction layer in charge of collecting, standardizing and publishing the measurement results of this multi-domain environment. It will act as the client and discoverer of network and datacenter performance metrics (subject to locally-determined policy restrictions) to feed the XIFI Services and Tools requests.

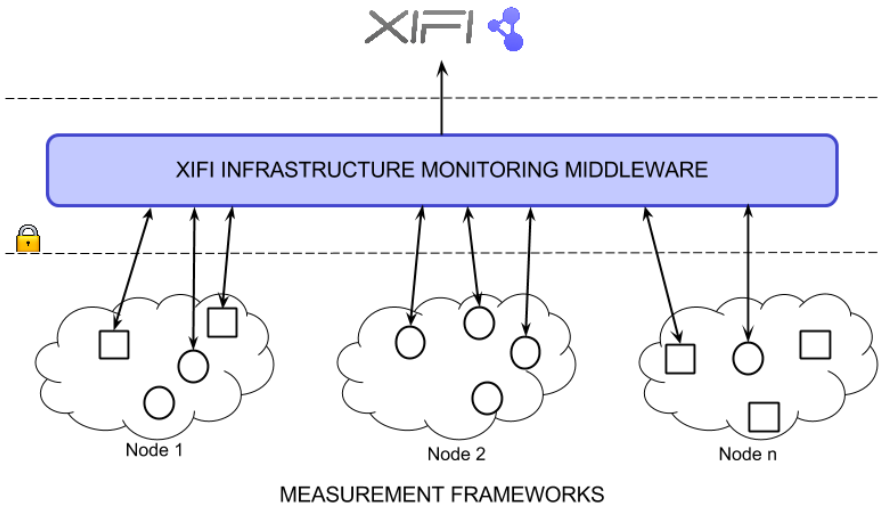


Figure 30: XIMM General Overview

6.1 Context

Previously to address its design, we need to analyze the context this federated monitoring framework will need to handle. As depicted from the overview of Figure 31, we can identify the following entities:

- **XIFI Portal:** within the XIFI context, this portal represents the single entry point where the user gains access to the services provided by XIFI. It will be hosted in the Master Node of the federation
- **XIFI Service Providers:** the facilities in charge of providing certain service within the federation (e.g. Software as a Service - SaaS). These facilities may be hosted within any of the nodes of the federation, but they need to be accessible from the Master Node because there is from where the user has access. They also might enable the interaction among them to provide a combined service.

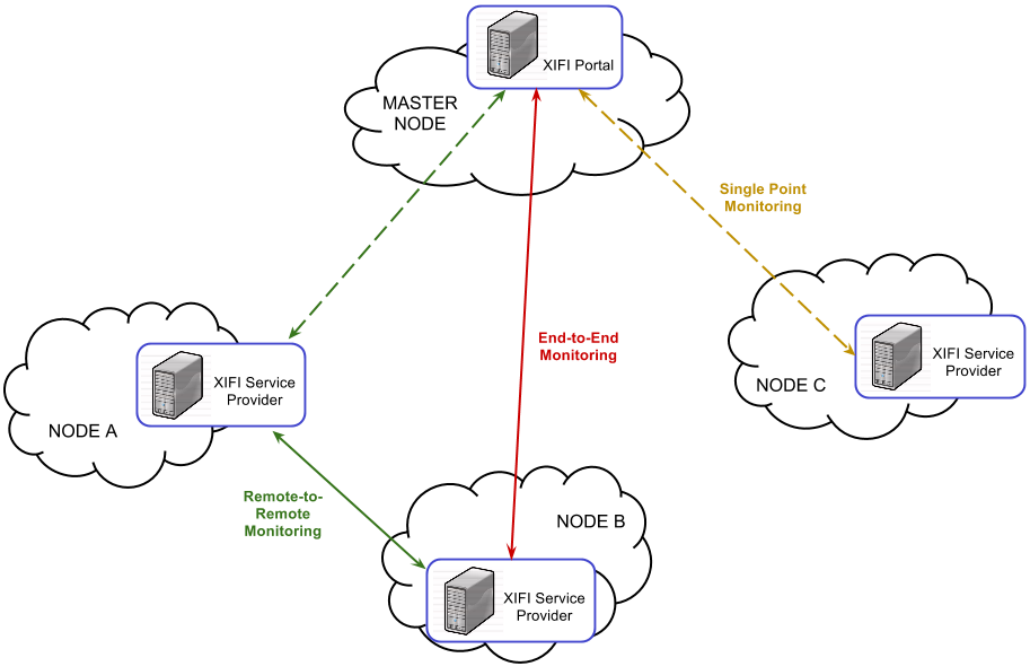


Figure 31: XIMM General Context

Considering these entities and their connectivity, the XIMM will need to face the following scenarios at first glance.

6.1.1 End-to-End Monitoring Scenario

The end-to-end performance monitoring is one of the most relevant operations the XIMM must fulfil in order to provide information related to the Quality of Service. A clear example of this scenario is when a user leverages, from the XIFI Portal, the service of a GE instance deployed in a remote node. He would like to know the availability of bandwidth between the peers. This simple case reveals that most of the interesting end-to-end performance issues span multiple administrative domains. The framework should work equally well for a set of connected infrastructures run by autonomous administrative entities as it does for a single domain.

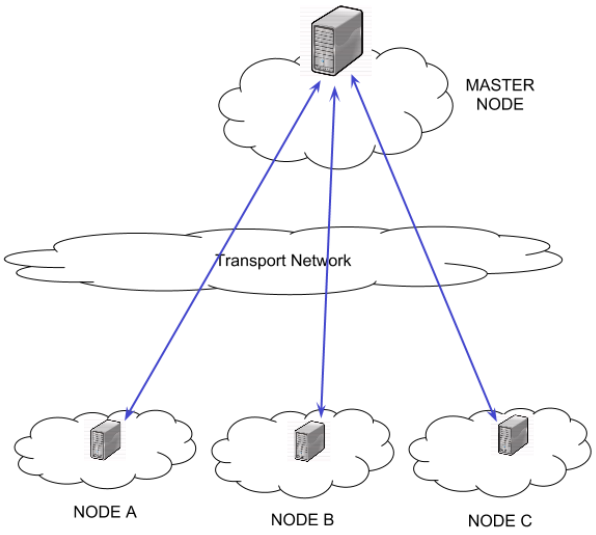


Figure 32: End-to-End Monitoring Scenario

6.1.2 Remote-to-Remote Monitoring Scenario

XIFI aims to work with available resources from a set of multi-domain infrastructures in order to provide services whose added value will be based on the federation of such infrastructures. Since this bond may imply the combination of some facilities from several domains, it will be required that the XIMM provides monitoring along the federation. This kind of measurement may be called Remote-to-Remote monitoring as a special case of End-to-End. This operation implies that the measurement is triggered and monitored from the Master Node, but the real test is performed between other two endpoints.

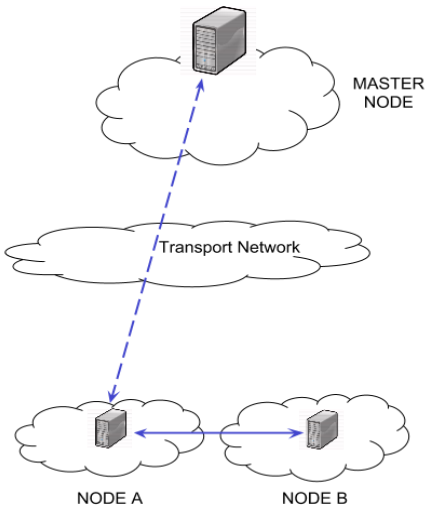


Figure 33: Remote-to-Remote Monitoring Scenario

6.1.3 Single Point Monitoring Scenario

In contrast to the previous cases, this scenario would only involve the monitoring of single points of relevance within the infrastructures. These points may be represented by a network device, such as a router, or a server belonging to a certain datacenter.

It is worth noting that there are a couple of possible cases depending on the configuration of the concrete infrastructure. The analysis of the Infrastructure Requirements (section 3.2) will concrete in which case each infrastructure is.

- If a management system (e.g. a Network Management System - NMS) runs the monitoring of the node, acting as a single provider of measurement data, the XIMM may obtain the data directly from it. This access might be under some restrictions since resources should be protected by certain policies settled by the infrastructure.
- However, if such infrastructure did not provide a centralized management system, the XIMM would need to branch along the domain to cover the hosts which XIFI has interest to monitor, i.e. where the GEs instances are. This results in a scattered set of measurement points.

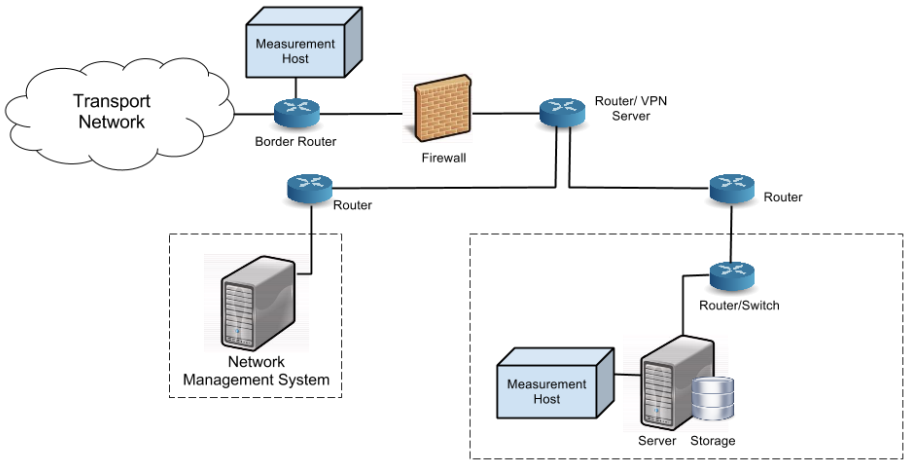


Figure 34: Single Point Monitoring Scenario

6.1.4 Datacenter and Generic Enablers Monitoring Scenario

Datacenter and Generic Enablers Monitoring is a particular case of Single Point Monitoring, but with special relevance. In such case, there is a need for monitoring either the performance of a physical node or a virtual machine. For that purpose, proper and distributed monitoring probes have to be installed in order to provide relevant information to the XIMM instance. Figure 35 depicts a high-level overview of such monitoring scenario.

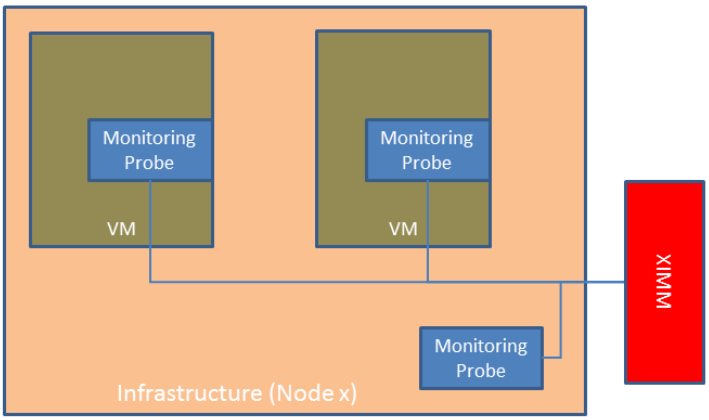


Figure 35: Datacenter & GEs Monitoring Scenario

6.2 Architecture Description

A typical Infrastructure Operator Centre is able to monitor and control the status and performance of the infrastructure of a single domain. By federating the measurement frameworks of a set of nodes which belong to autonomous administrative entities, XIFI Infrastructure Monitoring Middleware aims to provide a general picture of this environment, acting as if the performance data was being provided by a single domain.

According to the outcomes from the study [28], the most suitable approach to address a multi-domain measurement framework is to distribute the load of data among the different domains instead of processing the measurements from a central entity. This assessment must be subject to the characteristics of multi-domain networks and the impact of monitoring performance. Taking this guideline as a starting point, we shall proceed to define the basis of the middleware which aims to orchestrate the monitoring frameworks of the XIFI federation.

6.2.1 XIMM Ecosystem

Following a top-down approach, we may outline the XIMM adaptation mechanism as an ecosystem. The **XIMM Ecosystem**, as it is depicted in the Figure 36, will be composed of a set of XIMM instances distributed along the federation, settled in those points of interest from where we require some measurement data. The more instances are deployed the more fine-grained the global monitoring will be. The instances may interact with each other if an end-to-end test is required, or they may provide single point metrics.

XIMM instances are the actual adaptation mechanism by following a distributed scheme, being the entities in charge of interacting with the different measurement frameworks. However, they need an entity in a higher level of the federation. This entity shall be in charge of aggregating and filtering the incoming measurement data provided by each instance, querying when an on-demand request is made. The role of this entity has been represented in Figure 36 by the *Federation Aggregation Module*. This agent is out of the scope of this description since it is not part of the adaptation mechanism but the federation level. Nevertheless, although it might been represented by a single and centralized entity within the Master Node, we foresee it might also be composed of several secondary modules, the *Node Aggregation Modules*, scattered along the domains.

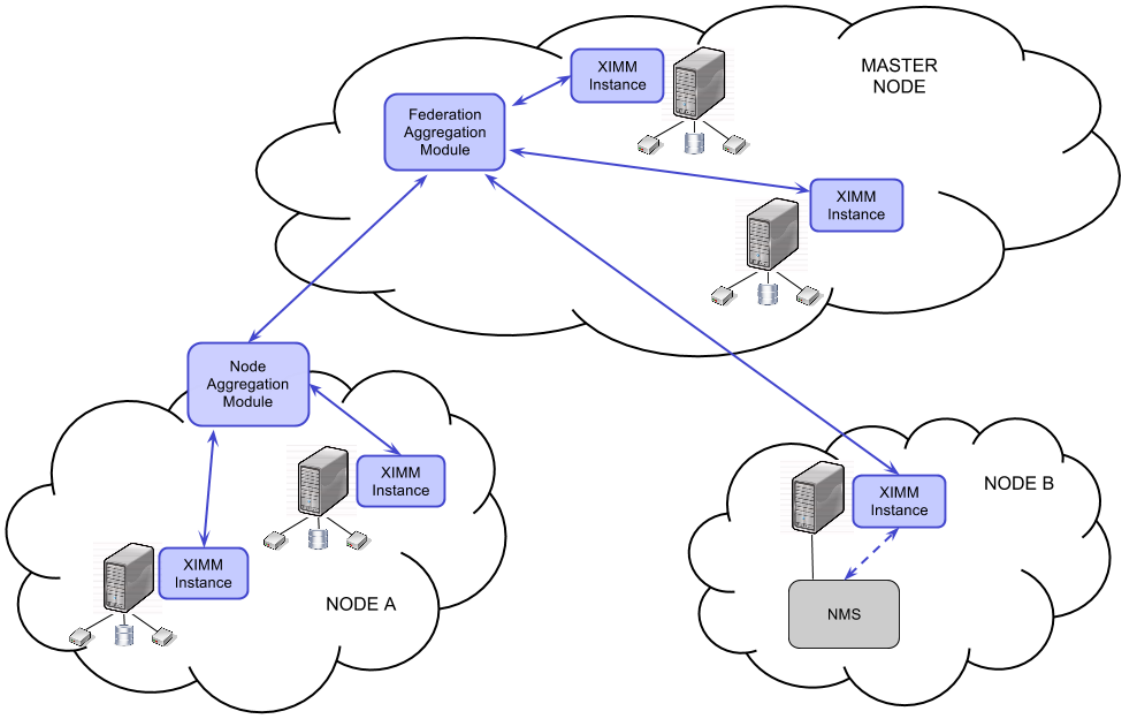


Figure 36: XIMM Ecosystem overview

6.2.2 XIMM Instance

Going into further detail, the step forward is considering the functionalities and impact of a **single instance of the XIFI Infrastructure Monitoring Middleware (XIMM)** within a domain, independently of the role this one plays in the federation and the location that might been settled to place the instance. As it has been already mentioned previously, a particular instance is the piece of software in charge of handling the measurement data coming from lower levels, the measurement framework relative to the concrete infrastructure, to expose it to the rest of the federation. From this description, it is inferred that a single instance will not create new measurement data. It will leverage the existing tools to provide some added-value services.

At first glance, from Figure 37 we clearly differentiate two bound tiers, the Measurement Framework and the proper instance. The Measurement Framework brings together all those tools and systems deployed by the infrastructures to monitor and control the performance of the domain, or at least a specific part of it. They gather data from physical devices and leverage storage facilities to carry out the measurements.

The instance will be required to handle three different types of measurements provided by these tools beneath. A couple of them related to network performance, active and passive measurement; and another based on datacenter performance, which shall include the monitoring of GEs instances deployed in VMs. However, all these metrics will only be accessible through security terms, following certain privacy policies agreed between the infrastructure and XIFI. A more detailed description of the instance will be approached in following sections.

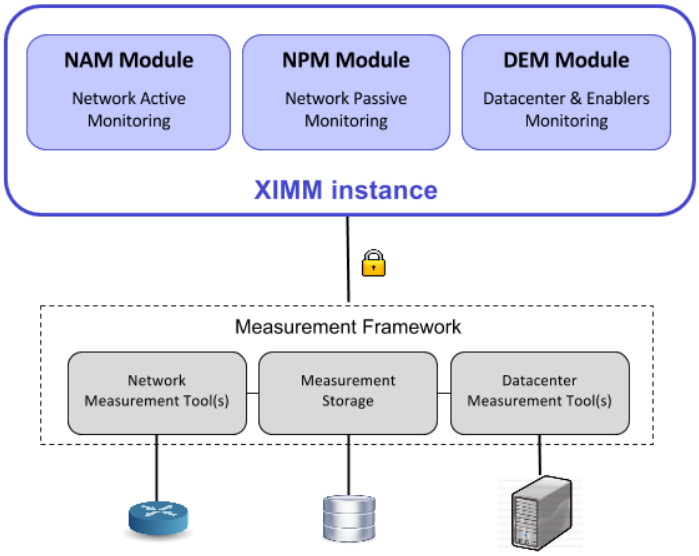


Figure 37: XIFI Instance overview

6.2.3 Main Requirements

Apart from the general requirements (section 3) which shall to be considered, the XIFI Infrastructure Monitoring Middleware needs to fulfil a set of specific requirements:

6.2.3.1 Functional Requirements

The XIFI Infrastructure Monitoring Middleware that is being addressed should be able to fulfil the following Functional Requirements:

- The system shall monitor and provide the status and performance of the infrastructures federation in terms of network and datacenter metrics. It is needed to link several multi-domain information sources under a common framework.
- The system shall span intra-infrastructure measurements in order to complete the solution of managing end-to-end measurements.
- The system shall encompass different existing monitoring tools to provide various metrics. Standardizing interfaces between various architecture components and providing a clear flexible system structure for easy extensibility.
- The system shall assure security issues, being able to deal with customized requirements on which metrics can be accessed. Infrastructure owners may wish to restrict some sort of data.
- The system shall enhance the existing measurement tools the infrastructures host, not replace them.

6.2.3.2 Technical Requirements

- **Scalability**:
    - Ability to increase the number of service capabilities. It must be possible to add and

remove components of the measurement infrastructure with a minimum effort of re-configuration. The number of management domains, network elements, measurement points or monitoring tools will increase and might vary as the federation includes more and more infrastructures

    - Ability to cover true end-to-end performance

- **Elasticity** with the change of the data model: accommodation of new types of metrics and parameters in the future
- **Cross-domain interoperability**:
    - Services need to be capable of federating locally and globally. It is required a component per domain that communicates with its neighbours to find needed services
    - Data collected on one infrastructure or between two of them must be available to other infrastructure, especially the Master Node
- **Adaptability** with other monitoring systems, technologies or implementations. Whichever existing monitoring approach or combination should be able to be used by the architecture to provide the necessary information
- **Open-based**: open software solutions and independence from commercial software, unless the commercial software may provide a key benefit to the system
- **Standard**: use of standard protocols
- **Security**: authenticated access to services and discovery of "trusted" services
- **Accuracy**: the ability to measure detailed activity with high accuracy and precision from the measurement tools through the application layer (e.g. latency)
- **Decentralization**: design following a multi-domain distributed monitoring approach

6.2.4 XIMM Main Components

Once described the general picture and assessed the main requirements, it is time to concrete and define the main components and the involved connectivity in a XIMM instance. Since NAM, NPM and DEM Modules are the main agents of this adaptation mechanism, they will be addressed separately in further description in following sections (6.2.6, 6.2.7 and 6.2.8). An extra functionality may be integrated within each one of the three modules. They can be designed to keep track of those results collected into a Measurement Cache Storage, providing historical data by requesting directly the instance.

This document does not aim at defining how to fulfil the implementation of these components. Nevertheless, the reader shall be aware that they have been specified by following designs and data models from existing technologies, special mention for PerfSONAR and Nagios tools. For further details, take a glance in see section 4.

Figure 38 depicts a detailed XIMM instance through a FMC component model.

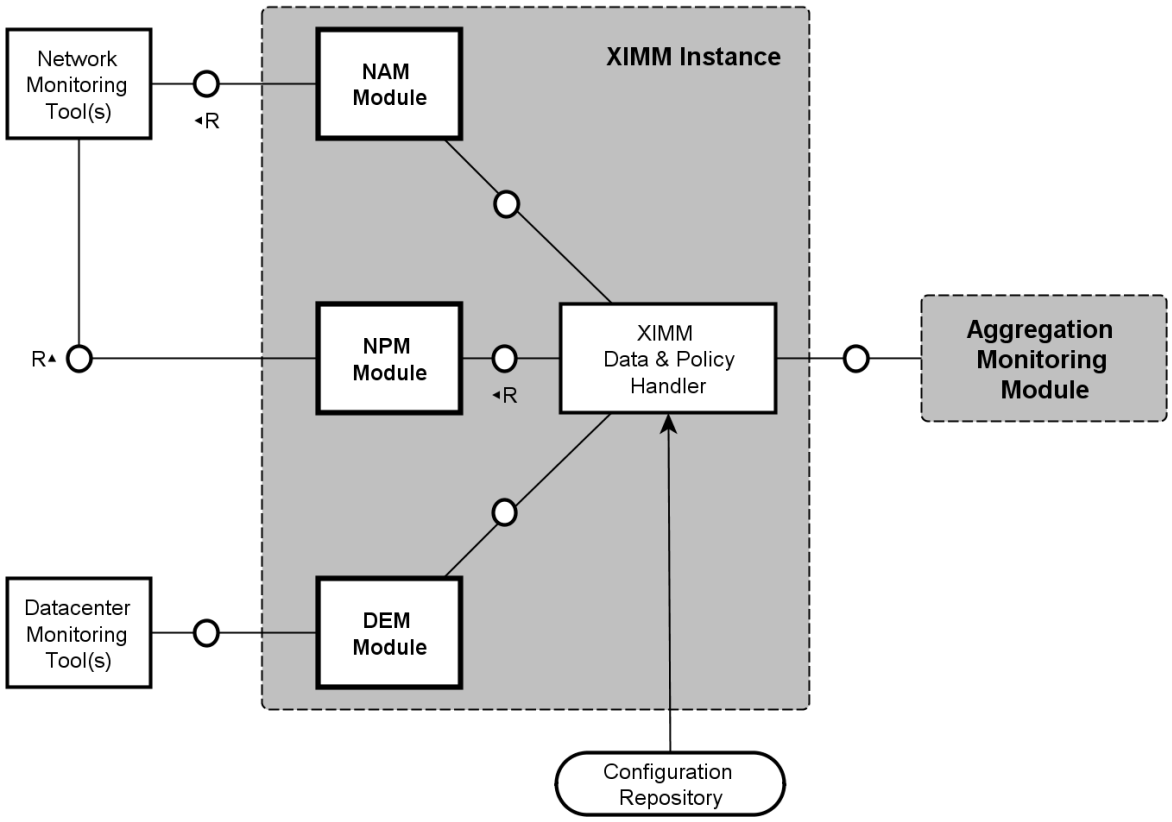


Figure 38: XIFI Instance components

6.2.4.1 NAM Module

This component will be in charge of managing the ***Network Active Monitoring*** **(NAM)** of the instance. To achieve such duty, the module will be required to interact with Network Monitoring Tools by fetching and publishing the measurement data. NAM Module cannot work as a standalone program since this type of monitoring requires at least two peers (end-to-end measurements). It mainly handles latency and bandwidth tests, both on-demand and scheduled.

6.2.4.2 NPM Module

The module responsible for handling the ***Network Passive Monitoring*** **(NPM)** data from network equipment. According to the nature of this type of monitoring, the data is generated by singles points in the network of the infrastructures. Data is expected to be collected through variables from the Simple Network Management Protocol (SNMP). However, results may also include the possibility to analyse link utilisation.

6.2.4.3 DEM Module

This adapter will handle the access to metrics of specific hosts and services which belong to the datacenter of the infrastructure and hold a meaningful behaviour for XIFI. Since the special interest of the federation lies on the performance of FI-WARE GEs instances, their monitoring is a duty that a XIMM instance is meant to address. Hence the module is for ***Datacenter and Enablers Monitoring*** **(DEM)**.

6.2.4.4 Measurement Framework

This framework refers to the **set of measurement tools**, both for network and datacenter metrics, **and storage facilities** dedicated to historical measurement data. Strictly speaking, they do not belong to the XIMM instance at first glance, but are closely integrated with it. In some case, a tool and the modules of the instance will belong to the same implementation, making the blurred boundaries difficult to distinguish. The main functionalities of these agents are the generation and storage of the monitoring data.

It is expected that some of these tools may be already deployed within the domain of each infrastructure, being used by the corresponding NOC to monitor the performance independently from XIFI. Nevertheless, in order to assure the complete functionality of the instance, some other measurement tools might need to be included to fulfil the general picture. It is important to make clear that these additional tools would not intend to replace the current monitoring systems owned by the infrastructures. It would a manner to enhance the instance.

Those metrics which can be provided as raw data are:

- **Network Active Measurements:** provide the capability to make a custom-specified measurement between a pair of measurement tools. The coordination between any two collectors to perform a measurement is essential
    - One Way Delay
    - Delay Variation (Jitter)
    - Packet Loss
    - Achievable Bandwidth
    - Layer 3 Path (e.g. Traceroute, Tracepath)
- **Network Passive Measurements:** allow specification of parameters that would be used in capturing and deducing measurements from network traffic or in retrieving network information status
    - Layer 3 Statistics. Information retrieved from network-attached devices (e.g. SNMP)
    - Link Utilisation
- **Datacenter Measurements**
    - CPU Usage
    - Memory Usage
    - Host Alive
    - Disk Free

6.2.4.5 Aggregation Monitoring Module

The aggregation module will be the client in charge of querying and/or receiving the measurement data from each XIMM instance to provide some added-value functionalities at federation level. Although the topology of this framework is out of the scope of this specification and will be addressed in other branch of the project, here we may suppose there will be an instance in each node of the federation, being controlled by a centralized entity in the Master Node.

According to the context of the XIFI Project within the FI-PPP, working in a tight relationship with FI-WARE Project, we may foresee the opportunity to work with the Context Broker GE [13] as this

aggregation agent. Based on its description, the Context Broker will enable publication of context information by entities, referred as Context Producers, so that published context information becomes available to other entities, the Context Consumers. Matching this terminology with our specific case, the modules within the XIMM instance will be the Context Producers and the Services and Tools of the XIFI Portal would be the Context Consumers. Both NAM, NPM and DEM modules will need to handle their specific context and be compliant with this Context Broker. This mainly implies that the modules are meant to use the FI-WARE NGSI APIs to interact with the Context Broker GE (See FI-WARE NGSI-9 API [18] and FI-WARE NGSI-10 API [17]).

6.2.4.6 XIMM Data & Policy Handler

In order to provide a single entry point and enhance the functionality of the XIMM instance, there has been included a fourth module as an intermediary entity between the three modules (also known as Context Producers in the Context Broker terminology) and the Aggregation Module. With the inclusion of this extra component:

- The XIMM instance is made independent from the querying entity. Here the Aggregation Module is being considered as the main requester. However this assumption may vary or be extended to other agents. This way the *Data Handler* forces the modules to pass through it, reducing the number of dependencies with an external connection.
- In the description of the context it was introduced that the access to infrastructure's resources will be limited by certain policies set by the own node. Leveraging a *Configuration Repository* where these policies may be stored, an *Access Handler* shall enforce the restrictions and control the access to the resources, performing the role of an authorization agent. Otherwise, each module should carry out this process by its own.
- The *Configuration Repository* may be also helpful by hosting some kind of valuable constraints and rules. The handler shall trigger tests based on what these configuration parameters state or provide the data in a certain format. The most clear example of parameter is the scheduled times to monitor.
- This handler may also contain a *Local Discovery* component in charge of listing and publishing the available capabilities of the concrete XIMM instance. Even considering that the *Aggregation Module* may expose to higher levels the components that are beneath, it is required that these components had been previously registered to become visible. Hence, this module may pre-process the data and provide it through a compact format.

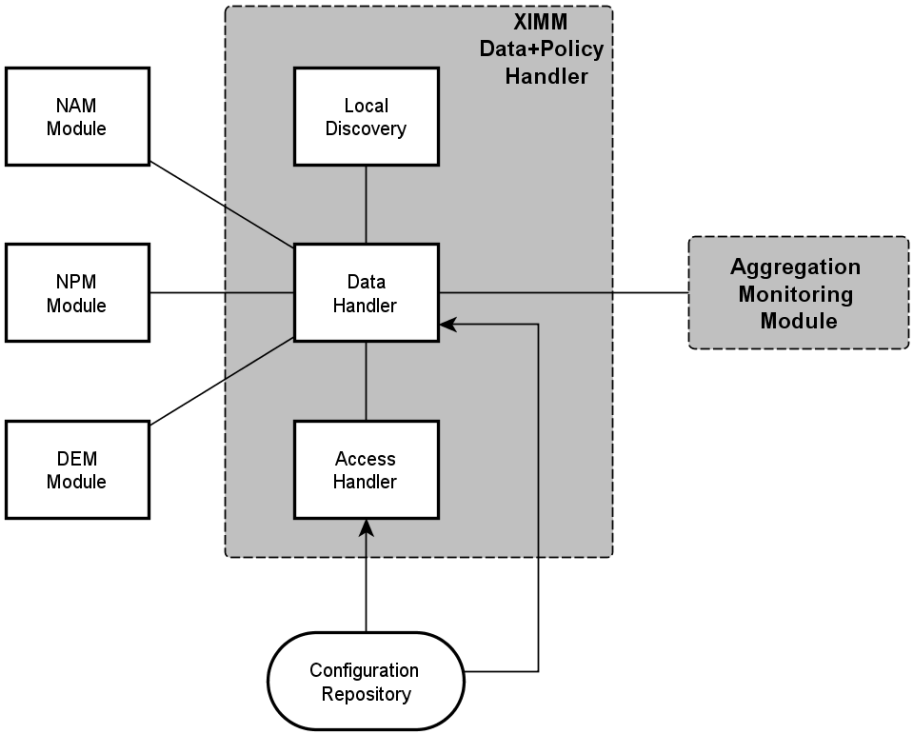


Figure 39: XIMM Data & Policy Handler components

It is worth noting that the existence of this module may be subject to future considerations according to the final specification and integration of the *Aggregation Module.*

6.2.5 XIMM Services

Previously to start defining each one of the XIMM modules in detail, it would be meaningful to deal with a 'Service' level point of view of the perceived architecture. This allows a separation of services into specific self contained units, and makes possible the layering of the federation. A large task can be split into independent services to avoid monolithic software blocks being difficult to maintain, although real implementations may imply the combination of some or all of them. These and new services can be dynamically added/dropped which results in an increasing flexibility and robustness. This approach is particularly well suited for this case because infrastructures are dynamic entities with routes, connections, and devices in a constant state of change.

Figure 40 shows those services which have been identified to be involved in the set of XIMM instances within a domain. We shall perceive a distribution of the services in two different tiers: the Adaptation and the Federation Layers. The inclusion of the Federation Layer in this description is due to the existing and close interaction with the services belonging to the Adaptation Layer. This Federation Layer will be represented by the *Federation Aggregation Module* we foresaw in the introduction, with the possibility of a secondary *Node Aggregation Module* in charge of a specific domain.

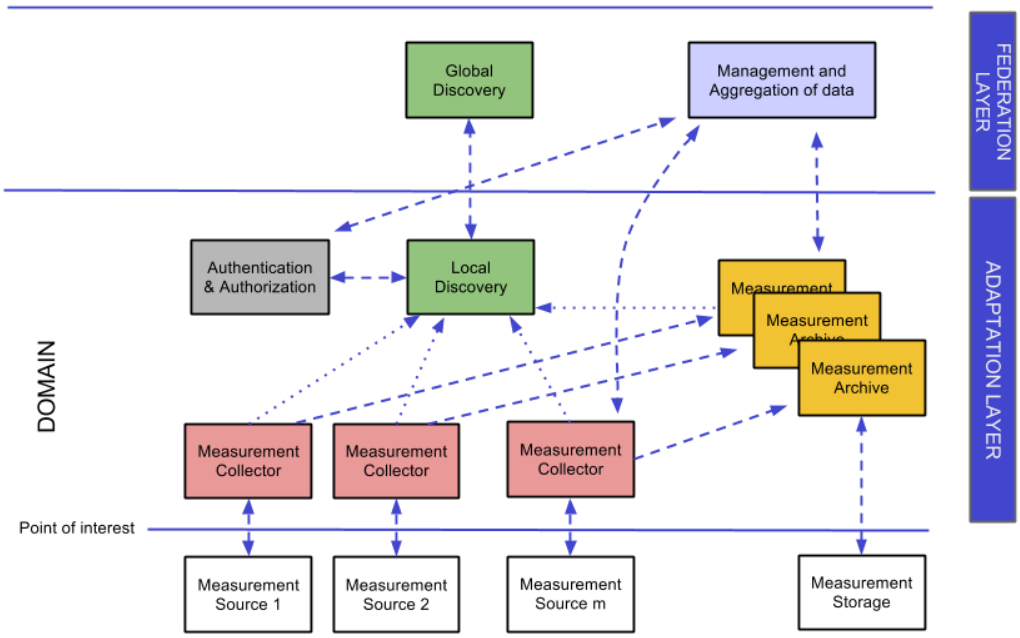


Figure 40: XIMM Services

The Adaptation Layer consists of a modular set of services that enable control and access to measurement data across the administrative domains. The aim of these services is to expose the measurement data, within domain-defined security restrictions, to the Federation Layer services.

Services will be required to support both push and pull data flow models. Each data provider needs to implement a publisher interface and each data consumer needs to implement a subscriber interface. When a data flow is requested, the consumer will provide a handle to a subscription interface if it wants a push interaction. If it does not provide a subscription handle, the data producer will create a publisher interface that the consumer can poll.

6.2.5.1 Measurement Collector Service

The *Measurement Collector* is a standard interface wrapper around a monitoring tool and will be the service responsible for collecting the data created in the process. The tests are executed by the service, leveraging the tools beneath; hence a common interface to these capabilities is required for ease of integration into the monitoring system as a whole. Once the process is completed, the data is either published to the 'consumer' or stored through a *Measurement Archive* as part of the historical dataset.

The list of functionalities that such service may provide:

- Collect the data generated by the monitoring tool wrapped in the service. The measurement requests can be of two types, on-demand and regular tests.
- Deal with on-demand requests coming from *Federation Services*. There is need for an interface which interacts with querying agents, processing incoming requests and publishing the associated outcomes through responses.
- If a scheduled test has been performed, the service shall provide an interface to store the data

over a *Measurement Archive*.

- An interface to register, de-register and refresh its status with a *Discovery Service*.

6.2.5.2 Measurement Archive Service

The *Measurement Archive* is in charge of dealing with historical measurements stored in an archive. It is the service designed to store and retrieve such performance data over time, but it does not create new raw data. As the *Measurement Collector*, it acts as a wrapper, this time on top an existing data archive. Its design allows a single type (or several related types) of measurement information to be stored in the same facility. Data must be available in a standardized format to assure interoperability. The archive can be, for example, a network's Round Robin Database, relational database or a proprietary database of a Network Management System.

6.2.5.3 Discovery Service

Discovery Service addresses the issue of resource registration and discovery of service's capabilities. In other words, it allows you to check which services are available and what kind of data they provide. Every time a service starts running, it can register with the *Discovery Service* to signal its availability and provide a description of its capabilities (for instance, in terms of a copy of the metadata storage that each service maintains).

Service instances managing datasets are only useful when they can be contacted by consumers and consumers can only function when there is data available. The *Discovery* acts as a service directory, where services can advertise themselves (provide their lookup information) and requestors are able to find any service they need. New services may identify themselves to the framework and offer their capabilities subject to locally-determined policies.

- *Global Discovery* interacts directly with the other portions of the framework at Federation Level. It is regularly synchronized with all other *Global Discoveries* instances to ensure they have an up-to-date global view of which services components are available. It responds to queries about services and data, e.g. services looking for each other or client applications looking for data.
- *Local Discovery* manages the lower layers of discovery. Although there may be a *Local Service* associated to a *Node Aggregation Module* at Federation Level, here we are considering the service provided by each XIMM instance in order to expose the services it contains.

6.2.5.4 Authentication & Authorization Service

A XIMM instance needs to control which consumer gains access to the services, at the same time that there might be some constraints in the accessibility of resources that the infrastructure may enforce. Therefore, the *Authentication and Authorization Service* will provide mechanisms in charge of managing the access to capabilities. Here it may be considered to use the Security service provided by FI-WARE, i.e. the Identity Management (IdM) GE [15] and access control.

6.2.6 XIMM-Network Active Monitoring (XIMM-NAM) Module

The set of XIMM-NAM modules distributed among different instances will be in charge of handling the active monitoring of the networks along the federation of infrastructures in XIFI. Active monitoring relies on the capacity to inject test packets and following them to measure the service provided. The volume and other parameters of the introduced traffic is fully adjustable, what implies testing what you want, when you need it. This emulation of scenarios will enable us to check if

Quality of Service (QoS) and *Service Level Agreements* (SLAs) are accomplished. It is worth noting that small traffic volumes are enough to obtain meaningful measurements, what avoids the overloading of the network.

Figure 41 provides a graphical description of the interactions among internal and surrounding components through a FMC diagram. As it is depicted, none of the services within the module will perform the measurements indeed. By definition, this is not intended to be the task for the module. The responsible entities for this action will be the measurement tools embedded in hosts of interest (or at least near them). The *Measurement Collector* services, as the name states, will collect the data and expose it to the requesting agent.

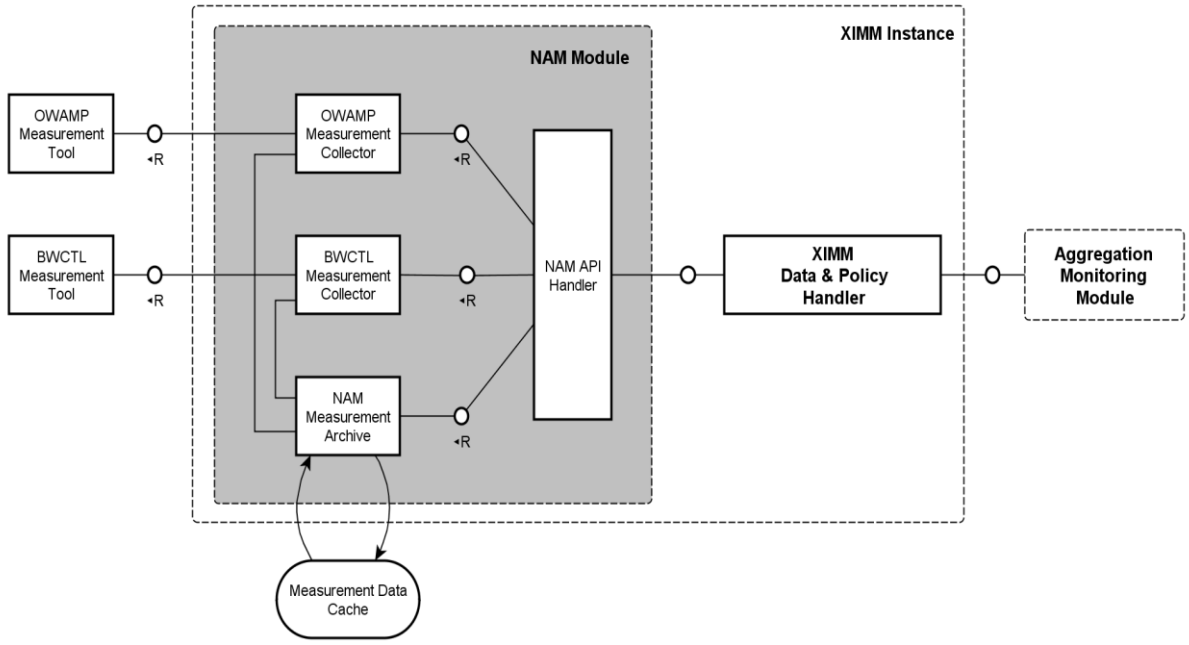


Figure 41: XIMM-NAM Module components

Considering the opportunity of keeping a local record of the results provided by the tools, the data may be sent to the *Measurement Archive* service which shall store the fetched outcomes in a cache storage. The inclusion of this component, although it is not mandatory, would enable the possibility of a future access to historical measurement results by requesting directly to the module.

In order to carry out network active monitoring it is required that peering modules work in a collaborative manner, i.e. through an end-to-end communication (source and destination). Between endpoints of interest the data can be obtained by both on-demand and regular measurement tests. Historical measurements represent results of regularly scheduled tests and shall cover **one-way delay**, **jitter**, **one-way packet loss**, **traceroute for a path** and **achievable throughput for a path**. Nevertheless, XIMM-NAM Modules also offer the possibility of requesting an **on-demand measurement of achievable throughput or one-way latency measurement** between endpoints.

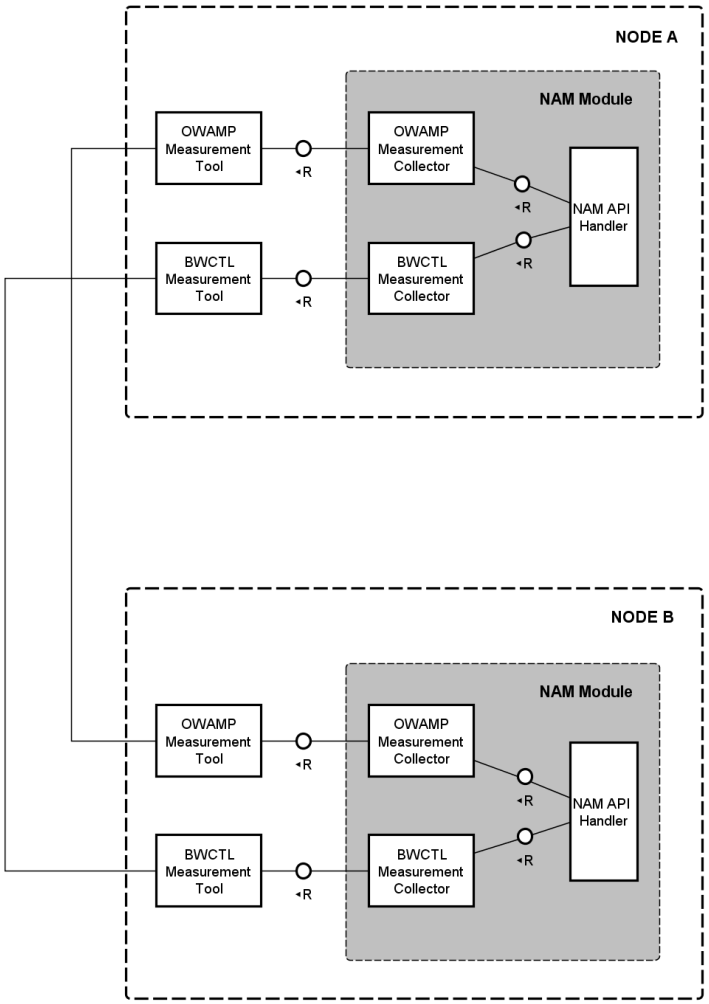


Figure 42: XIMM-NAM Modules interaction

6.2.6.1 OWAMP

Through the XIMM-NAM modules is possible to request one-way latency measurements based on the **One-Way Active Measurement Protocol (OWAMP)** defined by the IETF RFC 4656 [29]. One-way tests allow the user to better isolate the effects of specific parts of a network since traffic can be asymmetric at many sites. OWAMP has been designed to be deployable on as many systems as possible and not to introduce any new scalability problems. It allows the user to conduct only those measurement sessions desired, and to retain as much data as desired. OWAMP also does not dictate a choice of site(s) where measurement results are stored: it is possible to have all data stored at a central site or to store data at each receiver and fetch it as needed.

The OWAMP tool is a typical client-server application. The *owping* client contacts the *owampd* daemon on the peer host to request a specific test. *owampd* allows a system administrator to configure any given host as an OWAMP test endpoint. Also, *owampd* is responsible for implementing the policy restrictions imposed by the system administrator on that system.

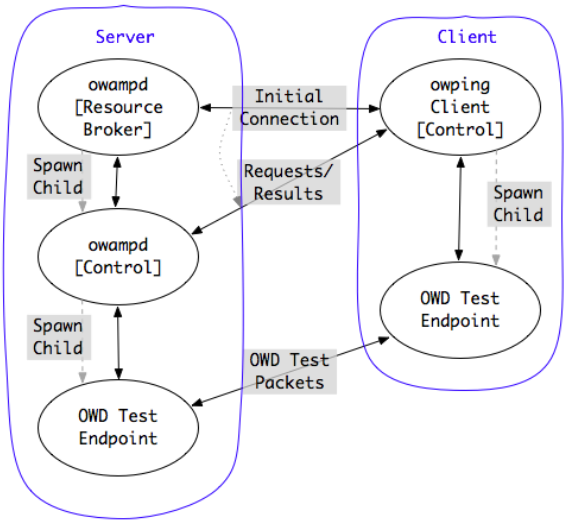


Figure 43: OWAMP Architecture [42]

Some requirements which need to be considered:

- OWAMP prefers a synchronized clock for measurements to be meaningful. Even more important, the clock needs to be stable.
- OWAMP should be run on real hardware. Virtualization packages will exhibit clock instability that will make most OWAMP measurements misleading.
- Possible power-management features of most PC hardware should be disabled. Speeding up and slowing down the processor makes for a very instable clock.

The *OWAMP Measurement Collector* will be the actor in charge of dealing with the OWAMP tool, collecting the data and exposing it to the rest of components. This *Measurement Collector* executes the measurement between two OWAMP-enabled endpoints and returns the results, e.g. in a XML response. If the *Measurement Cache* has been set up, the collector is able to send the data to the *Measurement Archive* associated.

There are some parameters that will be required to be considered for an OWAMP on-demand test:

- Packet count: number of packets being sent
- Wait time: average wait time between packets
- Timeout: maximum time to wait for a packet before considering it lost

A more detailed description can be found in OWAMP Details [42].

6.2.6.2 BWCTL

Bandwidth Test Controller (BWCTL) is a tool which allows the execution of on-demand and schedule authorized bandwidth tests in a non-conflicting manner between two endpoints. The aim is to determine the achievable or available bandwidth between a pair of hosts. BWCTL is a wrapper around

a throughput testing tool responsible for conducting the tests, such as Iperf [31] or Thrulay [64]. BWCTL prevents tests to and from a given location from interfering with one another.

BWCTL is composed of a client application (*bwctl*) and a scheduling and policy daemon (*bwctld*). The *bwctl client* works by contacting a *bwctld* process on the two test endpoint systems. These *bwctlds* manage and schedule the resources of the hosts on which they run. When a measurement is granted, the *bwctld* processes will run the test and return the results to the client from both sides. The local *bwctld* is not mandatory. The client will detect if this local daemon exists and use it if available. Otherwise it will provide the missing functionality when needed.

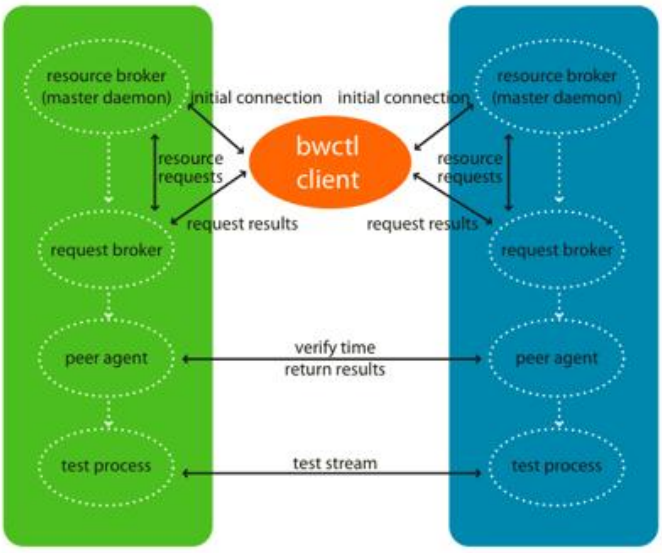


Figure 44: BWCTL Architecture [3]

Some requirements related to BWCTL are:

- It includes the Thrulay tester, but it would be best if Iperf is installed on the system.
- The *bwctld* daemon requires that NTP be running to synchronize the system clock.

There are several parameters to consider for a BWCTL on-demand test:

- Protocol: specifies the protocol used. The possibilities are TCP (by default) or UDP.
- TCP windows size
- UDP buffer size
- Maximum bandwidth: limits the maximum bandwidth (in Mbps) for UDP protocol
- Address type: both IPv4 (by default) and IPv6 are supported
- Reporting interval: the service should attempt to run a throughput every interval seconds

A more detailed description can be found in BWCTL Details [3].

6.2.6.3 NAM Measurement Archive

This service is responsible for dealing with the historical data originated from previously performed tests. The results are received from the collectors and stored for future queries, acting as a cache storage provider.

The type of data which is stored through the *OWAMP Measurement Collector* and accessible by the *Measurement Archive* is one-way delay, jitter, one-way packet loss and traceroute data. As example of a possible representation, Figure 45 displays a graph of the results from a query. The first segment would represent the IPDV (jitter); the second is oriented to delay (including minimum, mean and maximum values); and the third one to packet loss and packet duplicates, as well as hop count for the route.

Figure 45: Example of possible OWAMP MA data representation [55]

It is also possible the access to available throughput historical data, gathered by the *BWCTL Measurement Collector*. The first step will be selecting the concrete *Measurement Archive* to query. The measurements are carried out between two endpoints, so it will be required to specify the source and destination from all the available endpoints. A possible graphical representation is shown in Figure 46, where it may represent a series of individual available throughput measurements performed in a regular basis.

Figure 46: Example of possible BWCTL MA data representation [55]

6.2.6.4 NAM API Handler

The *NAM API Handler* is the intermediary agent responsible for interpreting and dealing with Context Broker-format and NAM API-format requests and responses.

- According to the Context Broker model [13], the handler will be the *Context Producer* feeding the *Aggregation Module* (the actual *Context Broker*). Hence it must provide an interface compliant with FI-WARE NGSI-9 API [18] and especially FI-WARE NGSI-10 API [17] to interact with the Context Broker GE.
- On the other hand, it is also required to provide an interface to work with the *Measurement Collectors* and the *Measurement Archive* using the corresponding NAM API.

NAM module is expected to support both 'push/pull' models regarding data transfer as it is displayed in Figure 47 (taking it as a high-overview picture).

Figure 47: XIMM-NAM Dataflow overview

- If the *Data & Policy Handler* triggers a scheduled test, the measurement will be performed by the proper tool and the results will be returned through the concrete *Measurement Collector*. The handler would 'push' (*updateData* operation) the results to the *Aggregation Monitoring Module* which will be acting as the client of the instance and work the data out to provide a service at federation level. As it has already been commented previously, the outcomes might be also 'pushed' to the *Measurement Data Cache*.
- On the other hand, the NAM module also shall be capable of supporting the case when the *Aggregator Module* may invoke it at any given time to query on values measurement data (*queryData* operation), either by fetching historical results from the cache or by requesting an on-demand test.

6.2.7 XIMM-Network Passive Monitoring (XIMM-NPM) Module

XIMM-NPM is the module responsible for handling the measurement data obtained from the passive monitoring of network devices, information which is extremely valuable in network trouble-shooting as real traffic is monitored. This type of monitoring does not imply an increase of load by injecting artificial traffic. The data is collected periodically through external means, the measurement tools, and kept into data storages. Unlike the NAM Module design (section 6.2.6), NPM Modules work independently one from each other since it is not required an end-to-end communication. The module only collects information about single specific points within the network.

The configuration of this module depends on the settings and accessibility of the concrete performance monitoring tool used by the infrastructure to carry out the network passive monitoring. Measurement data might be accessible either by querying directly the tool or by fetching the data from a certain storage where should be contained. In other words, the NPM module might need to deal with the data either with a *Measurement Collector* or with a *Measurement Archive* respectively. Since the actual implementation of the module may involve both services within an unique piece of software, there is no need to take this as a big issue.

As an attempt to make the module as much independent from specific tools as possible, the first attempt seems to be that the NPM module gains access to the storage where the measurement data is kept. Following this premise in our design, a *Measurement Archive* will be considered as the main agent to deal with the tool. In order to model the NPM Mole following picture (Figure 48) depicts a FMC diagram.

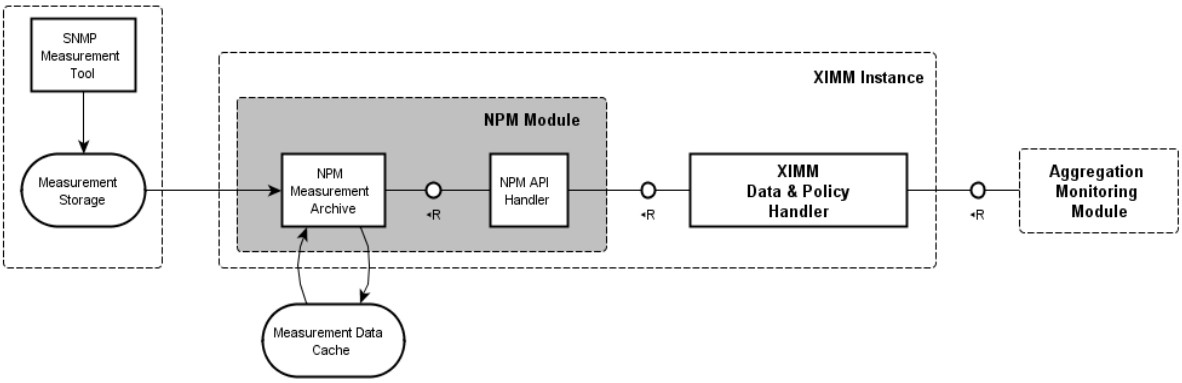


Figure 48: XIMM-NPM Module components

Another meaningful assumption that may be taken into consideration with regards to the *Measurement Archive* service is the possibility to keep records locally in a *Measurement Data Cache* when measurements were queried to the tool. Taking this action we would be able to access the logs directly by requesting the module, independently from any other actor.

6.2.7.1 NPM Measurement Archive

The *Measurement Archive* service provides an interface capable of exposing data collected via variables from the **Simple Network Management Protocol (SNMP)** found on network devices. Devices on the network may run a SNMP agent that is designed to continuously collect data about the state of that device and send the data to the configured storage. According to the premise taken in the design, there is from where the *Measurement Archive* might gain access to the data. It is out of scope to describe how the management station collects information from the network via agents and stores the data.

Using historical data from multiple *Measurement Archives* a path segment utilization for a given trace route output can be represented. It is possible accessing **link utilization data** for each segment hop of a given path: IP address, hostname, interface name, interface description, link capacity, maximum and average inbound and outbound link utilization for a specified time period. Figure 49 depicts a possible graphical interface of this data representation.

Hop	Address	Hostname	Interface Name	Description	Capacity	Max inbound utilization	Avg inbound utilization	Max outbound utilization	Avg outbound utilization
1	194.82.153.117	rachael-hh.net.ic.ac.uk	-	-	-	-	-	-	-
2	146.97.137.153	te0-4-0-1.londic-rbr1.ja.net	-	-	-	-	-	-	-
3	146.97.35.149	ae2.read-sbr1.ja.net	-	-	-	-	-	-	-
4	146.97.33.146	ae13.lond-sbr3.ja.net	-	-	-	-	-	-	-
5	62.40.124.197	janet.rt1.lon.uk.geant.net	ae0.0	Link (L) 20Gbps to JANET	20 Gbps	16.3 Gbps 81.5%	7.69 Gbps 38.4%	20.51 Gbps 100%	12.22 Gbps 61.1%
6	62.40.112.137	as1.rt1.ams.nl.geant2.net	as1.0	Link (L) to UK-GN	19.91 Gbps	8.57 Gbps 43.1%	5.52 Gbps 27.8%	15.45 Gbps 77.6%	8.56 Gbps 43%
7	62.40.124.230	esnet-gw.rt1.ams.nl.geant.net	-	-	-	-	-	-	-

Figure 49: Example of possible XIMM-NPM data representation [55]

6.2.7.2 NPM API Handler

The *NPM API Handler* is the intermediary agent responsible for interpreting and dealing with Context Broker-format and NPM API-format requests and responses.

- According to the Context Broker model [13], the handler will be the *Context Producer* feeding the *Aggregation Module* (the actual *Context Broker*). Hence it must provide an interface compliant with FI-WARE NGSI-9 API [18] and especially FI-WARE NGSI-10 API [17] to interact with the Context Broker GE.
- On the other hand, it is also required to provide an interface to work with the *Measurement Archive* using the NPM API.

The point which is relevant to notice is the fact that the NPM Module should only support a 'pull' model; that means that data can only be shared with the *Aggregation Module* when this is the actor invoking the handler to acquire the data (*queryData* operation). With this statement we are assuming

that the devices will be running correctly and only when the client is interested in checking their status, e.g. due to some unexpected behaviour, the XIMM instance will collect the data.

Figure 50: XIMM-NPM Dataflow overview

6.2.8 XIMM-Datacenter and Enablers Monitoring (XIMM-DEM) Module

XIMM-DEM, just as the XIMM-NPM model (section 6.2.7), is in charge of collecting and publishing measurement data from single points of interest within the infrastructures, but this time focusing on datacenter-based metrics. This kind of results basically includes performance data from hosts and services previously specified. Datacenter monitoring will be valuable for XIFI interests since it enables to check host resources, such as processor load or disk usage. Besides, service checking may be oriented to determine the actual performance of those FI-WARE GE instances deployed which provides the federation with service information.

Since none of the XIMM modules is intended to be a replacement of any measurement tool, the DEM Module will leverage the data obtained from an existing source, such as a system monitoring application, in charge of running the actual monitoring processes. There are several popular applications that can be taken into consideration for this purpose (special mention for Nagios (section 4.3.2) and OpenNMS [50]). Many of them rely on standalone programs called 'plugins' to perform the mechanisms for checking. This decentralization may be useful in our design since DEM Module can be compliant with most of them. The monitoring logic uses the results from the plugins to determine the current status of hosts and associated services on the infrastructure. Based on this data, the process shall take any necessary actions, e.g. running event handlers or sending out notifications.

Plenty of plugins may be engaged, monitoring different kind of devices and services. Nevertheless, it might be deemed necessary to develop new ones through available APIs (see as an example the Nagios Plugin API) if a more GE-oriented monitoring is required.

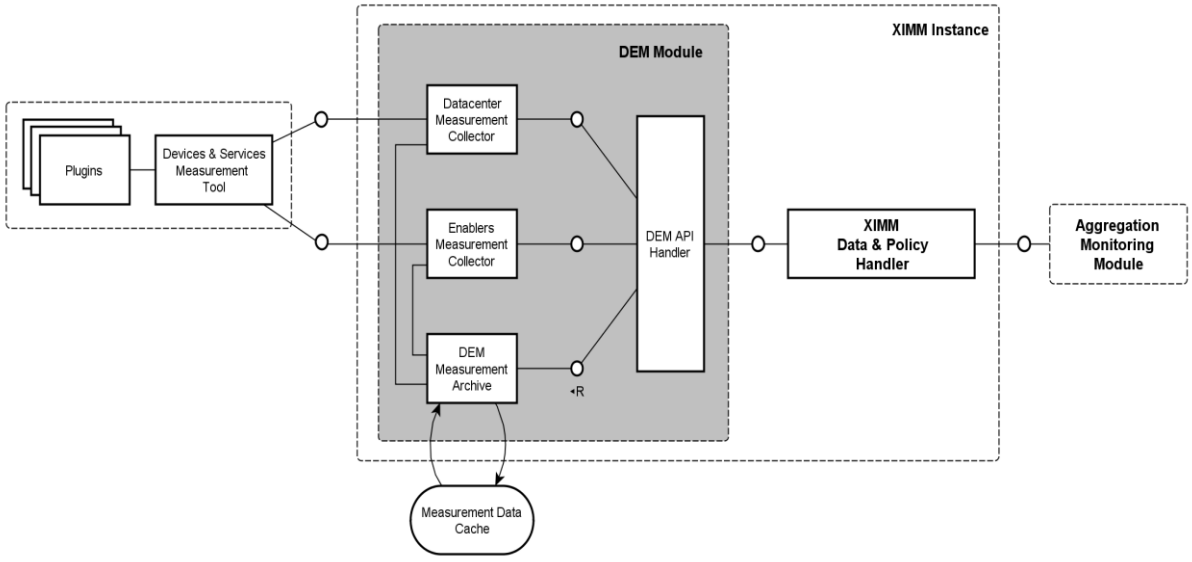


Figure 51: XIMM-DEM Module components

6.2.8.1 Datacenter Measurement Collector

According to the system measurement application model (e.g. based on the Nagios tool model), within each host and service definition it may be specified a 'contact list' which contains those agents that are intended to receive status, performance, or notification information. In the specific case of the DEM Module, the agent in charge of representing the role of such contact is the *Datacenter Measurement Collector*.

The way in which the system tool informs when a state change occurs is carried out through notifications. The decision to send out these alerts to the registered 'contacts' will be performed by the service check of the own tool. However, even if a 'contact' was included in the notification list, that fact would not directly imply that it will receive the notification. There is specified a set of filters which need to be configured so that the collector gets notified by the tool. Hence the *Datacenter Measurement Collector* shall be properly configured to be informed of such alerts triggered when processing a host or service check.

The *Data & Policy Handler*, leveraging the information contained in the *Configuration Repository*, may represent the agent responsible for setting up:

- The 'notification options'. Each contact definition contains options that determine whether or not host notifications can be sent out when the host goes down, becomes unreachable, or recovers.
- The 'notification period' option that specifies which time period contains valid notification times for the contact (the *Measurement Collector*). If the time that the notification is being made falls within a valid time range in the specified time period the collector will get notified.

The concept of notifications represents the main meaningful difference between the DEM Module with regards to the NAM Module definition (section 6.2.6). Since these alerts are not triggered by the own module's intelligence (the agent responsible is the monitoring tool), the *Measurement Collector* will be required to handle data flow in both directions.

Apart from the previously commented notifications, the system monitoring tool might also be able to provide the *Measurement Collector* with performance data as another way for gathering results. This exchange of data may be done through text files. Leveraging the inclusion of the plugins in the design, the specific performance data that these programs provide may enhance significantly the analysis of the host or service check. The more detailed information, the better understanding we will have from the situation.

As well as the other couple of XIMM Modules, the DEM Module will additionally be required to support on-demand requests coming from the *Aggregation Module*. Therefore the system monitoring tool should be compliant with on-demand checks in order to obtain the latest status information about a host or service.

6.2.8.2 Generic Enablers Measurement Collector

The *Generic Enablers Measurement Collector* is a special case of the previously detailed collector in charge of gathering quite specific performance data; it is concretely oriented to those instances of FI-WARE GEs within the datacenters of the infrastructures and whose service is intended to be federated and monitored. However, the fact that the GEs developed within FIWARE do not provide by themselves monitoring interfaces, leads to the introduction of a monitoring tool that will provide information about the GEs through the VMs. The analysis of the tools needed for monitoring these services will be a key task in the XIFI monitoring scheme since these GE instances are the main element of the service that the federation is intended to provide.

As it was introduced in the description of the generic collector, the usage of VM/GE-oriented plugins within the system measurement tool may result in a meaningful contribution to be taken into consideration. It would enable monitoring of the VM/GEs performance in a much more precise manner. Systems that can be used for monitoring VM resources and services, related to specific GE performance include Nagios plugins (such as NRPE, nagios-virt, nagios-plugins-OpenStack, etc.) or even plugins from other monitoring tools, such as Collectd libvirt plugin.

6.2.8.3 DEM Measurement Archive

The *DEM Measurement Archive* is a component which might have a useful task within the module. Although the data is obtained from the system measurement tool, and the *Aggregation Module* may store the results that are being provided, it would be meaningful to keep internally the records that are being collected. Hence the *Measurement Archive* would be the actor in charge of such operation, being fed by the collectors and getting the data into a storage cache.

6.2.8.4 DEM API Handler

The *DEM API Handler* is the intermediary agent responsible for interpreting and dealing with Context Broker-format and DEM API-format requests and responses.

- According to the Context Broker model [13], the handler will be the *Context Producer* feeding the *Aggregation Module* (the actual *Context Broker*). Hence it must provide an interface compliant with FI-WARE NGSI-9 API [18] and especially FI-WARE NGSI-10 API [17] to interact with the Context Broker GE.
- On the other hand, it is also required to provide an interface to work with the *Measurement Collectors* and the *Measurement Archive* using the corresponding DEM API.

DEM Module is expected to support both 'push/pull' data transfer models regarding the *Aggregation Module*, which will be acting as the consumer of the data collected. A high-level overview is depicted in the following picture (Figure 52).

Figure 52: XIMM-DEM Dataflow overview

The *updateData* operation will be invoked by the *DEM API Handler* when updated data of status is available, 'pushing' the results to the client. At the same time, the *DEM API Handler* shall be able to handle queries from the *Aggregation Module*, requesting on-demand status of hosts or services.

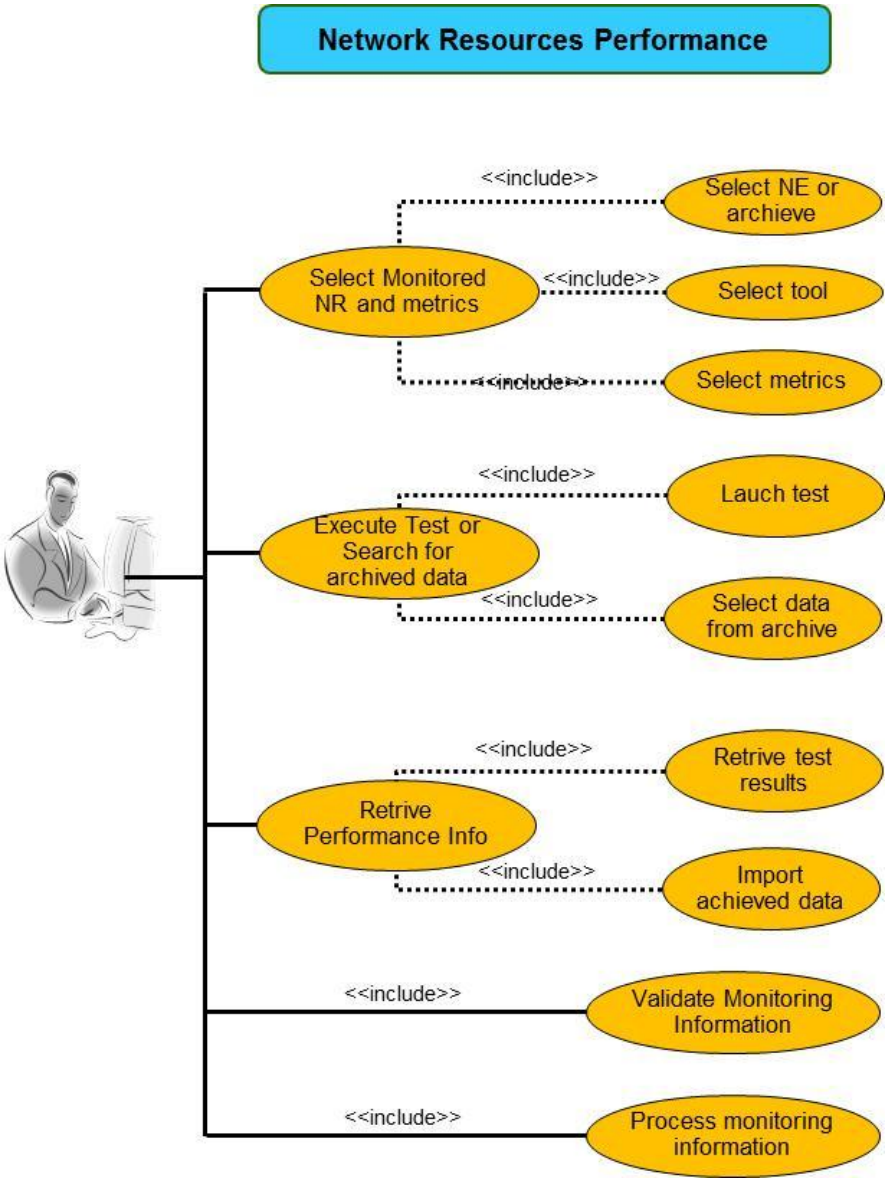
6.3 Scenarios of Use

6.3.1 Scenario 1: Monitoring the Network Resources Performance

Section	Description
Title & Description	Monitoring of the Network Resources Performance. The scenario is responsible for monitoring the Network Resources (NR) Performance. The whole lifetime of the NRs is considered, from the deployment and configuration, their operation, until their un-deployment.
Author(s)	Carlo Cinato, Luca Fantolino (TI)
Actors	The actors involved in this scenario include the following: • The administrator of the infrastructure that allows the monitoring of the NRs. • The user that asks (through the application such as the XIFI marketplace) for the monitoring information. • The XIMM infrastructure that support the monitoring operations. • The Network Elements that accommodates the NR assigned to the user. • The authentication and authorization mechanisms (such as IdM) that allow access for applications to the usage of the NRs. • The Network Monitoring tools involved in the measurements that command operations and gather monitoring data. • The Network Elements that are being monitored.
Objective	The objective of this scenario is (if needed) configure the measurements on the monitoring tools, monitor information on the NR, retrieve the monitoring information, consolidate it in a meaningful way and provide it to the application that requests it.

Pre-Conditions	The pre-conditions for this scenario to take place are the following: • The node of the infrastructure has joined the XIFI federation. • The Network Elements are available. • The monitoring tools has been deployed and configured. • The connection (including the interface) among the monitoring components and the application that asks for the monitor information has been in place.
Process Dialog	The sequence of steps detailing the interaction between the Actor and the System is the following: 1. The user of the application is authenticated and authorized to use the Network Monitoring component 2. The user asks for a list of the NRs Alternatively he may search for a specific NR. 3. The NR monitoring component indicates to the user the Network Elements involved (typically the end-point of a path) 4. The user asks for the list of Network performance metrics supported on the selected NR. Example of performance metrics are: • Connectivity (TRUE/FALSE outcome) • Path MTU • Packet Loss; • One-Way-Delay statistics (mean value, standard deviation) • One Way Jitter statistics • Sustainable TCP bandwidth • … 5. The application asks for one or more of the performance metrics described in step (4). 6. The NR monitoring component executes the required tests and/or retrieves the necessary information and performs preliminary (validity) checks, such as the type and the value range. If it has not been possible to retrieve the information, an error is generated. 7. The information is offered through the API to the application. In the case of error the application is notified appropriately.
Variations	Regarding step (3) instead of selecting a couple Network Element the application may select a (part of) sub-network and to retrieve mean performance metrics.
Post-Conditions	In case of successful execution, the application that invokes the scenario has retrieved the requested network resource performance indication. Several possible error may prevent the successful execution: • Error in authentication / authorization • Unavailability of resources (Network Element unreachable, monitoring tool unavailable, …) • Communication error between the application and the performance monitoring tool (protocol error, time-out, …)

Diagrams	**Use Case Diagram**



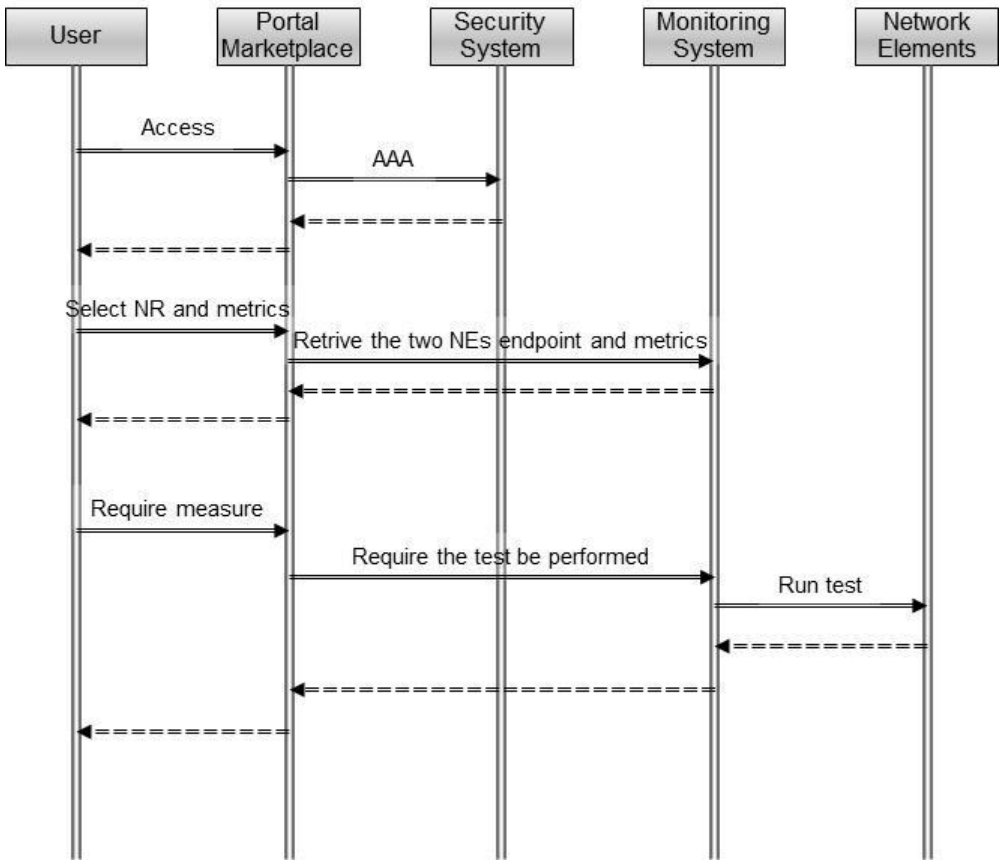
	Sequence Diagram 
Issues & Notes	The monitoring of the RM pose the following issues: • Different measurement tools use non uniform definition or mean of measure for the same metrics. For example jitter has several interpretation and may result in different values according to how the measure is performed. • The measure of throughput can be performed only if the NR is not in use. In addition this measure may impact on other NRs. So the request of such measure should carefully verified (e.g. special rights has to be granted to be able to require this measure)

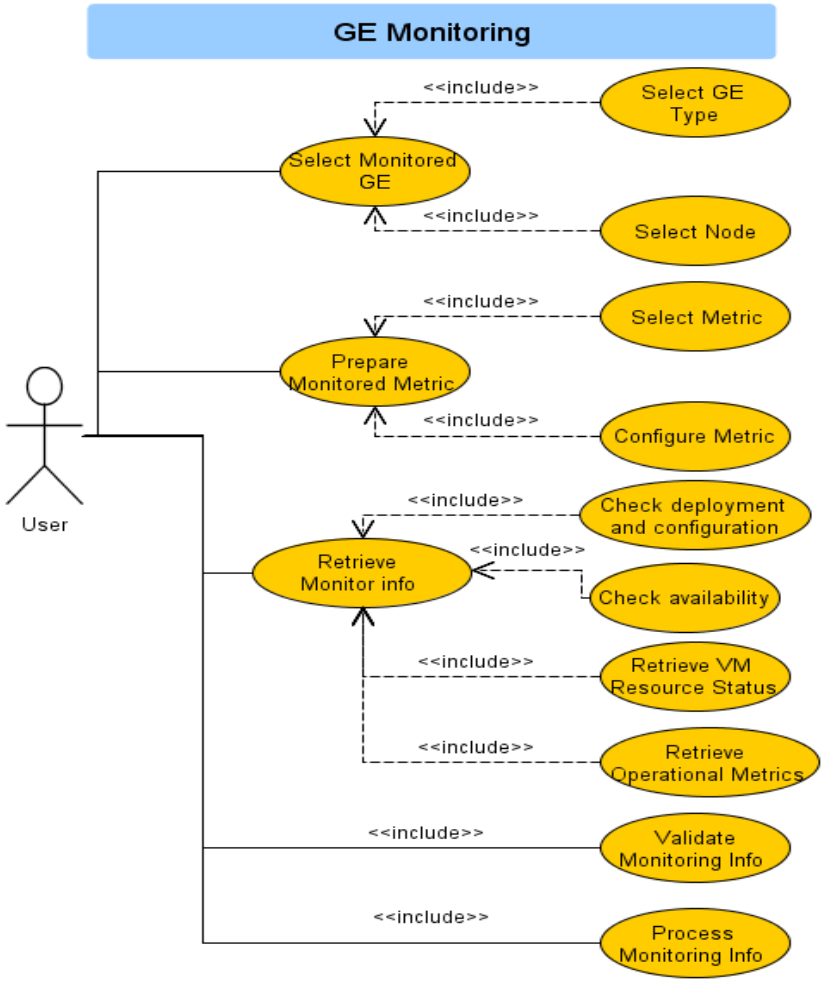
Table 10: XIMM Scenario 1

6.3.2 Scenario 2: Monitoring FI-WARE GEs

Section	Description
Title & Description	Monitoring FI-WARE GEs. The scenario is responsible for monitoring the GEs. The whole lifetime of the GEs is considered, from the deployment and configuration, their operation, until their un-deployment.
Author(s)	Andreas Papadakis (SYN), Panos Trakadas (SYN)

Actors	The actors involved in this scenario include the following: • The administrator of the infrastructure that allows the monitoring of the GEs. • The user that asks (through the application such as the XIFI marketplace) for the monitoring information. • The GE monitoring component that support the monitoring operations. • The VM that accommodates each GE. • The authentication and authorization mechanisms (such as IdM) that allow access for applications to the usage of the GEs. • The GE that is being monitored.
Objective	The objective of this scenario is to monitor information on the GE, retrieve the monitoring information, consolidate it in a meaningful way and provide it to the application that requests it.
Pre-Conditions	The pre-conditions for this scenario to take place are the following: • The node of the infrastructure has joined the XIFI federation. • The Cloud Chapter GEs that allow the operation of the GE have been deployed upon the node. • The GE (along with the products that are necessary for its operation) has been deployed and configured on the node. • The connection (including the interface) among the monitoring components and the application that asks for the monitor information has been in place.
Process Dialog	The sequence of steps detailing the interaction between the Actor and the System is the following: 1. The user of the application (e.g. the XIFI marketplace) is authenticated and authorized to use the monitoring component. 2. The user asks for a list of the GEs for monitoring to select. Alternatively he may search for a specific GE. 3. The GE monitoring component indicates to the user, the nodes where the GE (s) has (have) been deployed and configured. The user selects a node. At this point the pair (GE, node) has been selected. 4. The GE monitoring component presents to the user a set of possibilities regarding the monitoring information / checks. These include the following: • The check (verification) that the GE is properly deployed and configured on the node. This check is initially performed after the deployment and configuration of the GE, but it can also be performed at any time to verify that everything is in place (e.g. no dependencies are broken). • The check (verification) that the GE is available. The check of availability can be performed on demand or at regular intervals. • The count of the users (applications) that at the time make use of the GE (deployed in the selected node). This information does not come directly from the GEs themselves but from the horizontal authentication / authorization mechanisms. Since

	there are (currently unresolved) dependencies, this step is considered optional. • A set of monitoring metrics, provided by the tool which monitors the VM that accommodates the (selected) GE. • A set of monitoring metrics, provided by the tool which monitors the node that accommodates the (selected) GE. 5. The application asks for one or more of the checks / metrics described in step (4). 6. The GE monitoring component retrieves the necessary information and performs preliminary (validity) checks, such as the type and the value range. If it has not been possible to retrieve the information, an error is generated. 7. The information is offered through the API to the application. In the case of error the application is notified appropriately. 8. The user may ask for other information or for selecting other GEs and/ or nodes.
Variations	Variations can indeed exist in this scenario: • Regarding steps (2) and (3), the user may initially select the node and then the GE monitoring component will present the deployed GEs and after that the user makes the selection of the GE. • The scenario is modular, in the sense that not all involved operations may be invoked (information elements may be requested) during a single execution.
Post-Conditions	In case of successful execution, the application that invokes the scenario has retrieved the requested monitoring information. In case of unsuccessful information the scenario is terminated during an intermediary step. The cases of unsuccessful termination include the following: • Error in authentication / authorization. • Error in retrieving the requested information (reasons node, VM or responsible component is not available). • Retrieved information by the GE monitoring is erroneous (in terms of value type or accepted value range).

Diagrams	**Use Case Diagram** 

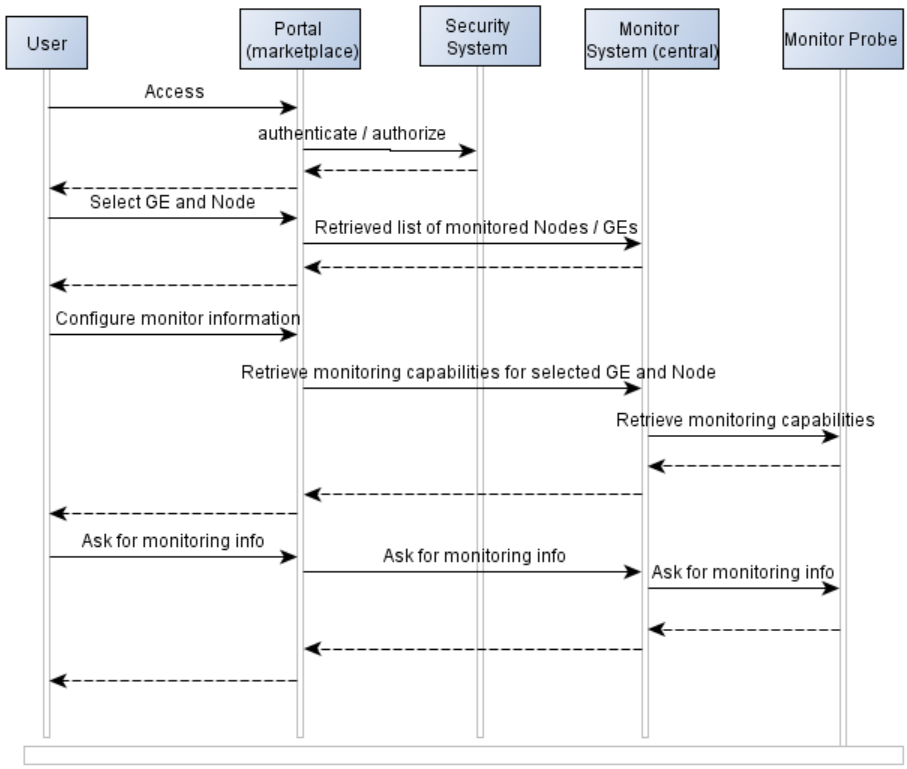
	Sequence Diagram 
Issues & Notes	The monitoring of the GE is a challenging task, considering the following issues: • The GEs themselves do not (and will not according to FIWARE considerations) provide a monitoring interface. This means that most of the information that is considered is provided by other components (the GE environment, the VM, the IdM etc.) • The load (and its quantitative estimation) is closely associated with the cloud environment and can be approached in an indirect manner. • The same tool is expected to be involved in both (4.d) and (4.e), while the metrics involved in these tasks may be a subset of the available monitoring metrics. • Some checks pre-suppose others (for example the availability check pre-supposes that the GE is correctly deployed and configured.

Table 11: XIMM Scenario 2

7 GE DEPLOYMENT AND CONFIGURATION ADAPTER

Due to the federated infrastructures nature of XIFI, there is a need for the design and development of an adapter that is capable of facilitating the management (deployment and configuration) of the Future Internet (FI) Generic Enablers (GEs) within the XIFI federation.

In other words, GEs Deployment and Configuration Adapter (GE-DCA) should deal with the Management of the Generic Enablers: 1) prior to, 2) during and 3) after installation on infrastructures (nodes) comprising the XIFI federation, to be offered to developers and users through the XIFI Portal.

At this point, it must be highlighted that the GE-DCA adapter does not deal with the management of the Cloud Hosting GEs provided by FI-WARE [12], as this subject is covered by other WPs in XIFI. Thus, GE-DCA takes as a prerequisite that these GEs have already been installed, configured and deployed on the XIFI nodes (see also the FMC Component Model provided by XIFI WP1 [66], along with relevant XIFI components and modules.

As it will become clear from the next sections, the goals of the GE-DCA adapter are:

- To ensure that resources are available prior to the deployment of a GEs to a XIFI node,
- To ensure that GEs will be easily and seamlessly deployed and configured on top of different infrastructures,
- To gather and store deployment information per GE and per XIFI node,
- To ensure that GEs information deployed on a single node will be automatically published to the XIFI federation in order to support add-on federation services developed in other XIFI taskforce.

7.1 Context

As discussed above, the Management of GEs follows the whole lifecycle, including management functionality (a) prior to, (b) during and (c) after the deployment of the GEs, as explained in the following:

- Prior to GE deployment on XIFI nodes

Prior to GE deployment, there is a need to verify that (hardware and software) resources required for the proper deployment and configuration of the GE are available within the host node. In this perspective, relevant information from several Cloud chapter GEs might be used, such as DCRM and IaaS SM.

- During GE deployment on XIFI nodes

According to the PaaS and SDC specifications and in other FI-WARE related documentation (i.e. “How to create recipes for the GEs” [21]), it is stated that “GE owner partner need to distribute their GEs installation and configuration software in an automatic way in terms of recipes”. This can lead to a cookbook, as guidance for GE deployment and configuration, resulting in concrete instructions on the required environments, products, artefacts, and blueprints in the Products catalogue and Product Instances Inventory [19] [20]. Although the development of the recipes (and cookbooks) is encouraged and ultimately needed, their availability is not guaranteed (at least within the time schedule of XIFI).

For that purpose, an alternative solution could be the deployment of a Virtual Machine image (at least for the simpler GEs) on top of different nodes. However, in the former case, GE-DCA must access the PaaS/SDC GEs and their corresponding inventories, while in the latter case, GE-DCA must access DCRM Image inventory.

- After GE deployment on XIFI nodes

After the deployment of a GE, the Management of GEs includes a monitoring module that will be capable of collecting measurements and inform interested XIFI components (such as Marketplace and SLA Management). We start from the fact that the GEs do not provide by themselves monitoring interfaces. This leads to the introduction of a monitoring tool that will provide information about the GEs, probably through the VMs. This component is in conformance to XIMM middleware solution, as described in section 6 of this deliverable and thus its functionality will not be covered in this section.

7.2 Architecture Description

Following the description of the goals and functionality of the GE-DCA adapter, an FMC compositional structure diagram is depicted below (Figure 53), highlighting the relations among the components related to GE-DCA. The upper box represents the Federation Platform (Master node), where "federation services", i.e. the services offered to support the management and provisioning of resources in a unified fashion are included, whereas the lower box represents a node of the XIFI federation, including all components needed for the proper operation of the node within the XIFI federation. It must be noticed that this diagram includes only the components and GEs that relate to the GE-DCA functionality. A complete FMC diagram of the XIFI federation is included in Deliverable 1.1 [66].

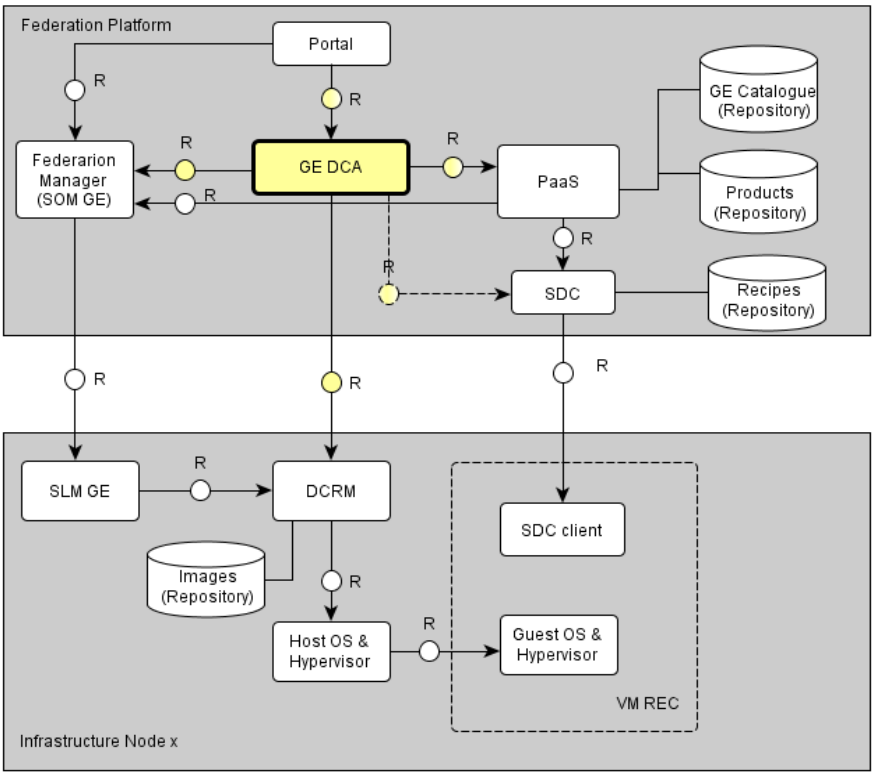


Figure 53: GE-DCA Architecture overview

According to Figure 53, XIFI Portal or any other component can check for resource availability within a specific node, deploy and configure a GE, request information regarding the GE instantiations running on a node or get information on the hosts where a specific GE is running. As it is obvious, GE-DCA adapter makes use of several FI-WARE Cloud chapter GEs, leveraging on the federation aspects of XIFI and contributing towards the checking of resource availability, the simplification of the deployment and configuration, and the provision of related information.

Starting from resource availability functionality, GE-DCA utilizes the DCRM API (e.g. *queryVirtualImages*, *getVirtualImageDetails*, *uploadVirtualImage*) to provide information through its RESTful API.

In case that GE-DCA is instructed to deploy a GE using recipes, it utilizes the API provided by PaaS (*enviromentInstance*, *applicationInstance*) and SDC (*productInstance*, *updateProductVersion*, *reconfigureProduct*), combining methods and querying the respective Repositories. It is noted that it is possible to use SDC API directly (either based on Puppet or Chef configuration management tools), bypassing PaaS GE, given that all information regarding environments, products and recipes is available.

From the aforementioned analysis it is noticed that GE-DCA not only provides a uniform API definition for managing the deployment and configuration of GEs, but, more importantly, collects and stores information for the federated XIFI infrastructure as a whole.

7.3 Description of GE-DCA Components

In order to achieve its goals, GE-DCA consists of several components:

- **DC API handler**: This component is responsible for handling the API operations related to resource availability check as well as deployment and configuration of a GE, either through a recipe or as a virtual image. Details on the DC API operations can be found in section 9.6.
- **DC Info API handler**: This component is responsible for handling the API operations related to information regarding the specific nodes within XIFI federation that the GEs are deployed and configured as well as the types of the GEs that are deployed on a specific node. Details on the DC API operations can be found in section 9.6.
- **GE-DCA Logic**: GE-DCA Logic can be seen as the orchestrator of all operations and functionality related to the GE-DCA adapter. Among its tasks, the GE-DCA Logic is responsible for managing data stored in the DC Registry, handling multiple requests/responses, managing communication with PaaS, SDC and DCRM GEs.
- **Deployment and Configuration Registry**: DC Registry is an important component of the GE-DCA adapter, responsible for storing data for installed GEs on the XIFI nodes (federation). Moreover, this component stores and updates information regarding GEs requirements, ranging from hardware and software requirements to knowledge of whether recipes or virtual images are present and where. This feature makes this component useful as the mechanism for the update of this type of knowledge (on behalf of the DCA) is currently manual (since, to our knowledge, FIWARE catalogue does not provide automatic updating capabilities).
- **Resource Check**: This component is responsible for the proper handling between GE-DCA and DCRM.
- **Recipe DC**: This component is responsible for the proper handling between GE-DCA and PaaS.

- **Image DC**: This component is responsible for the proper handling between GE-DCA and DCRM.

Figure 54 presents a graphical representation of the GE-DCA components.

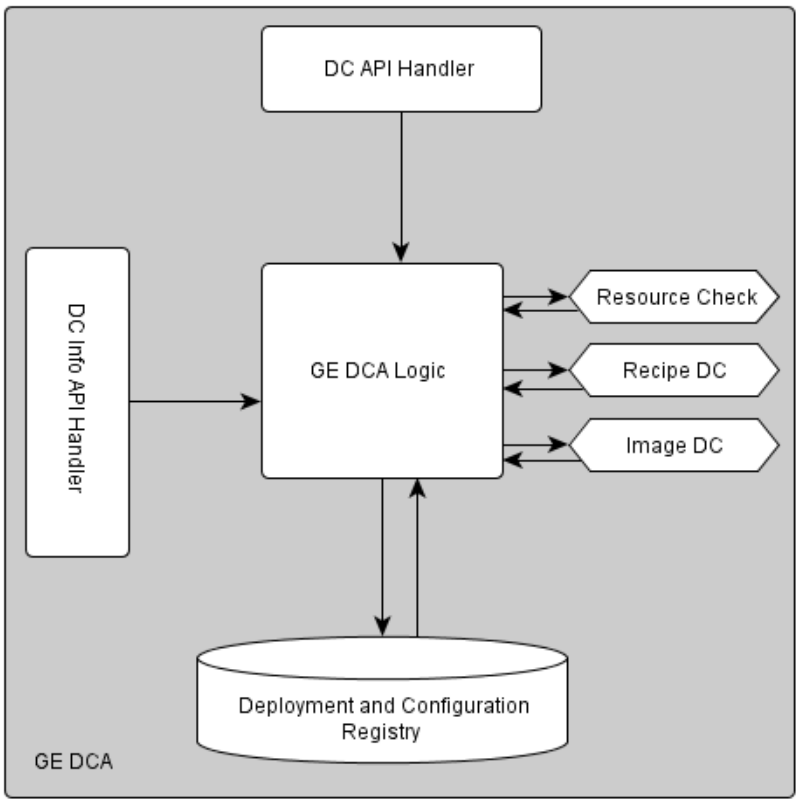


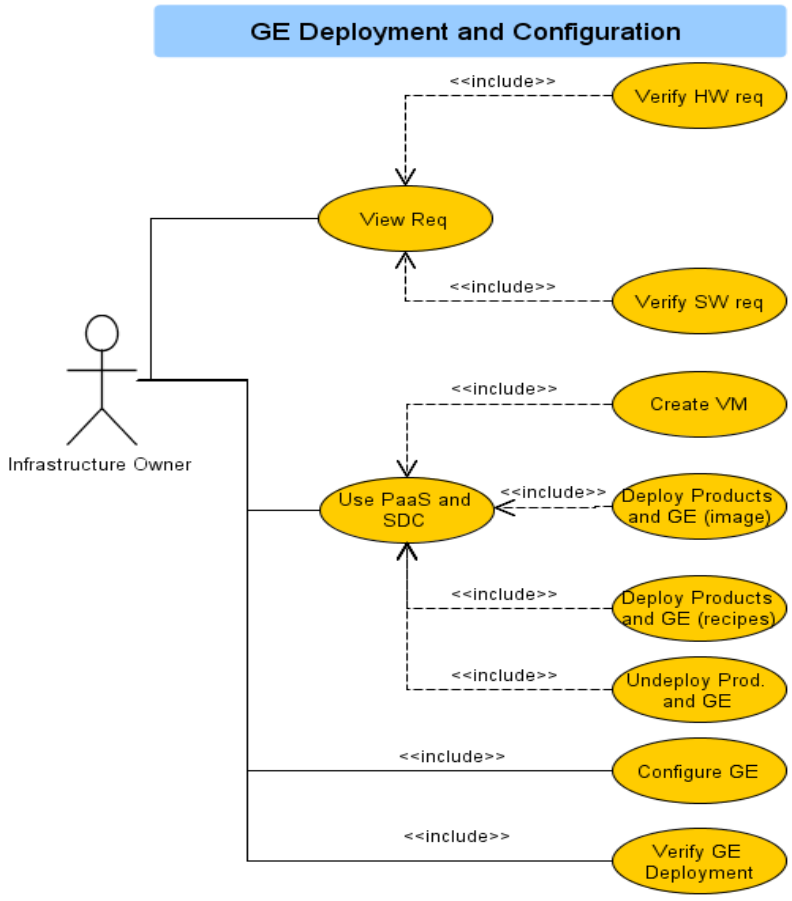
Figure 54: GE-DCA Internal architecture

7.4 Scenarios of Use

7.4.1 Scenario 1: Deployment and Configuration of a GE

Section	Description
Title & Description	Deployment and Configuration of a Generic Enabler. The administrator of the infrastructure wishes to deploy and configure a Generic Enabler on a node that belongs to the XIFI federation. Prior to the deployment the Generic Enabler is available in a packaged form in the repository. The node possesses the necessary hardware resources for the deployment of the GE.
Author(s)	Andreas Papadakis (SYN), Panos Trakadas (SYN)
Actors	• Infrastructure administrator • GE Repository

Objective	The owner wishes to deploy and configure a Generic Enabler upon a node of his infrastructure and make it available through the XIFI portal.
Pre-Conditions	The pre-conditions of the scenario are the following: • The infrastructure owner has joined the XIFI federation. This means that the XIFI tools are available on the infrastructure. • The necessary FIWARE GEs are available. These include the Cloud Chapter GEs (DCRM, IaaS SM) and specifically the PaaS Manager and the SDC GEs. Furthermore they (the PaaS Manager and SDC) can make available the software resources (products) that are necessary for the deployment of the GE. • The necessary hardware resources for the deployment of the GE are available on the node. • The GE to be deployed is available in the Repository. We consider two cases: o The GE, along with the products (OS, DB, other libraries) that support its operation, is pre-packaged as downloadable image. o The GE is accompanied by a recipe that allows the deployment and configuration of the products (that are necessary for its operation) and the GE through the PaaS Manager and the SDC.
Process Dialog	The sequence of the steps performed by the administrator is the following: 1. The administrator is authenticated and authorized. He is granted access to the node, which is part of the XIFI federation. 2. The administrator accesses the selected GE, through the Repository. The administrator is informed of the hardware and software requirements of the GE. 3. The administrator verifies the availability of the hardware resources that are necessary for the selected GE. In case that the resources are unavailable, the scenario is terminated. 4. The administrator uses the PaaS Manager to prepare the environment for the deployment of the GE. The PaaS Manager (with the co-operation of the SDC) is responsible for the following steps. 5. A Virtual Machine (dedicated to the GE) is created. 6. The products that support the operation of the GE are made available. We consider two possibilities (a) and (b) for performing this step: • The necessary products are packaged as image (available for download in the Repository) and they are deployed on the node • The recipe of the GE is available and this guides the deployment and the configuration of the necessary products, being downloaded from the Repository, using PaaS Manager and SDC GEs. This step can be executed multiple times, depending on the number of the necessary products. This step includes the deployment and configuration of the GE. In case that one software resource is unavailable or cannot be configured or deployed the scenario is terminated. 7. The scenario terminates.

Variations	The variations can be the following: • The way the products and the GE are deployed and configured as a Package or through a recipe. • There may be GEs that cannot be deployed and configured in an automatic way. These may include the Cloud Chapter GEs (which are expected to be deployed manually) or the other GEs with more specific requirements (e.g. from the IoT area).
Post-Conditions	The end result of the scenario is the availability of the selected GE in case of successful execution. In case of unsuccessful execution, we may have the following end results: • Termination due to unavailability of hardware resources. • Termination due to unavailability (in terms of deployment and configuration) of software resources. • Termination due to inability to deploy and configure the GE.
Diagrams	**Use Case Diagram** 

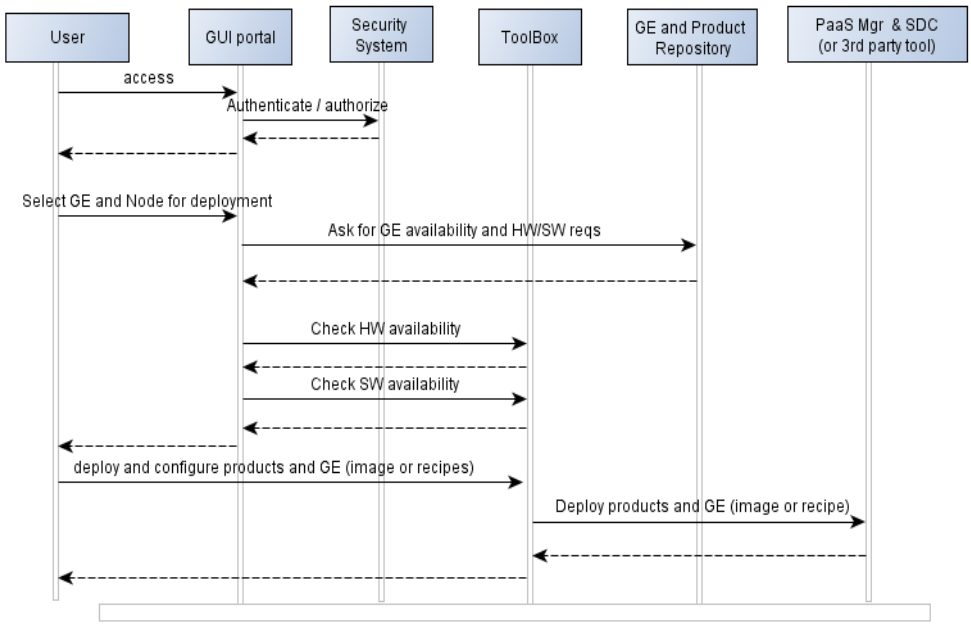
	Sequence Diagram 

Table 12: GE-DCA Scenario 1

8 INFRASTRUCTURE TOOLBOX

The operation of deploying 10 or 100 servers and configuring them correctly to run FI-WARE DCRM [14] and XIFI tools to connect a node to the federation is not a trivial task. In XIFI, the overall number of servers will easily hit 100 or more. It is also very important to ensure that the "maintenance and up-date" process are simplified as much as possible. Thus a high-degree of automation for these activities is highly recommended. In FI-WARE the natural focus, so far, was mainly on automating the installation of GEs on top of the Cloud, thus the problem of moving from a testbed to a more production type environment for managing DCRM set-up (and hence OpenStack set-up) was not considered. In XIFI we will focus on this activity to simplify as much as possible installation also for new nodes and private clouds by UC Trials. The tool that will support this process is the **Infrastructure Toolbox**.

8.1 Context

As previously said, automating the installation of a server is essential when the number of servers to be installed is relevant as in the XIFI context. What is required is a tool that can provide a ready-to-use server (with the operating system, the hypervisor and the cloud management software) starting from the bare bone metal.

Some specific requirements that shall be satisfied within a node are:

- Ability to automate the installation of host operating system, hypervisor and cloud software through the Preboot eXecution Environment (PXE).
- Ability to define and select a deployment model among the ones available (e.g. with or without high availability).
- Ability to discover the servers where to install the software.
- Ability to specify a "role" (controller, storage, compute etc) for each server.
- Ability to set up network configuration (e.g. VLAN).
- Ability to test the deployment so to verify that everything has been installed correctly.

In the following section a proposal for the architecture of the Infrastructure Toolbox is provided.

8.2 Architecture Description

In order to support the installation, updating and managing of XIFI nodes, we have designed a modular tool, named Infrastructure Toolbox (ITBox), where its architecture is shown in Figure 55. The architecture design has been driven by the analysis of existing tools that allow bare-metal deployment of IaaS platforms.

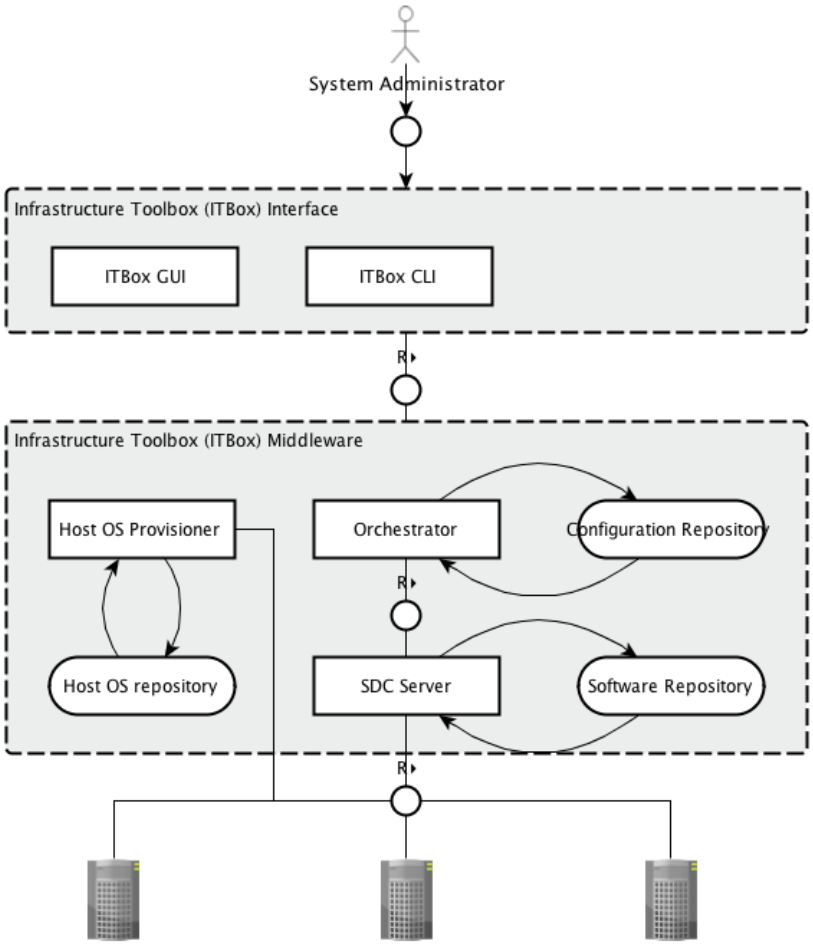


Figure 55: Infrastructure Toolbox architecture

The Infrastructure Toolbox architecture is composed by an interface layer, represented by the upper square and named Infrastructure Toolbox Interface and by a middleware layer, represented by the lower square and named Infrastructure Toolbox Middleware.

We now see in detail how these two layers are composed.

- **ITBox Interface**. That provides the System Administrator interfaces to run the set-up and installation of a new XIFI node. In particular the Interface provides:
    - ITBox Graphical User Interface. A web based interface with simplified functionalities for set-up and installation.
    - ITBox Command Line Interface. A shell based interface with advanced functionalities for set-up and installation.
- **ITBox Middleware**. That provides the services that run the actual provision and deployment of OS and services on top of bare-metal infrastructure.
    - The Host OS Provisioner. A server that installs via network OS on node discovered via PXE protocol or similar. It has a repository of Host OS that can be provisioned on

the bare-metal servers.

- The Orchestrator. A server that coordinates the deployment of services on the different servers according to the configuration passed by the ITBox Interface. The orchestration among different servers is needed to ensure proper set-up and configuration of the XIFI node (e.g. how to know when a cloud controller server is ready to register compute servers?). The orchestrator leverage on a repository of scripts.

- **Software Deployment and Configuration (SDC)**. A server that provides access to packages and scripts for the installation of services on a single node. It relies on a Software and Scripts repository. (note: this service has same role has the FI-WARE SDC GE, but a different scope, i.e. it is not meant for the installation of GEs on top of VMs provisioned by the DCRM, but for the installation of DCRM itself and additional services).

8.3 Scenarios of Use

8.3.1 Scenario 1: Set-up and installation of a new XIFI node

Section	Description
Title & Description	Set-up and installation of a new XIFI node. The administrator of the infrastructure wishes to set-up its node for the connection to XIFI federation.
Author(s)	Federico M. Facca (CREATE-NET), Alessandro Martellone (CREATE-NET)
Actors	Infrastructure administrator
Objective	Set-up a new node for connection with XIFI federation.
Pre-Conditions	• Infrastructure administrator (IA) agrees on XIFI Federation terms and conditions. • Infrastructure administrator (IA) has configured correctly the Infrastructure Toolbox. Servers are just bare bone metal (without any OS) but are connected to the same network.
Process Dialog	1. The IA accesses to the Toolbox by ITBox Interface. 2. The IA selects to deploy a new node. • The IA runs “Prepare” new servers. o The IA selects the servers to be installed. o The Infrastructure administrator selects a network OS hosted in the ITBox Middleware and boot it on the server through PXE protocol or similar. 3. Once booted, IA selects “Discover” function, and the Host OS Provisioner discovers the servers. Now, these servers are available to be part of the "XIFI node" configuration process. • The IA decides, among the different Host OSes, the one for the XIFI node - this OS will be later provisioned on all servers. o Then he selects among a number of predefined "Deployment Models" the one for his/her XIFI node (different deployment are

	available to better leverage on hardware heterogeneity, node purpose and scale). ○ Once selected the reference "Deployment Model" for its XIFI node, he needs to assign discovered Servers to available roles for the selected "Deployment Model" (for a discussion on deployment models see D5.1). ○ As last step, other parameters, related to network, monitoring framework, and so on, are required to be configured. 4. Once completed the configure step, the IA can start the provision phase. At this step, the Host OS Provisioner installs the selected Host OS on all configured Servers. The Host OS include as well services to allow the Orchestrator and SDC Server to control the deployment phase. 5. Completed this stage, the "deployment" phase starts: according to the deployment model selected and to the role assigned to servers. The Orchestrator coordinates the installation of services on the single servers and their configuration. The actual installation follows the guidelines of the Orchestrator. It relies on the SDC Server that host deployment scripts and information on software repositories.
Post-Conditions	All the servers has been installed and configured with the Operating System and the basic cloud management software, ready to host VMs and the installation of the rest of FI-WARE GEs.
Issues & Notes	Software repositories may be internal to XIFI (e.g. for the network adapters and the monitoring adapters), internal to FI-WARE (e.g. for the DCRM) or external (e.g. for the common OpenStack components).

Table 13: Infrastructure Toolbox Scenario 1

8.3.2 Scenario 2: Test of installation of a new XIFI node

Section	Description
Title & Description	Test of installation of a new XIFI node. The administrator of the infrastructure wishes to test its node just installed.
Author(s)	Federico M. Facca (CREATE-NET), Alessandro Martellone (CREATE-NET)
Actors	Infrastructure administrator
Objective	Test a new node installation for connection with XIFI federation.
Pre-Conditions	• Infrastructure administrator (IA) agrees on XIFI Federation terms and conditions. • Infrastructure administrator (IA) has configured correctly the Infrastructure Toolbox and executed the previous scenario.
Process Dialog	1. The IA accesses to the Toolbox by ITBox Interface. 2. The IA selects to “Test node”.

	• The IA selects one or more servers installed and run "Test node". 3. The Host OS Orchestrator obtains from SDC Server a set of preconfigured scripts test, and launches them. 4. Once completed the test step, ITBox Interface shows the results.
Post-Conditions	The IA can analyze the results and verify of the installation of the node has been completed correctly.

Table 14: Infrastructure Toolbox Scenario 2

9 APIs SPECIFICATION

9.1 XIFI Network Services Management (NSM) APIs Specification

The XIFI Network Management Controller aims at providing the interface to provisioning networking resources within and across XIFI nodes. The main API is based on the OpenStack Networking API 2.0 defined in OpenStack [52]. Networks are not restricted to a single Datacenter (XIFI Node) instance, and can expand several XIFI nodes.

OpenStack Networking API is based on RESTful interface.

9.1.1 Topology API

In order to facilitate the management of the network infrastructure in XIFI, the XIFI network controller exposes a Topology API to the XIFI infrastructure manager. It exposes the internal details of the connectivity inside XIFI nodes and between XIFI nodes.

The Topology API can be useful for other XIFI services that need to know the switches (physical or virtual) available in XIFI.

Operation	HTTP Method	Path
Get Full Topology in XIFI Node	GET	{BaseUri}/GetFullTopoXifiNode
Get Switch List in XIFI node	GET	{BaseUri}/GetSwitchListXifiNode
Get Link List in XIFI Node	GET	{BaseUri}/GetLinkListXifiNode
Get Full Topology of XIFI Federation	GET	{BaseUri}/GetFullTopoXifiFed
Get Inter XIFI Node Link List	GET	{BaseUri}/GetInterXifiLinkList
Get Switch Details	GET	{BaseUri}/GetSwitchDetails
Get Link Details	GET	{BaseUri}/GetLinkDetails

Table 15: XIFI Topology API operations

9.1.1.1 Get Full Topology of XIFI node

This operation is used to retrieve the whole Topology, including switches (physical and virtual) and links within a XIFI Node.

- **Resource:** /GetFullTopoXifiNode/{XIFI_Node_id}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Full Topology details	JSON
500	Internal Server Error	none	none
503	XIFI node not present	none	none

9.1.1.2 Get Switch List of XIFI node

This operation is used to retrieve the list of switches (physical and virtual) present in a XIFI Node that can be controlled through the Networking API.

- **Resource:** / GetSwitchListXifiNode /{XIFI_Node_id}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of Switch IDs	JSON
500	Internal Server Error	none	none
503	XIFI node not present	none	none

9.1.1.3 Get Link List in XIFI Node

This operation is used to retrieve the list of L2 links in a XIFI node. The availability of Traffic Engineering Information depends on the switch capabilities.

- **Resource:** / GetLinkListXifiNode/{XIFI_Node_id}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of Links with details	JSON
500	Internal Server Error	none	none
503	XIFI node not present	none	none

9.1.1.4 Get Full Topology of XIFI Federation

This operation is used to get the full list of switches and links of the XIFI Node federation.

- **Resource:** /GetFullTopoXifiFed
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Full Topology details	JSON
500	Internal Server Error	none	none

9.1.1.5 Get Inter XIFI Node Link List

This operation is used to retrieve the list of L2 links between XIFI nodes. Depending on the cases, there can be no L2 connectivity and thus, only L3 connectivity will be possible between those XIFI nodes.

- **Resource:** / GetInterXifiLinkList /{XIFI_Node_id_source}-/{XIFI_Node_id_destination}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of Links with details	JSON
500	Internal Server Error	none	none
503	XIFI node not present	none	none

9.1.1.6 Get Switch Details

This operation can be used to get information of a particular XIFI node given a switch_id.

- **Resource:** / GetSwitchDetails /{XIFI_Node_id}/{Switch_id}
- **Method:** GET
- **Query string parameters:** information to be queried or ALL if all information is needed.
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Switch details	JSON
500	Internal Server Error	none	none
503	XIFI switch not present	none	none

9.1.1.7 Get Link Details

This operation can be used to get information of a particular XIFI link, from its id. The id can be obtained after querying the link list.

- **Resource:** / GetSwitchDetails /{Link_id}
- **Method:** GET
- **Query string parameters:** information to be queried or ALL.
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Link details	JSON
500	Internal Server Error	none	none
503	XIFI Link not present	none	none

9.1.2 Internal XIFI Network Controller API

Internally, the XIFI Network Controller uses an API to setup L2 connections (point to point, point to multi-point or multi-point to multi-point). This API is intended to be used only internally between the XIFI Controller Neutron Plugin and the Controller itself running in the XIFI Master Node. However, it could be used for provisioning without going through OpenStack Networking API if needed.

9.1.2.1 Creation of a point-to-point link between hosts

- **Method:** PUT
- **Query string parameters:** information to be queried or ALL.
- **Request body:** this operation requires a request body, setting the following attributes

Attribute	Description
source	Source switch
destination	Destination switch
source_mac	Mac of the source host
dest_mac	Mac of the destination host
source_interface	Port through which the source port is connected to the source switch
destination_interface	Port through which the destination port is connected to the destination switch
operation	Type of operation to be done. In this case VIRTUAL_LAN_PATH_PROVISIONING

- **Possible responses:**

Code	Description
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not found

9.2 XIFI Infrastructure Monitoring Middleware (XIMM) APIs Specification

The **XIFI Infrastructure Monitoring Middleware (XIMM)** (section 6) is an abstraction layer which aims to provide homogeneous monitoring services by collecting and handling network and datacenter performance data from multi-domain infrastructures. Such piece of software will be designed to deal with the complexity and heterogeneity inherent in distributed systems. XIMM will act as the client and discoverer of multi-domain monitoring metrics (subject to locally-determined policy restrictions) to feed aggregation modules at federation level.

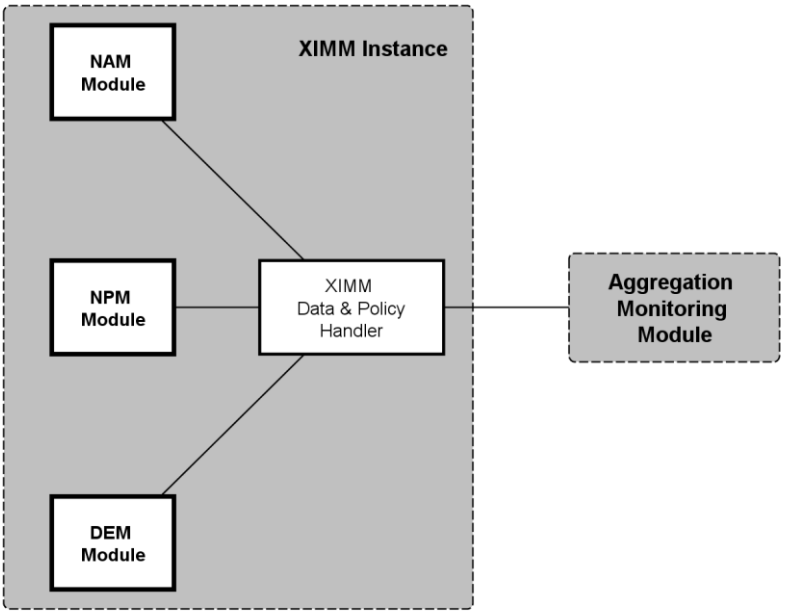


Figure 56: Main components in a XIMM instance

Figure 56 depicts the main components of a XIMM instance. This modular scheme will enable a better approach when specifying the architecture and operations. The *Data & Policy Handler* will just be an intermediary agent between the main modules and the *Aggregation Module*, providing a single and controlled entry point to the instance. The aggregator will act as the client in charge of requesting the monitoring data, processing it and providing added-value services at federation level.

It is expected that this *Aggregation Module* will be based on the Context Broker GE provided by FI-WARE [13]. According to the terminology of this scenario, each one of the three modules should perform the role of a Context Producer. Therefore, the modules will be required to be compliant with XIMM APIs and OMA NGSI Context Management APIs [40]. This last point is out of the scope of this specification and will not be addressed.

9.2.1 XIMM APIs Core

The XIMM APIs are RESTful, resource-oriented APIs accessed via HTTP that uses XML-based representations for information interchange. Each one of the main modules (NAM, NPM and DEM) will have associated their own API specification, forming the core set of operations to be handled:

- XIMM-NAM API Specification (section 9.3)
- XIMM-NPM API Specification (section 9.4)
- XIMM-DEM API Specification (section 9.5)

Functionality will be enhanced through general operations carried out by the *Data & Policy Handler*.

9.2.2 Intended Audience

In order to use these specifications, it is critical that the reader has a general understanding of the XIFI Infrastructure Monitoring Middleware (XIMM) description as well as each one of its functional modules:

- XIMM-Network Active Monitoring (XIMM-NAM) Module (section 6.2.6)
- XIMM-Network Passive Monitoring (XIMM-NPM) Module (section 6.2.7)
- XIMM-Datacenter & Enablers Monitoring (XIMM-DEM) Module (section 6.2.8)

It might also be recommended to be familiar with:

- RESTful web services
- XML data serialization formats
- FI-WARE Platform

9.2.3 Endpoints Format

With the aim of preserving the API separation, it is supported different endpoint configuration for each of the underlying APIs. Each API will have its own endpoint with the following URL template: *http://{serverRoot}:{serverPort}/vx.x*

For reference we will identify them in the following with these endpoints:

API	BaseURI
NAM API	http://{namAPIserver}:{namAPIport}/vx.x
NPM API	http://{npmAPIserver}:{npmAPIport}/vx.x
DEM API	http://{demAPIserver}:{demAPIport}/vx.x

Table 16: XIMM APIs BaseURIs

9.2.4 Resources Specification

Resource	Description
Node	This abstract resource will represent the concrete node/infrastructure belonging to the XIFI federation that will be considered in a specific period of time.
Instance	Each node will have deployed a set of XIMM instances in different hosts of interest which are requested to provide monitoring data. The module responsible for managing each instance is the Data & Policy Handler contained within
Measurement	According to the figure that depicts a XIMM instance, the functional

Service	modules will contain a set of Measurement Services in charge of both collecting data from monitoring tools and handling the access to measurement storage caches
Measurement Data	The results fetched from the monitoring tools which shall be handled by the Measurement Services
Status	This resource is in charge of pointing the current status of a specific Measurement Service or Instance, informing if it is operational
Start Time	If a test is triggered due to a regularly scheduled operation, this resource will signal the reference time
Stop Time	Based on the previous case, this resource would set the temporal reference to stop the test

Table 17: XIMM APIs Resources

9.2.5 Authentication and Authorization Operations

Within XIMM Data & Policy Handler, the Access Handler shall enforce the restrictions and control the access to the resources, performing the role of an authorization agent. Each HTTP request against the XIFI Infrastructure Monitoring Middleware (XIMM) needs to contain specific authentication credentials. The specific implementation of this API should use OAuth or Token. Authentication tokens are only available to users who have administrative rights in each node, and those tokens only authorize operations within node. All authentication schemes require that the API operate using SSL over HTTP (HTTPS).

9.2.6 Discovery Operations

XIMM Data & Policy Handler may also contain a Local Discovery component in charge of exposing service's capabilities to outside. Even considering an Aggregation Module, it needs that the available services register with it to become visible. Hence, this module may pre-process the data and provide it through a compact format. This module supports a number of operations:

- *DiscoveryQueryRequest* - Allows for queries, written in XQuery, to be performed.
- *DiscoveryUpdateRequest* - Allows a service to refresh information.

The API makes access to these messages available, and returns results in easier to handle mediums beyond raw XML messages.

Discovery operations will be considered as generic operations for the NAM, NPM and DEM modules, being the 'Data & Policy Handler' the module responsible.

Operation	HTTP Method	Path
List all instances	GET	{BaseUri}/instances
Get a specific instance	GET	{BaseUri}/instances/instance_id
Get a specific module	GET	{BaseUri}/instances/instance_id/module_id

Table 18: XIMM API Discovery operations

9.2.6.1 List all instances

- **Resource:** /intances
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Instances filtering descriptor	XML/JSON

- **XML "filtering descriptor" body request example:**

```
<filter>
    <instances>
        <instanceId>0111</instanceId>
        <instanceId>0112</instanceId>
        <instanceId>0112</instanceId>
    </instances>
</filter>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of intance available	XML/JSON
500	Internal Server Error	none	none
502	Data mismatch	none	none

- **XML OK Response body example:**

```
<intancesIds>
      <intanceId>1111</intanceId>
      <intanceId>1122</intanceId>
      <intanceId>444</intanceId>
</intancesIds>
```

9.2.6.2 Get a specific instance

- **Resource:** /instances/{instance_Id}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	instance details	XML/JSON
500	Internal Server Error	none	none
503	Instance not available	none	none

- **XML OK Response body example:**

```
<instance>
        <instanceId>1111</instanceId>
        <instanceData>
            <domain>domIta</domain>
            <module>External_BWCTL</module>
            <module>Measurement_Archive</module>
        </instanceData>
</instances>
```

9.2.6.3 Get a specific module

- **Resource:** /instances/{instance_Id}/module_id
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	module details	XML/JSON
500	Internal Server Error	none	none
503	Module not available	none	none

- **XML OK Response body example:**

```
<instances>
    <instanceId>0111</instanceId>
    <moduleId>2111</moduleId>
        <moduleData>
            <domain>domIta</domain>
            <module>External_BWCTL</module>
```

```
        </moduleData>
</instances>
```

9.3 XIMM-NAM API Specification

The XIMM-NAM module exposes an internal API towards the XIMM Data & Policy Handler. The same API is exposed externally by XIMM Data & Policy Handler to provide standard access to clients who need monitoring functionalities from XIFI infrastructure.

The XIMM-NAM API is related to Network Active Monitoring functionalities only, both historical and on-demand measurements. The NAM Module behind the NAM API Handler manages the different sources of measurement data and the distribution of modules and data in a transparent manner, hence the API client does not need information of the way the data are collected and stored and how the XIMM platform is distributed.

9.3.1 API Operations

The XIMM-NAM API supports a number of operations which include:

- **Endpoint Operations** - The Endpoint resources represents an endpoint for which measurement parameters can be produced.
- **Measurements Operations** - The Measurements resources allows the user to consume measurement parameters.

9.3.2 Endpoint Operations

Endpoint is the network resource where the monitoring functionalities are performed and from where the data measurements are collected. Thus each measurement performed by a monitoring tool has the corresponding endpoint data in the XIMM-NAM API. The endpoint contains the details description of the measure requested, as described in the following.

- Endpoint data: Network resource information from where the measurements are performed and collected. It can be a physical port or a logical port.
- Remote endpoint data: Remote network resource information where the NAM measurements are terminated. It can be a physical port or a logical port.
- Time information: start time and stop time define the start time and stop time of the measure requested. If start time is missing the meaning is “starts from now”; if stop time is missing the meaning is “never stop”. Interval is the measures aggregation period.
- Measures information: lists the measures performed on the specific endpoint. It contains the measures characteristics, if any.

Operation	HTTP Method	Path
Setup a new endpoint	POST	{BaseUri}/endpoints
List all endpoints	GET	{BaseUri}/endpoints
Get a specific endpoint	GET	{BaseUri}/endpoints/endpoint_id
Delete a specific endpoint	DELETE	{BaseUri}/endpoints/endpoint_id

Table 19: XIMM-NAM API Endpoint operations

9.3.2.1 Setup a new endpoint

- **Resource:** /endpoints
- **Method:** POST
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Endpoint setup's details descriptor	XML/JSON

- **XML descriptor body request example:**

```
<descriptor>
  <endpointData>
      <domain>domIta</domain>
      <device>switchA2</device>
      <interface>5.2.0</interface>
      <logicalInterface>300</logicalInterface>
  </endpointData>
  <remoteEndpointData>
      <domain>domEsp</domain>
      <device>32442</device>
      <interface>2.0.0</interface>
      <logicalInterface>150</logicalInterface>
  </remoteEndpointData>
  <time>
      <startTime>2013-08-25 18:00</startTime>
      <stopTime></stopTime>
      <interval>15</interval>
  </time>
  <measuresData>
      <measure>
          <measureType>oneWayDelay</measureType>
          <pingInterval>10>/pingInterval>
          <pingNumber>100</pingNumber>
      </measure>
```

```
        <measure>
            <measureType>oneWayPacketLoss</measureType>
        </measure>
    </measuresData>
</descriptor>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Endpoint identifier	XML/JSON
500	Internal Server Error	none	none
501	Network resources not present	none	none
502	Data mismatch	none	none

9.3.2.2 List all endpoints

- **Resource:** /endpoints
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Endpoint filtering descriptor	XML/JSON

- **XML "filtering descriptor" body request example:**

```
<filter>
    <endpointData>
        <domain>domIta</domain>
    </endpointData>
    <remoteEndpointData>
        <domain>domEsp</domain>
    </remoteEndpointData>
    <measuresData>
        <measure>
            <measureType>oneWayDelay</measureType>
        </measure>
        <measure>
            <measureType>oneWayPacketLoss</measureType>
        </measure>
    </measuresData>
</filter>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of endpoint identifiers	XML/JSON
500	Internal Server Error	none	none
502	Data mismatch	none	none

- **XML OK Response body example:**

```
<endpointIds>
        <endpointId>1111</endpointId>
        <endpointId>1122</endpointId>
        <endpointId>444</endpointId>
</endpointIds>
```

9.3.2.3 Get a specific endpoint

- **Resource:** /endpoints/{endpoint_Id}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Endpoint details	XML/JSON
500	Internal Server Error	none	none
503	Endpoint not present	none	none

- **XML OK Response body example:**

```
<endpoints>
    <endpoint>
        <endpointId>1111</endpointId>
        <endpointData>
            <domain>domIta</domain>
            <device>switchA2</device>
            <interface>5.2.0</interface>
            <logicalInterface>300</logicalInterface>
        </endpointData>
        <remoteEndpointData>
            <domain>domEsp</domain>
            <device>32442</device>
            <interface>2.0.0</interface>
            <logicalInterface>150</logicalInterface>
        </remoteEndpointData>
        <time>
            <startTime>2013-08-25 18:00</startTime>
            <stopTime></stopTime>
```

```
            <interval>15</interval>
            </time>
        <measuresData>
            <measure>
                <measureType>oneWayDelay</measurement>
                <pingInterval>10>/pingInterval>
                <pingNumber>100</pingNumber>
            </measure>
            <measure>
                <measureType>oneWayPacketLoss</measureType>
            </measure>
        <measuresData>
    </endpoint>
</endpoints>
```

9.3.2.4 Delete a specific endpoint

- **Resource:** /endpoints/{endpoint_id}
- **Method:** DELETE
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	none	none
500	Internal Server Error	none	none
504	Endpoint not present	none	none
505	Endpoint not deleted	none	none

9.3.3 Measurement operations

Measurement is the resource that allows the user to get the monitoring data gathered from the monitoring tools underlying the XIMM infrastructure. The data could be historical data or real time data. The monitoring data are associated to an endpoint resource, so the user, providing an endpoint identifier, can select the family of data of interests. Defining a filter the user can retrieve only the monitoring data related to the following information.

- Time information: start time and stop time define the start time and stop time of the measure requested. If start time is missing the meaning is: get data from the oldest data available, if stop time is missing the meaning is: get data until the newest data available. Interval is the measures aggregation period.
- Measures information: define the type of measures requested

Operation	HTTP Method	Path
Get a specific measure	GET	{BaseUri}/measurements

Table 20: XIMM-NAM API Measurement operations

9.3.3.1 Get a specific measure

- **Resource:** /measurements
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Data measured filter	XML/JSON

- **XML descriptor body request example:**

```
<filter>
    <endpoints>
        <endpointId>1111</endpointId>
        <time>
            <startTime>2013-09-01 00:00</startTime>
            <stopTime>2013-09-04 00:00</stopTime>
            <interval>60</interval>
        </time>
        <measuresData>
            <measureType>oneWayDelay</measureType>
        </measuresData>
    </endpoints>
</filter>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Measurement values	XML/JSON
500	Internal Server Error	none	none
503	Endpoint not present	none	none
504	Data mismatch	none	none

- **XML OK Response body example:**

```
<measurements>
    <endpoint>
        <endpointId>endpoints.1111</endpointId>
        <measuresData>
```

```
            <measureType>oneWayDelay</measureType>
            <values>
                <sample>
                    <startInterval>2013-09-01 00:00</startInterval>
                    <value>1034</value>
                </sample>
                <sample>
                    <startInterval>2013-09-01 01:00</startInterval>
                    <value>970</value>
                </sample>
                <sample>
                    <startInterval>2013-09-01 02:00</startInterval>
                    <value>932</value>
                </sample>
                ...
            </values>
        <measuresData>
    </endpoint>
</measurements>
```

9.4 XIMM-NPM API Specification

The XIMM-NPM module exposes an internal API towards the XIMM Data & Policy Handler. The same API is exposed externally by XIMM Data & Policy Handler to provide standard access to clients who need monitoring functionalities from XIFI infrastructure.

The set of XIMM-NPM API is related to Network Passive Monitoring functionalities only, both historical and on-demand measurements. The NPM Module behind the NPM API Handler manages the different sources of measurement data and the distribution of modules and data in a transparent manner, hence the API client does not need information of the way the data are collected and stored and how the XIMM platform is distributed.

9.4.1 API Operations

The XIMM-NPM API supports a number of operations which include:

- **Endpoint Operations** - The Endpoint resources represents an endpoint for which measurement parameters can be produced.
- **Measurements Operations** - The Measurements resources allow the user to consume measurement parameters.

9.4.2 Endpoint Operations

Endpoint is the network resource where the monitoring functionalities are performed and from where the data measurements are collected. Thus each measurement performed by a monitoring tool has the corresponding endpoint data in the XIMM-NPM API. The endpoint contains the details description of the measure requested, as described in the following.

- Endpoint data: Network resource information from where the measurements are performed and collected. It can be a physical port or a logical port.
- Time information: start time and stop time define the start time and stop time of the measure requested. If start time is missing the meaning is “starts from now”; if stop time is missing the

meaning is “never stop”. Interval is the measures aggregation period.

- Measures information: lists the measures performed on the specific endpoint. It contains the measures characteristics, if any.

Operation	HTTP Method	Path
List all endpoints	GET	{BaseUri}/endpoints
Get a specific endpoint	GET	{BaseUri}/endpoints/endpoint_id

Table 21: XIMM-NPM API Endpoint operations

9.4.2.1 List all endpoints

- **Resource:** /endpoints
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Endpoint filtering descriptor	XML/JSON

- **XML "filtering descriptor" body request example:**

```
<filter>
    <endpointData>
        <domain>domIta</domain>
    </endpointData>
    <measuresData>
        <measureType>linkUtilization</measureType>
        <measureType>bytesTransmitted</measureType>
    </measuresData>
</filter>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of endpoint identifiers	XML/JSON
500	Internal Server Error	none	none
502	Data mismatch	none	none

- **XML OK Response body example:**

```
<endpointIds>
        <endpointId>1111</endpointId>
        <endpointId>1122</endpointId>
        <endpointId>444</endpointId>
</endpointIds>
```

9.4.2.2 Get a specific endpoint

- **Resource:** /endpoints/{endpoint_Id}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Endpoint details	XML/JSON
500	Internal Server Error	none	none
503	Endpoint not present	none	none

- **XML OK Response body example:**

```
<endpoints>
    <endpoint>
        <endpointId>1111</endpointId>
        <endpointData>
            <domain>domIta</domain>
            <device>switchA2</device>
            <interface>5.2.0</interface>
            <logicalInterface>300</logicalInterface>
        </endpointData>
        <time>
            <startTime>2013-08-25 18:00</startTime>
            <stopTime></stopTime>
            <interval>15</interval>
            </time>
        <measuresData>
            <measureType>linkUtilization</measurType>
            <measureType>bytesTransmitted</measureType>
        </measuresData>
    </endpoint>
</endpoints>
```

9.4.3 Measurement operations

Measurement is the resource that allows the user to get the monitoring data gathered from the monitoring tools underlying the XIMM infrastructure. The data could be historical data or real time data. The monitoring data are associated to an endpoint resource, so the user, providing an endpoint identifier, can select the family of data of interests. Defining a filter the user can retrieve only the monitoring data related to the following information.

- Time information: start time and stop time define the start time and stop time of the measure requested. If start time is missing the meaning is: get data from the oldest data available, if stop time is missing the meaning is: get data until the newest data available. Interval is the measures aggregation period.
- Measures information: define the type of measures requested

Operation	HTTP Method	Path
Get a specific measure	GET	{BaseUri}/measurements

Table 22: XIMM-NPM API Measurement operations

9.4.3.1 Get a specific measure

- **Resource:** /measurements
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Data measured filter	XML/JSON

- **XML descriptor body request example:**

```
<filter>
    <endpoint>
        <endpointId>1111</endpointId>
        <time>
            <startTime>2013-09-01 00:00</startTime>
            <stopTime>2013-09-04 00:00</stopTime>
            <interval>60</interval>
        </time>
        <measuresData>
            <measureType>linkUtilization</measureType>
        </measuresData>
    </endpoint>
</filter>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Measurement values	XML/JSON
500	Internal Server Error	none	none
503	Endpoint not present	none	none
504	Data mismatch	none	none

- **XML OK Response body example:**

```
<measurements>
    <endpoint>
        <endpointId>1111</endpointId>
        <measuresData>
            <measureType>linkUtilization</measureType>
            <values>
                <sample>
                    <startInterval>2013-09-01 00:00</startInterval>
                    <value>45</value>
                </sample>
                <sample>
                    <startInterval>2013-09-01 01:00</startInterval>
                    <value>48</value>
                </sample>
                <sample>
                    <startInterval>2013-09-01 02:00</startInterval>
                    <value>48</value>
                </sample>
                ...
            </values>
        <measuresData>
    </endpoint>
</measurements>
```

9.5 XIMM-DEM API Specification

The XIMM-DEM module exposes an internal API towards the XIMM Data & Policy Handler. The same API is exposed externally by XIMM Data & Policy Handler to provide standard access to clients who need monitoring functionalities from XIFI infrastructure.

The set of XIMM-DEM API is related to Datacenter and Enablers Monitoring functionalities only, both historical and on-demand measurements. The DEM Module behind the DEM API Handler manages the different sources of measurement data and the distribution of modules and data in a transparent way, therefore the API client does not need information of the way the data are collected and stored and how the XIMM platform is distributed.

9.5.1 API Operations

The XIMM-DEM API supports a number of operations which include:

- **Endpoint Operations** - The Endpoint resources represents an endpoint for which measurement parameters can be produced.
- **Measurements Operations** - The Measurements resources allow the user to consume measurement parameters.

9.5.2 Endpoint Operations

Endpoint is the network resource where the monitoring functionalities are performed and from where the data measurements are collected. Thus each measurement performed by a monitoring tool has the corresponding endpoint data in the XIMM-DEM API. The endpoint contains the details description of the measure requested, as described in the following.

- Endpoint data: Network resource information from where the measurements are performed and collected. It can be a physical port or a logical port.
- Time information: start time and stop time define the start time and stop time of the measure requested. If start time is missing the meaning is “starts from now”; if stop time is missing the meaning is “never stop”. Interval is the measures aggregation period.
- Measures information: lists the measures performed on the specific endpoint. It contains the measures characteristics, if any.

Operation	HTTP Method	Path
Create a new endpoint	POST	{BaseUri}/endpoints
List all endpoints	GET	{BaseUri}/endpoints
Get a specific endpoint	GET	{BaseUri}/endpoints/endpoint_id
Delete a specific endpoint	DELETE	{BaseUri}/endpoints/endpoint_id

Table 23: XIMM-DEM API Endpoint operations

9.5.2.1 Create a new endpoint

- **Resource:** /endpoints
- **Method:** POST
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Endpoint creation's details descriptor	XML/JSON

- **XML descriptor body request example:**

```
<descriptor>
    <endpointData>
        <domain>domIta</domain>
        <device>vm</device>
        <interface>5.2.0</interface>
        <logicalInterface>300</logicalInterface>
    </endpointData>
    <time>
        <startTime>2013-08-25 17:00</startTime>
        <stopTime></stopTime>
        <interval>15</interval>
    </time>
```

```
    <measuresData>
        <measure>
            <measureType>diskused</measureType>
            <pingInterval>10>/pingInterval>
        </measure>
        <measure>
            <measureType>cpuload</measureType>
        </measure>
    </measuresData>
</descriptor>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Endpoint identifier	XML/JSON
500	Internal Server Error	none	none
501	Resource unreachable	none	none
502	Data mismatch	none	none

9.5.2.2 List all endpoints

- **Resource:** /endpoints
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Endpoint filtering descriptor	XML/JSON

- **XML "filtering descriptor" body request example:**

```
<filter>
    <endpointData>
        <domain>domIta</domain>
    </endpointData>
    <measuresData>
        <measure>
            <measureType>diskused</measureType>
        </measure>
        <measure>
            <measureType>cpuload</measureType>
        </measure>
    </measuresData>
</filter>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of endpoint identifiers	XML/JSON
500	Internal Server Error	none	none
502	Data mismatch	none	none

- **XML OK Response body example:**

```
<endpointIds>
        <endpointId>1111</endpointId>
        <endpointId>1122</endpointId>
        <endpointId>444</endpointId>
</endpointIds>
```

9.5.2.3 Get a specific endpoint

- **Resource:** /endpoints/{endpoint_Id}
- **Method:** GET
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Endpoint details	XML/JSON
500	Internal Server Error	none	none
503	Endpoint not present	none	none

- **XML OK Response body example:**

```
<endpoints>
    <endpoint>
        <endpointId>1111</endpointId>
        <endpointData>
            <domain>domIta</domain>
            <device>vm</device>
            <interface>5.2.0</interface>
        </endpointData>
        <time>
            <startTime>2013-08-25 17:00</startTime>
            <stopTime></stopTime>
            <interval>15</interval>
            </time>
        <measuresData>
            <measure>
                <measureType>diskused</measurement>
                <Interval>10</Interval>
            </measure>
```

```
            <measure>
                <measureType>cpuload</measureType>
            </measure>
        <measuresData>
    </endpoint>
</endpoints>
```

9.5.2.4 Delete a specific endpoint

- **Resource:** /endpoints/{endpoint_id}
- **Method:** DELETE
- **Query string parameters:** none
- **Request body:** none
- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	none	none
500	Internal Server Error	none	none
504	Endpoint not present	none	none
505	Endpoint not deleted	none	none

9.5.3 Measurement operations

Measurement is the resource that allows the user to get the monitoring data gathered from the monitoring tools underlying the XIMM infrastructure. The data could be historical data or real time data. The monitoring data are associated to an endpoint resource, so the user, providing an endpoint identifier, can select the family of data of interests. Defining a filter the user can retrieve only the monitoring data related to the following information.

- Time information: start time and stop time define the start time and stop time of the measure requested. If start time is missing the meaning is: get data from the oldest data available, if stop time is missing the meaning is: get data until the newest data available. Interval is the measures aggregation period.
- Measures information: define the type of measures requested

Operation	HTTP Method	Path
Get a specific measure	GET	{BaseUri}/measurements

Table 24: XIMM-DEM API Measurement operations

9.5.3.1 Get a specific measure

- **Resource:** /measurements
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Data measured filter	XML/JSON

- **XML descriptor body request example:**

```
<filter>
    <endpoints>
        <endpointId>1111</endpointId>
        <time>
            <startTime>2013-09-01 00:00</startTime>
            <stopTime>2013-09-04 23:59</stopTime>
            <interval>60</interval>
        </time>
        <measuresData>
            <measureType>diskused</measureType>
        </measuresData>
    </endpoints>
</filter>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Measurement values	XML/JSON
500	Internal Server Error	none	none
503	Endpoint not present	none	none
504	Data mismatch	none	none

- **XML OK Response body example:**

```
<measurements>
    <endpoint>
        <endpointId>endpoints.1111</endpointId>
        <measuresData>
            <measureType>diskused</measureType>
            <values>
                <sample>
                    <startInterval>2013-09-01 00:00</startInterval>
                    <value>55</value>
                </sample>
                <sample>
                    <startInterval>2013-09-01 01:00</startInterval>
                    <value>54</value>
```

```
                    </sample>
                    <sample>
                        <startInterval>2013-09-01 02:00</startInterval>
                        <value>55</value>
                    </sample>
                    ...
                </values>
            <measuresData>
        </endpoint>
</measurements>
```

9.6 GE Deployment and Configuration Adapter (GE-DCA) API Specification

The GE DCA module exposes its API towards the XIFI portal and the Federation Manager (SOM GE). The methods included in the GE-DCA API are related to the configuration and deployment of the GEs onto the XIFI nodes. As already discussed the GE-DCA is homogenizing the configuration and deployment mechanisms (i.e. mainly as images and through recipes). While it does not provide new functionality in terms of deployment and configuration (the related GEs have already been developed within FI-WARE project), it leverages on the federation aspects of XIFI and contributes towards the checking of resource availability, the simplification of the deployment and configuration, and the provision of related information.

As stated earlier, GE-DCA deals with the deployment and configuration of all GEs, apart from the ones included in the FI-WARE Cloud chapter, whose deployment and configuration is treated in other XIFI WPs.

9.6.1 API Operations

The GE DCA API supports a number of operations which include:

- **Endpoint Operations** - The Endpoint resources represents an endpoint for which measurement parameters can be produced.
- **Measurements Operations** - The Measurements resources allows the user to consume measurement parameters.

9.6.2 Deployment and Configuration Operations

Prior to the deployment and configuration of GE, the DCA has to perform a check on the availability of the necessary resources. These resources are both hardware and software and they are depending on the specific GE (as described in the administration and installation guide). GE requirements are known beforehand to the DCA. The mechanism for the update of this type of knowledge (on behalf of the DCA) is currently manual (since to our knowledge FIWARE catalogue does not provide automatic updating capabilities). If the availability of the resources is verified, the DCA can perform the deployment and configuration of the GE. This is offered in a simplified way, requiring a minimal set of arguments:

- GE: The type of GE that is going to be deployed.
- Node: The host where the deployment will take place.
- Attributes: Parameters that allow for a level of customization, including versioning of the GE and of the products composing the necessary environment, initialization data, logging

capabilities, access and usage information as well as the timing of the deployment.

Operation	HTTP Method	Path
Check resource availability	GET	{BaseUri}/deployment
Deploy GE as image	POST	{BaseUri}/deployment
Deploy GE through recipe	POST	{BaseUri}/deployment

Table 25: GE-DCA API Deployment and Configuration operations

9.6.2.1 Check resource availability

- **Resource:** /deployment
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
GE and node details descriptor	XML/JSON

- **XML descriptor body request example:**

```
<descriptor>
    <nodeDescriptor>
        <fqn>XIFI.infrastructure1.node1</fqn>
        <hostname>olympos13</hostname>
        <ip>83.235.159.221</ip>
    </nodeDescriptor>
    <geDescriptor>
        <type>CEP</type>
    </geDescriptor>
</descriptor>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	verification response	none
500	Internal Server Error	none	none
501	Node unreachable	none	none
502	GE mismatch	none	none

- **XML OK Response body example:**

```
<resourceAvailabilityCheck>
        <result>0</result>
        <endpointId>1122</endpointId>
        <justification>444</justification>
</resourceAvailabilityCheck>
```

9.6.2.2 Deploy GE through Image

- **Resource:** /deployment
- **Method:** POST
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Node descriptor	XML/JSON
GE descriptor	XML/JSON
DC Attributes	XML/JSON

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Measurement values	XML/JSON
500	Internal Server Error	none	none
503	Node unreachable	none	none
504	GE image not available	none	none

9.6.2.3 Deploy GE as Recipe

- **Resource:** / deployment
- **Method:** POST
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
Node descriptor	XML/JSON
GE descriptor	XML/JSON
DC Attributes	XML/JSON

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	Measurement values	XML/JSON
500	Internal Server Error	none	none
503	Node unreachable	none	none
504	GE recipe not available	none	none

9.6.3 Deployment and Configuration Information Operations

At any time the DCA can provide information on the specific nodes within XIFI federation that the GEs are deployed and configured and the types of the GEs that are deployed on a specific node. The set of arguments include the following:

- GE: The type of GE of interest.
- Node: The host of interest.

Operation	HTTP Method	Path
Get list of GEs per node	GET	{BaseUri}/deployment
Get list of nodes with a GE	GET	{BaseUri}/deployment

Table 26: GE-DCA API Deployment and Configuration Information operations

9.6.3.1 Get list of GEs per node

- **Resource:** / deploymentInfo
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
GE descriptor	XML/JSON

- **XML descriptor body request example:**

```
<descriptor>
    <nodeDescriptor>
        <fqn>XIFI.infrastructure1.node.1</fqn>
        <hostname>olympos13</hostname>
        <ip>83.235.159.221</ip>
```

```
    </nodeDescriptor>
</descriptor>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	list of GEs	none
500	Internal Server Error	none	none
501	Nodes unreachable	none	none
502	GE mismatch	none	none

- **XML OK Response body example:**

```
<GEs>
    <geDescriptor>
        <type>CEP</type>
        <dcAttributes>
            <version>1.0.2</version>
        </dcAttributes>
    </geDescriptor>
</GEs>
```

9.6.3.2 Get list of nodes per GE

- **Resource:** / deploymentInfo
- **Method:** GET
- **Query string parameters:** none
- **Request body:**

Body Content	Representation
GE descriptor	XML/JSON

- **XML descriptor body request example:**

```
<descriptor>
    <geDescriptor>
        <type>CEP</type>
        <dcMode>Recipe</dcMode>
    </geDescriptor>
</descriptor>
```

- **Possible responses:**

Code	Description	Body Content	Representation
200	OK	List of nodes	none
500	Internal Server Error	none	none
501	Nodes unreachable	none	none
502	GE mismatch	none	none

- **XML OK Response body example:**

```
<Nodes>
    <node>
        <nodeDescriptor>
            <fqn>XIFI.infrastructure1.node1</fqn>
            <hostname>hercules</hostname>
            <ip>147.102.7.80</ip>
        </nodeDescriptor>
        <numDeployedGEs>1</numDeployedGEs>
        <nodeDescriptor>
            <fqn>XIFI.infrastructure1.node2</fqn>
            <hostname>pasifai</hostname>
            <ip>147.102.7.132</ip>
        </nodeDescriptor>
        <numDeployedGEs>3</numDeployedGEs>
    </node>
</Nodes>
```

10 CONCLUSIONS

This document has described the specification of the infrastructure adaptation components, which are going to be developed in WP3, in the form of APIs open specifications. With the specification described, we provide the APIs and adapters to support the integration of infrastructures into the XIFI framework and federation with a common set of functionalities, interoperability and a high quality of the service offered.

The adaptation components will be developed based on the tools which has been described in the first sections and offered as toolkit for the infrastructures hosting the FI-WARE GEs to offer a high-end service to the developers with the quality, reliability, flexibility and fast response required. These specifications are open, enabling future updates and external support to fine tune them to the different infrastructures.

The achievement of these specifications can be summarized as follows:

- XIFI will use the network adaptation mechanisms to control and access the network resources (involving backbone/access - wired/wireless networks) and assure the control of the required quality of the service in an interconnected environment between the infrastructures and the Internet networks. With the presented network architectural approach it is possible to provide a Network-as-a-Service (NaaS), offering a common interface to the network layer functions and parameters, so services are able to control network properties appropriately according to their needs. This is important to support the QoS and flexibility in the user access.

- XIFI will use the monitoring APIs to provide the adaptation components for the monitoring of the infrastructures at different levels to expose and control the relevant parameters to assure the required level of interoperability with the tools adopted by XIFI federation and ensure shared resource management. The XIMM middleware will allow such federated monitoring in a unified way across infrastructures and interconnecting networks.

- XIFI will use the Generic Enablers deployment and configuration toolbox to facilitate the management (configuration, control, etc.) of the XIFI federation Generic Enablers and its deployment on top of the platforms of the infrastructures willing to offer FI-PPP GEs and their functionalities.

REFERENCES

[1] A PCE-based Architecture for Application-based Network Operations. December 2012. http://tools.ietf.org/html/draft-farrkingel-pce-abno-architecture-00

[2] A Survey of Distributed Enterprise Network and Systems Management Paradigms. Journal of Network and Systems Management. September 1999. http://infoscience.epfl.ch/record/110/files/Martin-FlatinZH99.pdf

[3] Bandwidth Test Controller (BWCTL) Details. http://www.internet2.edu/performance/bwctl/architecture.html

[4] Cloud computing infrastructure requirements. ITU-T Recommendation Y.3510. May 2013. http://www.itu.int/rec/T-REC-Y.3510-201305-P

[5] Cloud Computing Networking: Challenges and Opportunities for Innovations. IEEE Communications Magazine. July 2013. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6553678

[6] Collectd Canonical Repository. https://github.com/octo/Collectd/

[7] Collectd Documentation. http://Collectd.org/documentation.shtml

[8] Collectd. http://Collectd.org/

[9] Distributed Management Task Force. http://dmtf.org/standards/cloud

[10] Distributed Nagios eXecutor. http://library.nagios.com/library/products/nagiosxi/documentation/308-using-dnx-with-nagios

[11] FI-WARE Catalogue. http://catalogue.fi-ware.eu/

[12] FI-WARE Cloud Hosting GEs Catalogue. http://catalogue.fi-ware.eu/enablers?chapter_tid=2

[13] FI-WARE Context Broker GE. http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Data.PubSub

[14] FI-WARE DCRM Open Specification. https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.DCRM

[15] FI-WARE Identity Management Open Specification. https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Identity_Management_Generic_Enabler

[16] FI-WARE Monitoring Open Specification. https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.Monitoring

[17] FI-WARE NGSI-10 Open RESTful API Specification. http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI-10_Open_RESTful_API_Specification

[18] FI-WARE NGSI-9 Open RESTful API Specification. http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI-9_Open_RESTful_API_Specification

[19] FI-WARE PaaS Open Specification. http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.PaaS

[20] FI-WARE SDC Open Specification. http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.SDC

[21] FI-WARE. How to create recipes for the GEs. https://forge.fi-

ware.eu/plugins/mediawiki/wiki/fiware/index.php/How_to_create_recipes_for_the_GEs

[22] FI-WARE. http://www.fi-ware.eu/

[23] Foreman. http://theforeman.org/

[24] Functional requirements and reference architecture. Part 2. http://www.itu.int/en/ITU-T/focusgroups/cloud/Documents/FG-coud-technical-report.zip

[25] Future Internet Public-Private Partnership (FI-PPP). http://www.fi-ppp.eu/

[26] GEANT BoD Product brief. http://www.geant.net/Media_Centre/Media_Library/Media%20Library/BoD-product-brief-18.5.12.pdf

[27] GÉANT cNIS. http://geant3.archive.geant.net/service/cnis/pages/home.aspx

[28] Global vs Per-Domain Monitoring of Multi-Domain Networks. IRISA/ University of Rennes. http://www.irisa.fr/prive/bcousin/Articles/Articles%202011/monitoring.LCN2011.pdf

[29] IETF RFC 4656: A One-way Active Measurement Protocol (OWAMP). Internet 2 Network Working Group. September 2006. http://tools.ietf.org/html/rfc4656

[30] InterDomain Controller Protocol (IDCP). http://www.controlplane.net/

[31] Iperf. http://sourceforge.net/projects/Iperf/

[32] Logical layers. http://en.wikipedia.org/wiki/Telecommunications_Management_Network

[33] Mantychore. Project Presentation. November 2010. http://www.mantychore.eu/wp-content/files/MFP7%20D1%20r4.pdf

[34] Mirantis Fuel for OpenStack. https://fuel.mirantis.com/

[35] Morfeo 4CaaSt Project. http://www.4caast.eu/

[36] Nagios Distributed Monitoring. http://exchange.nagios.org/directory/Addons/Distributed-Monitoring

[37] Nagios Fusion. http://www.nagios.com/products/nagiosfusion

[38] Nagios Plugin Directory. http://exchange.nagios.org/directory/Plugins

[39] Nagios. http://www.nagios.org/

[40] NGSI Context Management. Open Mobile Alliance (OMA). August 2010. http://technical.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf

[41] NIST Cloud Computing Reference Architecture. September 2011. http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf

[42] One-Way Ping (OWAMP) Details. http://www.internet2.edu/performance/owamp/details.html

[43] Open Cloud Computing Interface. http://occi-wg.org/

[44] Open Datacenter Alliance Usage Model: Software-Defined Networking (rev. 1.0). Open Datacenter Alliance. 2013. http://www.opendatacenteralliance.org/docs/Software_Defined_Networking_Master_Usage_Model_Rev1.0.pdf

[45] Open Grid Forum. http://www.ogf.org/

[46] Open Networking Foundation. https://www.opennetworking.org/

[47] OpenFlow. http://archive.OpenFlow.org/

[48] OpenNaaS. http://www.OpenNaaS.org/

[49] OpenNMS Monitors. http://www.opennms.org/wiki/Monitors

[50] OpenNMS Project. http://www.opennms.org/

[51] OpenStack. http://www.openstack.org/

[52] OpenStack Networking API 2.0. http://docs.openstack.org/api/openstack-network/2.0/content/ch_preface.html

[53] PerfSONAR MDM Documentation. https://forge.geant.net/forge/display/PerfSONAR/Home

[54] PerfSONAR MSM – PerSONAR PS Comparision. GÉANT. March 2012. https://indico.in2p3.fr/getFile.py/access?contribId=21&sessionId=2&resId=0&materialId=paper&confId=6900

[55] PerfSONAR UI User Guide. http://downloads.PerfSONAR.eu/repositories/documents/PerfSONARUI_user_guide_1.1.pdf

[56] PerfSONAR. http://www.PerfSONAR.net/

[57] PerfSONAR-PS Toolkit. http://psps.PerfSONAR.net/toolkit/

[58] PerfSONAR-PS. http://psps.PerfSONAR.net/

[59] Puppet. http://puppetlabs.com/puppet/what-is-puppet

[60] Rackspace Private Cloud. http://www.rackspace.com/knowledge_center/product-page/rackspace-private-cloud

[61] SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains. June 2012. http://tools.ietf.org/html/draft-yin-sdn-sdni-00

[62] Software-Defined Networking: The New Norm for Networks. Open Networking Foundation. April 2012. https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[63] The NIST Definition of Cloud Computing. National Institute of Science and Technology. Retrieved 22 May 2013.http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[64] Thrulay. http://e2epi.internet2.edu/thrulay/

[65] Tuskar. http://github.com/tuskar/tuskar

[66] XIFI Deliverable D1.1. http://wiki.fi-XIFI.eu/Public:D1.1

[67] XIFI Wiki. http://wiki.fi-XIFI.eu

APPENDIX A: GENERIC ENABLERS SW AND HW REQUIREMENTS

Category	Name	SW Reqs	HW Reqs
Applications/ Services Ecosystems and Delivery Framework	Application Mashup - Wirecloud	OS: Any DB: Any (MySQL, PostgreSQL, Sqlite3, etc.) Web Server: Any Apache based (Apache, Nginx, Gunicorn, etc) Language: python<3.0 (Django)	CPU: Not CPU intensive (less than 5% per user) RAM: Less than 256MB per user HDD: > 10GB I/O Intensivity: (more than 12 http requests per sec.)
	Light Semantic Composition Editor	Not mentioned	Not mentioned
	Marketplace - SAP RI	Database Manager - MySQL (5.5) Server MySQL Client Java 1.6.x Application Server Apache Tomcat 6.x OS support Ubuntu 11.04 and 11.11, 12.04 LTS or Microsoft Windows 7	CPU: 1-2 cores with at least 2.4 GHZ Physical RAM: 2G-4GB Disk Space: 10GB
	Mediator Mediator_TI	Ubuntu 10.04, 12.04 LTS Java 1.6.x - mandatory	CPU 1-2 cores with at least 2.4 GHZ Physical RAM 2GB Disk Space 30GB
	Repository - SAP RI	OS: Ubuntu 11.04, 11.11, 12.04 LTS OS: Windows 7 DB: MongoDB 2.x Lang: Java 1.6.x App. Server: Apache Tomcat 6.x Repository Software	CPU: 1-2 cores (>2.4GHz) RAM: 1-2GB HDD: 25GB
	Service Composition - Ericsson Composition Engine (ECE)	Not mentioned	Not mentioned

	Service Mashup - Mashup Factory	Not mentioned	Not mentioned
Cloud Hosting	IaaS Data Center Resource Management	Ubuntu 12.04 x86_64 MySql server tgt (linux SCSI target user-space tool) Open-iScsi (iSCSI implementation) RabbitMQ server KVM, libvirt Apache server LVM (Logical volume manager) ntp (Network time protocol)	5 physical nodes (servers): host1: Cloud Controller node, Pivot, DOVE VM (Quantum based) host2: Keystone, Dashboard, Swift proxy and 3 * Swift storage nodes VMs host 3, 4 & 5: Compute node, zk service CPU: 8 cores (>=2.4 GHz, VT-x enabled) RAM: 16GB Disk Space: 1TB
	IaaS Service Management - Claudia	Not mentioned	Not mentioned. It is tested with a Virtual Data Center with characteristics: Management 1Gbps, Service 1Gbps, Storage 50.000GB, Comput capacity 10.000 units, memory 10.000GB
	Object Storage	OpenStack with Swift CDMI interface	Swift and the CDMI interface resource constraints (e.g., 100MB memory, 10MB disk space). Sufficient storage should be available to accommodate all objects (likely to be TBs of data. A Swift installation is recommended to have at least 5 storage nodes for redundancy purposes
Data/Context Management	Complex Event Processing (CEP)	OS: Non-specific DB: No database is used Web Server: Tomcat Language: J2SE 6 or later (Java)	Not specified
	Compressed Domain Video Analysis	The RESTful web server and the core library of Codoan is provided as single executable file (Windows: codoan.exe, Linux: codoan) OS: Windows XP or Linux	Minimum: CPU: 4 cores RAM: 2GB HDD: - Recommended: CPU: 4 cores RAM: 4GB HDD: -

		DB: - Web Server: mongoose Language: C++	
	Location Platform - LOCS	The Location Platform can be accessed via the OMA RESTFul Terminal Location in order to: - Obtain the current terminal location via A-GPS, WiFi and Cell-Id location methods (R1) - Manage client-specific subscriptions to area (circle) notifications and activate intelligently various location methods depending on the end-user environment (R2) The location platform services can be directly accessed via a web browser, where the parameters of the location request are indicated directly in the URL	Not mentioned
	Media-enhanced Query Broker	Java JRE 6 The implementation at its core is based on the Spring Framework (e.g., enabling extensibility and inversion of control) and MAVEN (e.g., quality assurance and build management).	Not mentioned
	Publish/Subscribe Context Broker - Context Awareness Platform	CentOS 32-bit Java SE Development Kit (JDK) v1.6.0 JBOSS AppServer v5.1.0GA MySQL Server v5.0 DBMS	Server with 2 CPUs RAM: 2GB HDD: minimum 50 GB free hard disk space on used partition
	Publish/Subscribe Context Broker - SAMSON Broker	CentOS v6.3 EPEL MySQL database	CPU: 2 cores RAM: 2GB HDD: 50GB
	Semantic Annotation	Not mentioned	Not mentioned
	Semantic Application Support	Not mentioned	Not mentioned
Interface to Networks and Devices	Cloud Edge	OS: Ubuntu Linux 12.04LTS DB: mysql Web Server: apache2/lighttpd Language: Ruby	Non specific (x86 machine). It has been tested on a x86 PC (should be of minimal requirements: ATOM-Class processor, 1 or 2GB of RAM and a small low cost

			HD))
IoT Services Enablement	Gateway Data Handling (Esper4FastData)	Servlet Version: OS: Windows DB: Embedded in Esper4FastData Web Server: Tomcat 6 Language: Java Mobile Version: Not relevant (android) OSGi : OS: Linux DB: Web Sever: Language: Java	Servlet Version: Minimum: CPU: >1x1GHz RAM: 512MB HDD: 100MB OSGi Version: Minimum: CPU: 1x1GHz RAM: 512MB HDD: 100MB
	Gateway Data Handling (SOL/CEP)	NO ACCESS	NO ACCESS
	Gateway Device Management	NO ACCESS	NO ACCESS
	Protocol Adapter-ZPA	JavaSE 1.6 OSGI framework GAL (Gateway Abstraction Layer) & acquire and install an USB dongle	access restricted to members of the PPP Programme
	Things Management	Not mentioned	Not mentioned
Security	Data Handling -PPL	OS: Not specified DB: MySQL EasyPHP 5.3.5 Web Server: Apache Language: Java, PHP	Minimum: CPU: 1 core RAM: 512MB HDD: 500MB Recommended: CPU: > 2 cores RAM: 4GB HDD: 5GB
	DB Anonymizer	OS: Not specified DB: MySQL (tested on v5.5) Web Server: Tomcat (tested on 6.X and 7.X.) Language: Java	Not Speficied. Tested against: CPU: 8 cores RAM: 16GB HDD: 500GB
	Identity Management GCP	The communication between customer, GCP and service is curried out over the OpenID (v2.0) protocol	Not mentioned
	Identity Management One-IDM	SAML 2.0	Not mentioned
	Security Monitoring	1. MulVAL OS: CentOS v5 RedHat Enterprise Linux 5, Virtual Server Administration Apache	1. MulVAL: RAM: 1GB HDD: 20GB 2. SLS: Not specified

		2. SLS: OSSIM (Atos) MySQL Python Apache OpenVAS Osiris Nagios	

Table 27: GEs Software and Hardware Requirements