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

eXperimental Infrastructures for the Future Internet

D3.5: Infrastructures monitoring and interoperability adaptation components API v2

Revision: v.1.1

Work package	WP3
Task	Task T3.2
Due date	30/09/2013
Submission date	03/10/2014
Deliverable lead	Universidad Politecnica de Madrid (UPM)
Authors	Jose Gonzalez (UPM), Federico Alvarez (UPM), Fernando Garcia (UPM), Pablo Rodriguez (TID), Fernando Lopez (TID), Panos Trakadas (SYN), Panos Karkazis (SYN), Silvio Cretti (CNET), Attilio Broglio (CNET)
Reviewers	Sandor Laki (Wigner), Yahya Al-Hazmi (TUB), Federico Facca (CNET)

Abstract	This deliverable provides a more detailed description of the different XIFI infrastructure-related monitoring adaptation components following the APIs and open specifications from D3.2. It also provides the software references with the manuals required for the installation and usage of these components.
Keywords	Future Internet, Adapter, API, Infrastructure, Monitoring, Network, Datacenter, FIWARE, FI-Lab, Generic Enabler, Management, Federation

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	15.09. 2014	Version ready for internal review	Jose Gonzalez (UPM), Federico Alvarez (UPM) et al.
V1.1	19.09.2014	Final revision	Jose Gonzalez (UPM), Federico Alvarez (UPM) et al.

Disclaimer

This report contains material which is the copyright of certain XIFI Consortium Parties and may only be reproduced or copied with permission in accordance with the XIFI consortium agreement.

All XIFI Consortium Parties have agreed to publication of this report, the content of which is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License[1].

Neither the XIFI Consortium Parties nor the European Union warrant that the information contained in the report is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using the information.

Copyright notice

© 2013 - 2015 XIFI Consortium Parties

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Nature of the Deliverable:	P (Prototype)	
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	

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

EXECUTIVE SUMMARY

This document presents the reference documentation of the infrastructure monitoring adaptation toolkit, updating the latest description released in M9 with deliverable D3.2 [29].

Within the framework of the XIFI architecture (latest version available in [27]), the monitoring adaptation layer is the middleware between the set of industry-driven monitoring tools that produces performance data related to diverse resources of each infrastructure node to be federated, and the Federation Monitoring, the component of the monitoring architecture at higher level. The connection with the Federation Monitoring is described in this material, but the reader shall notice that its assessment is out of the scope of this deliverable. To find a detailed specification, D2.5 is the proper document [28].

Hence, the abstraction layer under consideration, designated as XIFI Infrastructure Monitoring Middleware (XIMM), will be the responsible for collecting heterogeneous performance data from each infrastructure of the federation, standardizing the attributes to a common data format and publishing the data to the Federation Monitoring.

The components within such toolkit are:

- NGSI Adapter, the standardizing actor that links several monitoring adapters with the Federation Monitoring
- Network Active Monitoring (NAM) Adapter, which handles bandwidth and latency-based tests among multiple domains
- Datacenter & Enabler Monitoring (DEM) Adapter, in charge of performing the monitoring of the Virtual Machines and the Generic Enablers deployed within the XIFI federation
- OpenStack Data Collector (ODC), which retrieves status data from an OpenStack installation
- Network Passive Monitoring (NPM) Adapter, which collects performance data from network resources

The components are described in their corresponding sections, offering a description of the needed elements to install them in the XIFI infrastructures: architecture, installation manual, user manual, related components, background and, of course, where to find the software.

The content enclosed in this deliverable establishes the basis for a stable architecture tested and deployed in XIFI infrastructure nodes. Future versions that integrate improvements and new features will take this documentation as a starting point.

TABLE OF CONTENTS

EXECUTIVE SUMMARY **3**
TABLE OF CONTENTS **4**
LIST OF FIGURES **6**
LIST OF TABLES **7**
ABBREVIATIONS **8**
1 INTRODUCTION **9**
1.1 Scope 9
1.2 Preceding documentation 9
1.3 Overview 9
2 COMPONENTS DESCRIPTION **11**
2.1 NGSI Adapter 11
2.1.1 Summary 11
2.1.2 Component Responsible 12
2.1.3 Motivation 12
2.1.4 User Stories Backlog 13
2.1.5 State of the art 13
2.1.6 Architecture 14
2.1.7 Release Plan 15
2.1.8 Test Cases 15
2.1.9 Installation Manual 17
2.1.10 User Manual 19
2.2 Network Active Monitoring-NAM Adapter 20
2.2.1 Summary 20
2.2.2 Component Responsible 22
2.2.3 Motivation 22
2.2.4 User Stories Backlog 23
2.2.5 State of the art 25
2.2.6 Architecture 25
2.2.7 Release Plan 29
2.2.8 Test Cases 29
2.2.9 Installation Manual 30
2.2.10 User Manual 31
2.3 Datacenter & Enabler Monitoring-DEM Adapter 36
2.3.1 Summary 36
2.3.2 Component Responsible 38

2.3.3 Motivation 38
2.3.4 User Stories Backlog 39
2.3.5 State of the art 40
2.3.6 Architecture 41
2.3.7 Release Plan 43
2.3.8 Test Cases 43
2.3.9 Installation Manual 48
2.3.10 User Manual 63
2.4 OpenStack Data Collector-ODC 70
2.4.1 Summary 70
2.4.2 Component Responsible 72
2.4.3 Motivation 72
2.4.4 User Stories Backlog 72
2.4.5 State of the art 73
2.4.6 Architecture 73
2.4.7 Release Plan 73
2.4.8 Test Cases 73
2.4.9 Installation Manual 74
2.4.10 User Manual 74
2.5 Network Passive Monitoring-NPM Adapter 75
2.5.1 Summary 75
2.5.2 Component Responsible 77
3 CONCLUSIONS 78
REFERENCES 79

LIST OF FIGURES

Figure 1: XIMM General Architecture 10
Figure 2: NGSI Adapter within the Architecture of a XIFI Node.....11
Figure 3: NGSI Adapter Architecture 14
Figure 4: NGSI Adapter - Sequence Diagram.....15
Figure 5: NGSI Adapter – Check logs at console 20
Figure 6: NAM Adapter within the Architecture of a XIFI Node.....21
Figure 7: Multi-domain connectivity in the context of FIWARE Lab 23
Figure 8: NAM Adapter General Architecture.....26
Figure 9: NAM Measurement Collector - Internal Architecture.....27
Figure 10: Sequence Diagram – Historical Results.....28
Figure 11: Sequence Diagram - On-demand Test 29
Figure 12: Configuration of NAM Server Parameters 33
Figure 13: Configuration of NAM Host Parameters 34
Figure 14: BDW Scheduled Tests.....35
Figure 15: Default values for NAM Scheduled Tests 35
Figure 16: DEM Adapter within the Architecture of a XIFI Node 37
Figure 17: DEM Active Adapter General Architecture 41
Figure 18: DEM Passive Adapter General Architecture 42
Figure 19: DEM Adapter - Sequence Diagram.....42
Figure 20: OpenStack Data Collector within the Architecture of a XIFI Node 71
Figure 21: OpenStack Data Collector Architecture 73
Figure 22: NPM Adapter within the Architecture of a generic XIFI Node.....75

LIST OF TABLES

Table 1: NGSI Adapter Context Details 12
Table 2: NGSI Adapter Dependencies Summary....12
Table 3: NGSI Adapter Reference Details 12
Table 4: NGSI Adapter User Stories Backlog....13
Table 5: NGSI Adapter Release Plan 15
Table 6: NGSI Adapter Test cases 17
Table 7: NAM Adapter Context Details 21
Table 8: NAM Adapter Dependencies Summary....22
Table 9: NAM Adapter Reference Details 22
Table 10: NAM Adapter User Stories Backlog....24
Table 11: NAM Adapter Release Plan 29
Table 12: DEM Adapter Context Details 37
Table 13: DEM Adapter Dependencies Summary 38
Table 14: DEM Adapter Reference Details 38
Table 15: DEM Adapter User Stories Backlog 40
Table 16: DEM Adapter Release Plan 43
Table 17: OpenStack Data Collector Context Details....71
Table 18: OpenStack Data Collector Dependencies Summary 72
Table 19: OpenStack Data Collector Reference Details 72
Table 20: OpenStack Data Collector User Stories Backlog 72
Table 21: OpenStack Data Collector Release Plan 73
Table 22: NPM Adapter Context Details 76
Table 23: NPM Adapter Dependencies Summary 76
Table 24: NPM Adapter Reference Details....77

ABBREVIATIONS

API	Application Programming Interface
CB	Context Broker
DEM	Datacenter and Enablers Monitoring
GE	Generic Enabler
GEi	Generic Enabler instance
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IdM	Identity Management
JSON	JavaScript Object Notation
NAM	Network Active Monitoring
NGSI	Next Generation Service Interfaces
NPM	Network Passive Monitoring
NRPE	Nagios Remote Plugin Executor
ODC	OpenStack Data Collector
QoS	Quality of Service
REST	REpresentational State Transfer
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WP	Work Package
XIMM	XIFI Infrastructure Monitoring Middleware
XML	eXtensible Markup Language

© 2013 - 2015 XIFI Consortium Parties

1 INTRODUCTION

1.1 Scope

Deliverable *D3.5 - Infrastructures monitoring and interoperability adaptation components API v2* provides an updated documentation in relation to the latest version of the XIFI Monitoring Architecture. Each single software component of such architecture will be assessed in detail.

1.2 Preceding documentation

Deliverable *D3.2 - Infrastructures monitoring and interoperability adaptation components toolkit and API* [26] released on month 9 provided a first description of those software components in charge of adapting the monitoring functionalities that need to be provided in the XIFI federated environment. This first documentation helped the consortium to settle a stable framework, where an initial architecture served as basis to implement the corresponding software components that have been deployed and tested throughout this period.

Though the M18 architecture maintains the main outlines defined, due to meaningful changes concerning particular and general issues, some important updates have been required and will be presented in this document as an upgraded documentation. Nonetheless, thanks to the experience gathered, it is expected that this renewed architecture remains in time as stable, leaving future updates as improvements and additional features.

1.3 Overview

As stated above, the XIFI monitoring architecture keeps its foundations by means of the abstraction layer named **XIFI Infrastructure Monitoring Middleware (XIMM)**, which is composed of the following main components (Figure 1):

- NGSI Adapter
- Network Active Monitoring-NAM Adapter
- Datacenter & Enabler Monitoring-DEM Adapter
- OpenStack Data Collector
- Network Passive Monitoring-NPM Adapter

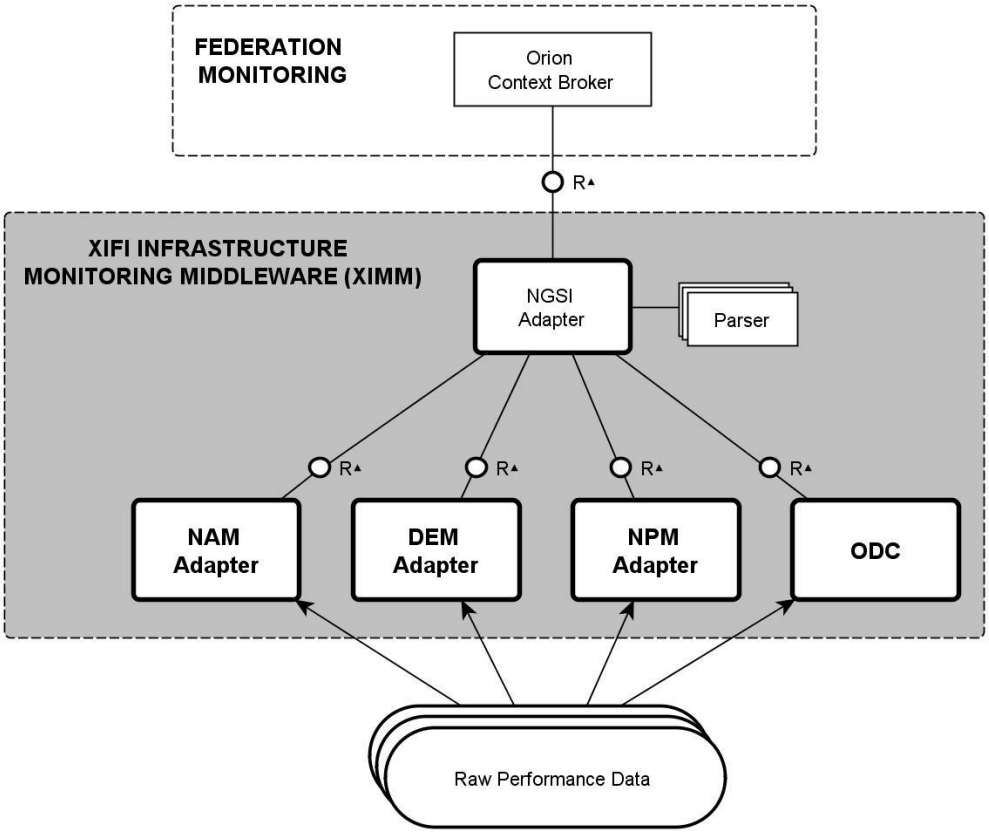

Figure 1: XIMM General Architecture

The XIMM fulfils required adapting tasks to make raw performance data of infrastructure's resources available for the federation layer (by means of the Orion Context Broker [23]). Such Federation Monitoring is out of the scope of this documentation. In order to find further details, its specification can be found in deliverable D2.5 [28].

In general terms, this middleware is in charge of collecting, standardizing and publishing multi-source performance data in an accessible manner to higher layers within the XIFI architecture. Therefore, it represents an important block since without this low-level information no added-value services would be feasible in other work packages, especially WP4.

Following sections will describe each single component, emphasizing in proper details and specifications.

2 COMPONENTS DESCRIPTION

Section 2 addresses the specification of each single component that composes the adaptation layer in the XIFI monitoring architecture. Each subsection will describe the particular architecture of the component, as well as the test cases and installation manuals to deploy them in a given node.

2.1 NGSI Adapter

2.1.1 Summary

Taking into consideration Figure 2, NGSI Adapter is the component responsible for adapting raw data from *Monitoring Adapters & Collectors* into a common, context-based representation (NGSI Entity Context), and updating such information into a *Context Broker* [23] deployed within the node, which belongs to the Federation Monitoring architecture.

NGSI Adapter itself is completely agnostic to the monitoring tools used to produce and gather the raw data, and the exact format of the data originated by their probes. It delegates the concrete adaptation into an extensible set of dynamically loadable parsers, which receive raw data from the probe (sent to NGSI Adapter as part of a HTTP request) and return the corresponding NGSI context. Thus, the adapter is as much independent from other components (e.g. NAM, DEM) as possible, only requiring specific parsers for the probes used by them. Therefore, a single NGSI Adapter is able to handle any kind of data coming from the different probes.

NGSI Adapter asynchronously processes all incoming adaptation requests, so that data collectors, to be introduced in forthcoming subsections, are not blocked. Besides, possible temporary connection errors when issuing update requests to Context Broker are handled through an exponential back off retry policy.

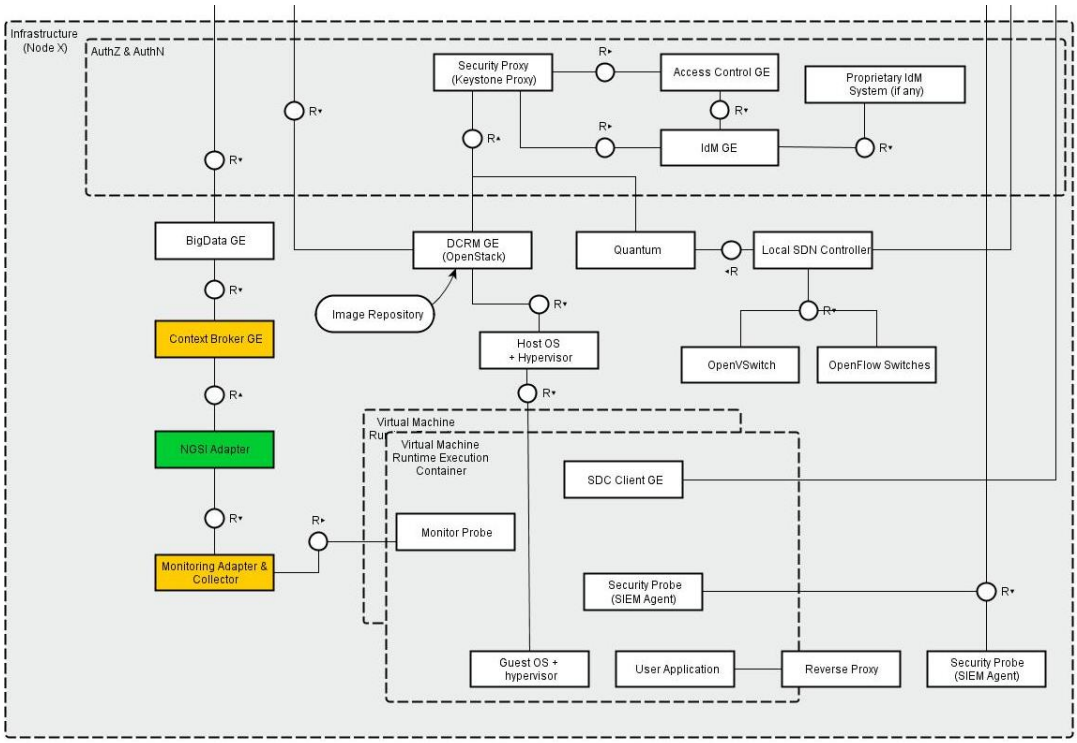

Figure 2: NGSI Adapter within the Architecture of a XIFI Node

Tables below provide specific details with regard to the component:

Reference Scenarios	UC-5 - Network and Data Centre operations
Reference Stakeholders	**Infrastructure owners and operators**: those who need to keep both network and platform under a reliable functioning condition.
Type of ownership	Deployment and Extension
Original tool	Based on FIWARE Context Broker and BigData GEs
Planned OS license	Apache License Version 2.0
Reference OS community	None at the moment

Table 1: NGSI Adapter Context Details

Consist of	• NGSI Adapter core process • Probe-specific parsers (dynamically loaded by core process) • Specific measurement collectors in XIMM modules
Depends on	• Orion Context Broker • NAM, DEM, ODC and NPM Adapters

Table 2: NGSI Adapter Dependencies Summary

2.1.2 Component Responsible

Developer	Contact	Partner
Pablo Rodriguez	pablo.rodriguezarchilla@telefonica.com	TID
Fernando Lopez	fernando.lopezaguilar@telefonica.com	TID

Table 3: NGSI Adapter Reference Details

2.1.3 Motivation

Given the heterogeneity of monitoring tools that may be used within the XIFI federation, and having adopted Orion Context Broker [23] as publish/subscribe mechanism for monitoring data, an adaptation entity is needed in order to transform those data in custom format into the NGSI context vocabulary of Context Broker.

2.1.4 User Stories Backlog

Id	User Story Name	Actors	Description
1	Implement a generic adaptation mechanism to NGSI for monitoring probes	Infrastructure Administrator	As a Monitoring GEi owner, I need to implement a generic adaptation mechanism to convert raw data from monitoring probes into the NGSI format used by Context Broker GE.
2	Analyze perfSONAR-PS module adaptation to NGSI	Infrastructure Administrator	As a Monitoring GEi owner, I want to make an analysis and evaluation of the perfSONAR-PS adapter extension.
3	Analyze openNMS module adaptation to NGSI	Infrastructure Administrator	As a Monitoring GEi owner, I want to make an analysis and evaluation of openNMS adapter extension.
4	Specific Nagios adapters to NGSI	Infrastructure Administrator	As a Monitoring GEi owner, I need to implement the specific components of the Nagios probes for DEM attributes.
5	Analysis of the Nagios and Nagios Event Broker	Infrastructure Administrator	Analysis of the Nagios and Nagios Event Broker (NEB) module in order to develop specific modules to integrate with Nagios.
6	Design of the Nagios Event Broker (NEB) modules associated to NPM and DEM modules	Infrastructure Administrator	Design of the Nagios Event Broker (NEB) modules associated to NPM and DEM modules.
7	Implement Nagios Event Broker (NEB) module for DEM	Infrastructure Administrator	Implement Nagios Event Broker (NEB) module for DEM (hosts / VMs) monitoring to automate the connection between Nagios as general purpose monitoring tool and NGSI Adapter.
8	Implement Nagios Event Broker (NEB) module for NPM	Infrastructure Administrator	Implement Nagios Event Broker (NEB) module for NPM (network elements) snmp-based monitoring to automate the connection between Nagios as general purpose monitoring tool and NGSI Adapter.
9	Add timestamp to the measures coming from Nagios	Infrastructure Administrator	Each measure collected should be associated to a timestamp.

Table 4: NGSI Adapter User Stories Backlog

2.1.5 State of the art

NGSI Adapter is a component originally developed within the FIWARE Monitoring GEi components [6] which aim to aggregate data from different monitoring tools that might have been already installed within the infrastructure owners. This solution shall be capable of providing monitored data through a

standardized format (FIWARE NGSI-10 API [7]) that unifies developments in FI-PPP projects and is compatible with the Context Broker [5] and Big Data [4] GEs.

2.1.6 Architecture

Figure 3 depicts the basic architecture of this component. NGSI Adapter is a standalone server asynchronously processing incoming HTTP requests. The actual adaptation task to process raw input data is delegated to a specific parser module that is dynamically loaded according to the kind of request.

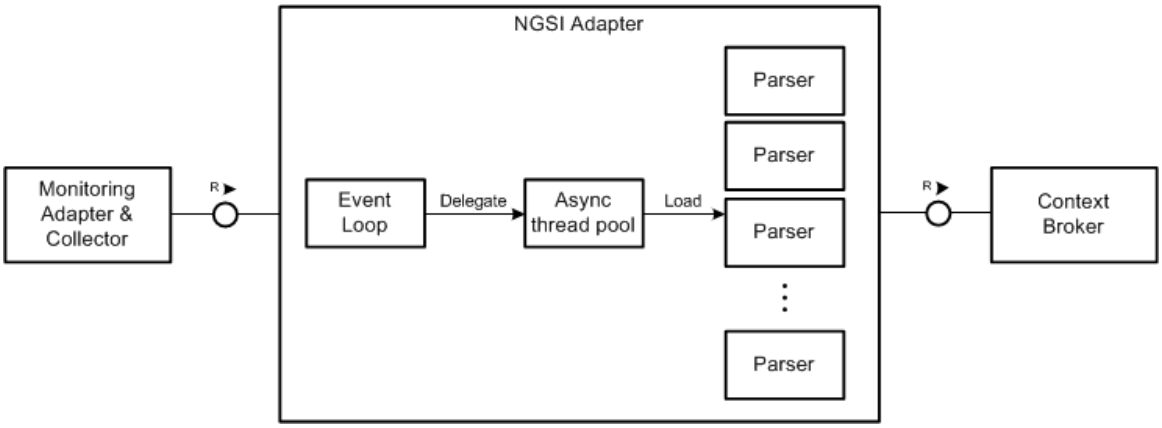

Figure 3: NGSI Adapter Architecture

2.1.6.1 Basic actors

- **Measurement Collectors**

The Measurement Collectors are the core modules within each XIMM Adapter. They will be the responsible actors for collecting the data generated by the different monitoring probes and invoking NGSI Adapter.

- **NGSI Adapter Parsers**

NGSI Adapter knows nothing about the data format from monitoring probes, so its functionality requires being augmented with several parsers given as dynamically loadable software modules for each of the sources. The NGSI Adapter will select the right parser based on the incoming requests, then will load the parser and let it extract monitoring core information to standardize into NGSI format.

2.1.6.2 Main interactions

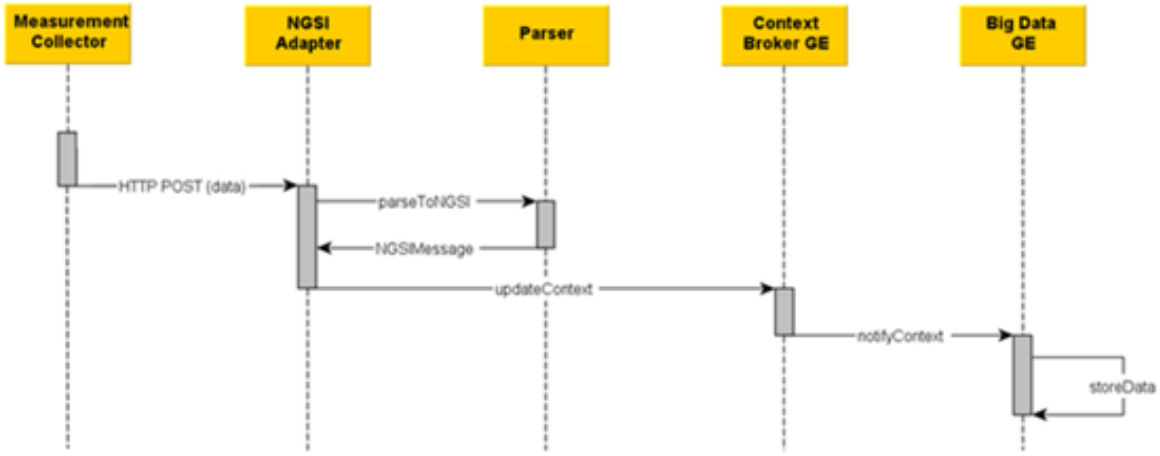

Figure 4: NGSI Adapter - Sequence Diagram

2.1.7 Release Plan

Version ID	Milestone	User Stories
v1.0.0	M9	1,2,3,4,5
v1.0.1	M12	6,7,8
v1.1.0	M18	9

Table 5: NGSI Adapter Release Plan

2.1.8 Test Cases

Table 6 lists each unit test case defined for the NGSI Adapter. In order to not overload this document, a single reference is placed to check the code.

Description	Test script
Test nagios_parser - parse_fails_extra_perf_data_first_line	test/unit/test_nagios_parser.js
Test nagios_parser - parse_fails_extra_perf_data_another_line	test/unit/test_nagios_parser.js
Test nagios_parser - parse_fails_third_perf_data_compound_line	test/unit/test_nagios_parser.js
Test nagios_parser - parse_ok_singleline_text_output_only	test/unit/test_nagios_parser.js
Test nagios_parser - parse_ok_multiline_text_output_only	test/unit/test_nagios_parser.js

Test nagios_parser - parse_ok_singleline_text_output_singleline_perf_data	test/unit/test_nagios_parser.js
Test nagios_parser - parse_fails_singleline_text_output_multiline_perf_data	test/unit/test_nagios_parser.js
Test nagios_parser - parse_ok_multiline_text_output_singleline_perf_data	test/unit/test_nagios_parser.js
Test nagios_parser - parse_ok_multiline_text_output_multiline_perf_data	test/unit/test_nagios_parser.js
Test nagios_parser - get_update_request_fails_missing_entity_id	test/unit/test_nagios_parser.js
Test nagios_parser - get_update_request_fails_missing_entity_type	test/unit/test_nagios_parser.js
Test nagios_parser - get_update_request_fails_empty_request	test/unit/test_nagios_parser.js
Test check_disk - get_update_request_fails_with_invalid_check_disk_content	test/unit/test_check_disk.js
Test check_disk - get_update_request_fails_with_multiple_partitions_check_disk_content	test/unit/test_check_disk.js
Test check_disk - get_update_request_ok_with_valid_check_disk_content	test/unit/test_check_disk.js
Test check_disk - parse_ok_free_space_percentage	test/unit/test_check_disk.js
Test check_load - get_update_request_fails_with_invalid_check_load_content	test/unit/test_check_load.js
Test check_load - get_update_request_ok_with_valid_check_load_content	test/unit/test_check_load.js
Test check_load - parse_ok_cpu_load_percentage	test/unit/test_check_load.js
Test check_mem - get_update_request_fails_with_invalid_check_mem.sh_content	test/unit/test_check_mem.sh.js
Test check_mem - get_update_request_ok_with_valid_check_mem.sh_content	test/unit/test_check_mem.sh.js
Test check_mem - parse_ok_used_mem_percentage	test/unit/test_check_mem.sh.js
Test check_procs - get_update_request_fails_with_invalid_check_procs_content	test/unit/test_check_procs.js
Test check_procs - get_update_request_ok_with_valid_check_procs_content	test/unit/test_check_procs.js
Test check_procs - get_update_request_ok_with_another_threshold_metric	test/unit/test_check_procs.js
Test check_procs - parse_ok_number_of_procs	test/unit/test_check_procs.js

Test check_users - get_update_request_fails_with_invalid_check_users_content	test/unit/test_check_users.js
Test check_users - get_update_request_ok_with_valid_check_users_content	test/unit/test_check_users.js
Test check_users - parse_ok_number_of_users_logged_in	test/unit/test_check_users.js

Table 6: NGSI Adapter Test cases

2.1.9 Installation Manual

2.1.9.1 Requirements

NGSI Adapter should work on a variety of operating systems, particularly on the majority of GNU/Linux distributions (e.g. Debian, Ubuntu, CentOS), as it only requires a V8 JavaScript Engine to run a Node.js server.

Hardware Requirements:

The minimal requirements are:

- RAM: 2 GB

Software Requirements:

NGSI Adapter requires Node.js [16] and its package manager npm [17]. These requirements are automatically checked when installing the `ngsi_adapter` Linux package. For a manual installation, please check Joyent documentation [12]:

- **Node.js**

  Installation using Ubuntu: in order to obtain the most recent version of Node.js or installing on older Ubuntu and other `apt`-based distributions using the following commands:

```
$ curl -sL https://deb.nodesource.com/setup | sudo bash -
$ sudo apt-get install nodejs
```

  Installation using Debian packages (as root):

```
# apt-get install curl
# curl -sL https://deb.nodesource.com/setup | bash -
# apt-get install nodejs nodejs-legacy
```

  Installation using CentOS (requires EPEL: Extra Packages for Enterprise Linux):

```
$ sudo yum install \
  http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-
8.noarch.rpm
$ sudo yum install nodejs npm --enablerepo=epel
```

Installation using other Linux distribution: https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager .

2.1.9.2 Software Repository

- NGSI Adapter: https://github.com/Fiware/fiware-monitoring/tree/master/ngsi_adapter.

2.1.9.3 Setup Guidelines

To install NGSI Adapter in Ubuntu or similar distributions, please perform the following procedure:

1. Log on as root (or user with sudoer privileges) to the machine on which you want to host the NGSI Adapter:

   a) To install a stable version of NGSI Adapter package, please download the `.deb` file from https://forge.fi-ware.org/frs/?group_id=7&release_id=529#cloud-monitoring-3-5-2-title-content:
      - Use `gdebi` or any other tool to install the package.
      - Follow instructions given as part of installation process (for example, to upgrade node version). All dependencies will be automatically resolved by the installer.

   b) To download the latest NGSI Adapter version from GitHub, use following procedure:
      - Download https://github.com/Fiware/fiware-monitoring/archive/master.zip. This includes the `ngsi_adapter` directory corresponding to the NGSI Adapter component.
      - Check that you have installed all the prerequisite software (see the prerequisite software in section 2.1.9.1).
      - Finally, install the `npm` dependencies:

```
$ cd ngsi_adapter/src
$ npm install
```

2. If needed, install additional parsers (in case those provided by default together with the core NGSI Adapter aren't suitable for the probes that will generate monitoring data):

```
$ cp <someparser>.js ngsi_adapter/src/lib/parsers/
```

3. Check configuration and logging values prior running NGSI Adapter:
   - `ngsi_adapter/config/options.js` defines general parameters.
   - `ngsi_adapter/config/logger.js` configures logger (level, rolling, etc).

   Command line options override default values defined at `options.js` file. To get some help about those options, please run:

```
$ adapter --help
```

When installed as a package, a Linux service `ngsi_adapter` is created (but not started). Configuration options to run the service (overriding default values) are set in the variable `DAEMON_ARGS` defined in `/etc/init.d/ngsi_adapter` script.

2.1.10 User Manual

2.1.10.1 API Specification

NGSI Adapter API consists of a single HTTP POST asynchronous operation that is described at https://github.com/Fiware/fiware-monitoring/tree/master/ngsi_adapter#requests.

2.1.10.2 Handbook

NGSI Adapter has neither a visual interface nor returns data, as it acts as an asynchronous message-driven middleware.

The usual way to run NGSI Adapter is by starting the Linux service created when the package is installed:

```
$ sudo service ngsi_adapter start
```

Alternatively, in order to use it and get familiar with its behaviour, it could be run manually on a text console and use any REST client (either browser-based or command line tools like cURL [3] or Wget [10]):

- Run with default configuration values:

```
$ ./adapter
```

- Run specifying some configuration values:

```
$ adapter --listenPort {localport} --brokerUrl
    http://{host}:{port}/
```

In another console, to issue a request (e.g. simulating monitoring data from Nagios's `check_load` probe):

```
$ curl http://localhost:1337/check_load -s -S -d @- <<'...load data'
```

Then check logs at console. For example, URL is a missing required parameter:

```
$ ./adapter
info: Server running at http://127.0.0.1:1337/
info: << HTTP POST
info: >> 200 OK
info: << Body: ...load data
error: updateContext(): "Missing entityId and/or entityType"
```

Figure 5: NGSI Adapter – Check logs at console

A full, valid test would first require to register a new entity 'myEntity' of type 'myType' into an existing Context Broker (see [23]), and then issue a valid `check_load` adaptation request:

```
$ curl 'http://localhost:1337/check_load?id=myEntity&type=myType' \
  -s -S -d @- <<EOF
OK - load average: 0.00, 0.00, 0.00|load1=0.000;1.000;1.000;0;
    load5=0.000;5.000;5.000;0; load15=0.000;15.000;15.000;0;
EOF
```

If debug logging level was enabled (editing file `config/logging.js`), we will see NGSI request sent to Context Broker.

2.2 Network Active Monitoring-NAM Adapter

2.2.1 Summary

Network Active Monitoring (NAM) Adapter is the component in charge of handling cross-domain active measurements between XIFI infrastructure nodes of the federation, providing a standard mechanism able to handle latency and bandwidth-related tests. Historical measurements represent results of regularly scheduled tests and cover one-way delay, jitter, one-way packet loss, and achievable throughput for a path. In addition, NAM Adapter also offers the possibility to request an on-demand test representing achievable throughput or one-way latency measurements between endpoints.

Figure 6 depicts in general terms how the adapter (components marked in green) is enclosed in the architecture of a generic XIFI Node, according to the latest version provided in deliverable *D1.5 - Federated Platform Architecture v2*. Those components with a direct relationship are marked in yellow: NGSI Adapter to parse the data to a standard format; and the Context Broker which belong to the Federation Monitoring.

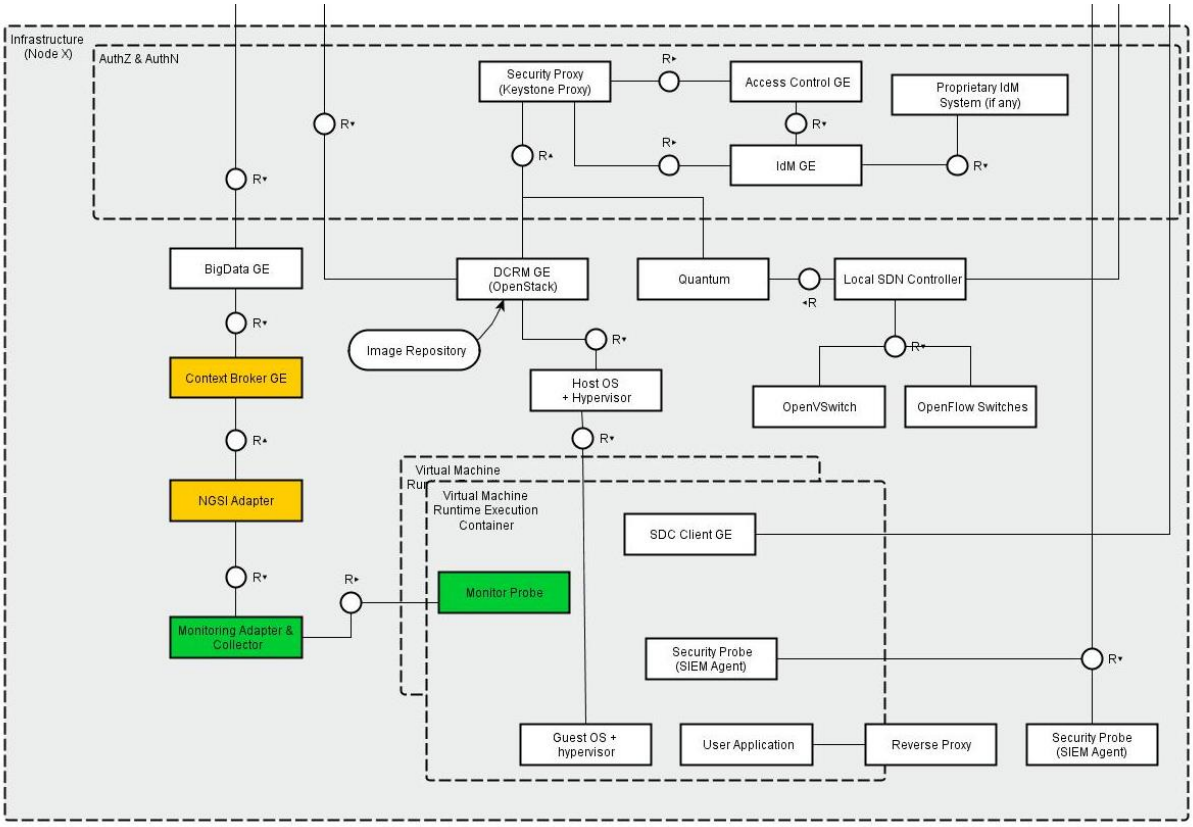

Figure 6: NAM Adapter within the Architecture of a XIFI Node

Tables below provide specific details with regard to the component:

Reference Scenarios	UC-5 - Network and Data Centre operations
Reference Stakeholders	• Developers: those who aim a) to be aware of the network performance in real time for experimentation b) to install a private XIFI monitoring node • Infrastructure owners: those who want to join XIFI federation or build a private XIFI node
Type of ownership	Development and Extension
Original tool	PerfSONAR [25]
Planned OS license	Apache License Version 2.0 [1]
Reference OS community	PerfSONAR community

Table 7: NAM Adapter Context Details

Consist of	• Monitoring Probes
	  ○ OWD Monitoring Probe
	  ○ BDW Monitoring Probe
	• Measurement Collectors
	  ○ OWD Measurement Collector
	  ○ BDW Measurement Collector

Depends on	• NGSI Adapter • XIFI Federation Monitoring ○ Orion Context Broker GE [23] • XIFI Monitoring Dashboard

Table 8: NAM Adapter Dependencies Summary

2.2.2 Component Responsible

Developer	Contact	Partner
Jose Gonzalez	jge@gatv.ssr.upm.es	UPM
Fernando Garcia	fgp@gatv.ssr.upm.es	UPM
Federico Alvarez	fag@gatv.ssr.upm.es	UPM

Table 9: NAM Adapter Reference Details

2.2.3 Motivation

A NAM instance is deployed in the cloud environment configured by each federated infrastructure to check the performance of inter-domain connectivity across nodes. Since the XIFI ecosystem is not compound by a single domain but a collection of domains (referred to with nodes), ensuring the connectivity between nodes is an important task to achieve.

NAM Adapter shall accomplish two main tasks:

- Provide the adaptation layer between monitoring probes at the lowest level, spread over multiple nodes, and the federation layer.
- Provide a distributed monitoring mechanism among infrastructure nodes, being able to trigger latency and bandwidth-related tests.

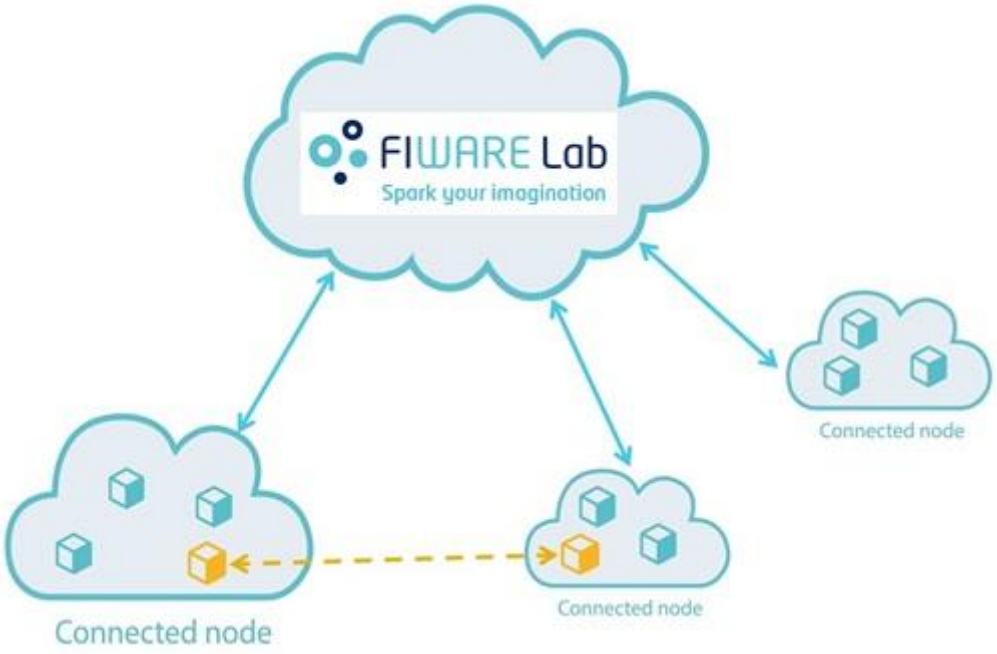

Figure 7: Multi-domain connectivity in the context of FIWARE Lab

To achieve such tasks, NAM Adapter requires two peers to establish the communication, i.e. end-to-end measurements (to better understand how an end-to-end test is performed, the reader may find detailed sequence diagrams in section 2.2.6.2).

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 are 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.

2.2.4 User Stories Backlog

Id	User Story Name	Actors	Description
1	Define the context	Component developer	Definition of the requirements and functionality to be accomplished by the component.
2	Define the architecture	Component developer	Identification and definition of the main functional blocks of the component's architecture.
3	Identify available technology	Component developer	Identification of available technology that enables to perform network active tests.
4	Play with PerfSONAR	Component developer	Setup and test PerfSONAR in a local environment. Extrapolate to a multi-domain test.

Id	User Story Name	Actors	Description
5	Assess PerfSONAR functionally	Component developer	Assessment into detail, finding strengths and weaknesses, bearing in mind the architecture requirements. Split into atomic functional modules, taking those which are relevant.
6	Refinement of the technology to the component's needs	Component developer	Definition of a detailed architecture, developing those functional blocks which shall fit the existing ones.
7	Create documentation	Component developer	Create those manuals to support the installation and usage of the software developed.
8	Test software and documentation	Component developer/ Infrastructure Administrator	Test the software and manuals as a standalone component.
9	Refine the component v1	Component developer	Refinement of those inconsistencies gathered from the feedback provided.
10	Integration with Federation Monitoring	Component developer	The integration of software elements with the Federation Monitoring through the NGSI Adapter.
11	Deployment stage v1	Component developer/ Infrastructure Administrator	Deploy the software package in each infrastructure node
12	Test integration v1	Component developer/ Infrastructure Administrator	Setup a full-packaged test environment, integrating the rest of associated components of the architecture.
13	Refine the component v2	Component developer	Refinement and enhancement of the component according to the performance of the integrated solution.
14	Deployment stage v2	Component developer/ Infrastructure Administrator	Re-deployment of the new version of the software package
15	Test integration v2	Component developer/ Infrastructure Administrator/ User	Test the component's performance all along the architecture.
16	Integration with Monitoring Dashboard	Component developer	To complete the integration with the XIFI architecture, it is required to fit the adapter with the XIFI Monitoring Dashboard.

Table 10: NAM Adapter User Stories Backlog

2.2.5 State of the art

2.2.5.1 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 [19], 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) [26] is a framework that implements these protocols.

The web service-based infrastructure presented in PerfSONAR was developed 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. This framework is made up of several 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.
- Measurement Archive (MA) services are used to record and publish historical monitoring data which are stored in an archive.
- The Lookup Service (LS) enables users to discover other services (e.g. MP and MA) 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.

Each PerfSONAR MP uses specific network measurement tools to perform the measurement of the network characteristic between the selected end points. For example, Iperf [11] is used by the Bandwidth Test Controller (BWCTL) service [2] to fulfil TCP or UDP achievable throughput measurements; and the One-Way Active Measurement Protocol (OWAMP) service [24] is used to retrieve one-way delay, jitter and packet loss measurements.

Though PerfSONAR represented the model to be inspired by, the available open-source implementations do not fit the expected XIFI architecture. Several assessments were carried out to determine compatibility of architectures, but the results were not satisfactory. Therefore, NAM Adapter's software package was conceived from scratch, fulfilling those requirements and constraints proper of the general architecture, and leveraging useful concepts and tools from the PerfSONAR framework.

2.2.6 Architecture

Diagram below (Figure 8) provides a graphical description of the interactions among internal and surrounding modules of a single NAM Adapter instance.

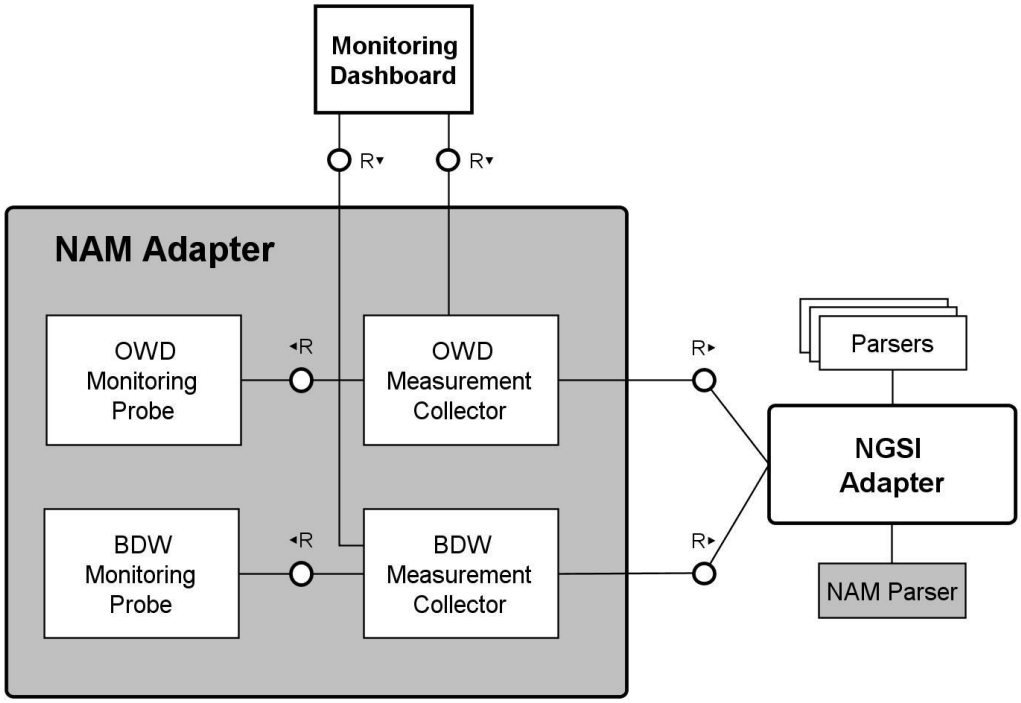

Figure 8: NAM Adapter General Architecture

Some remarks to be taken into consideration with regard to the design depicted in Figure 8:

- Each instance of NAM Adapter is expected to be deployed in a physical host, in principle in a monitoring node of the cloud environment configured by the Infrastructure Owner. The installation in a virtual resource would be a feasible choice, but it may not provide accurate results. More details can be found in the corresponding section Installation Manual (section 2.2.8).
- In relation to the previous point, the number of instances per domain will directly depend on the amount of nodes to be monitored.
- NGSI Adapter requires several parsers to process the raw data coming from the collectors of different adapters in XIMM. Concretely, NAM's associated parser (highlighted in grey as well) is not part of the adapter as it is. However, its specification is necessary to fulfil the integration process with the rest of the architecture.

2.2.6.1 Basic actors

- **Monitoring Probes**

The Monitoring Probes are tools used to actually perform the measurement tests between given nodes. They play the role of servers providing the clients with the raw network monitoring data. The interface to interact with these tools is command line-based.

Technically speaking, these probes should not been taken as a part of the adapter by definition. However, monitoring systems, such as the case of Nagios [13], do not include these specific tools. Therefore, NAM's software package requires including a couple of probes by default to enable tests:

- *One-Way Delay (OWD) Monitoring Probe*: Taking PerfSONAR's OWAMP service [24] as a reference, NAM's specific OWD Probe overcomes some functional constraints, which in terms of efficiency are not optimal, to enhance the operability of the service.

- *Bandwidth (BDW) Monitoring Probe*: Following the Internet 2's PerfSONAR distribution with regard to bandwidth-related tests (BWCTL [2]), NAM's BDW Probe is based on the network throughput tool Iperf [11] according to reliability test results.

- **Measurement Collectors**

The Measurement Collectors are the core modules within the adapter. They will be the responsible actors for collecting the data generated by the previously described probes (by playing the role of clients), processing and publishing the raw data to the NGSI Adapter. Same as the probes, there are two types, OWD and BDW (see Figure 8), according to the data they deal with.

Figure 9 provides a detailed view of the internal architecture of the adapter, focusing on the elements composing the collector.

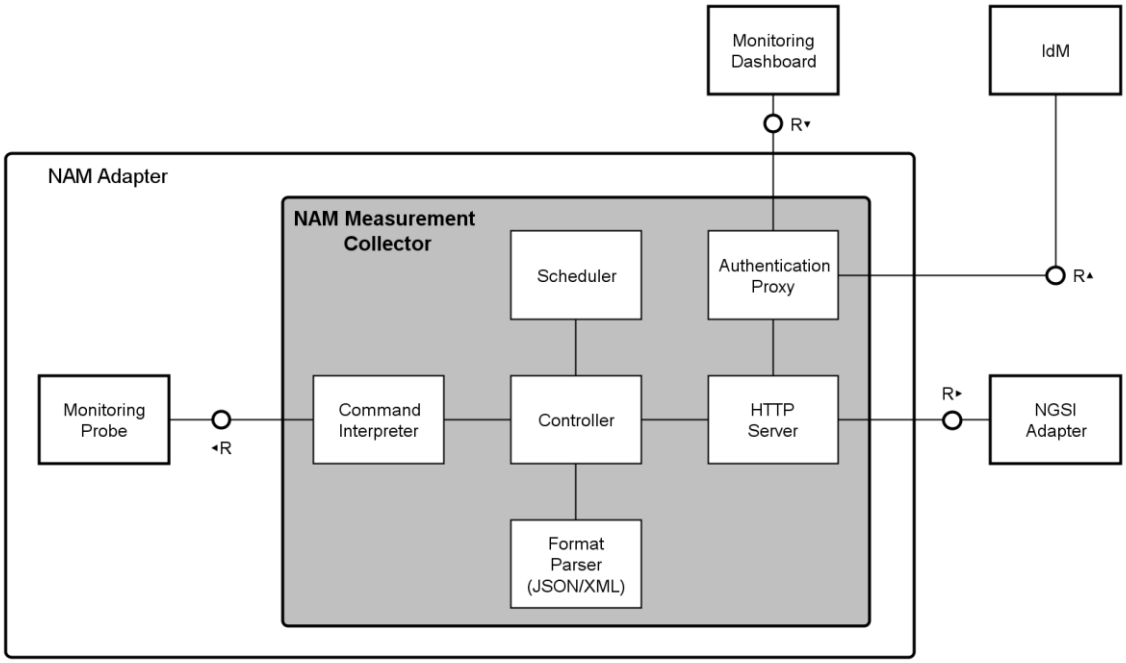

Figure 9: NAM Measurement Collector - Internal Architecture

A collector consists of the following sub-modules:

- *Command Interpreter* is in charge of dealing with the probe through command line-based operations
- *Format Parser* adjusts the result obtained from the command to a standard response, e.g. JSON or XML format
- *Scheduler* is the element responsible for the timing in scheduled tests, triggering the process when the setup time slot is reached
- *HTTP Server* will handle the exchange of request/responses
- *Authentication Proxy*, which intercepts external requests to validate if they represent an authenticated access to the services through OAuth-based [18] mechanisms. This

proxy is based on the implementation released in FIWARE [9]

- *Controller* is the central entity which manages the sub-modules

- **NGSI Adapter**

As it is described in section 2.1, the NGSI Adapter is the component in charge of standardizing collected raw data into a FIWARE NGSI-10 [7] format, compatible with the Orion Context Broker [23] of the Federation Monitoring. To accomplish such duty, the NGSI Adapter leverages the NAM Parser implemented to accommodate both data formats.

- **Monitoring Dashboard**

Monitoring Dashboard is the Graphical User Interface (GUI) to handle monitoring information collected and processed from the federation. Such dashboard shall be interactive-ready to request and fetch data from the NAM-driven tests.

- **Identity Management (IdM)**

In order to protect the access to the REST-based API provided by the NAM Adapter, an authentication mechanism is integrated to comply with the XIFI Federated Identity Management [32]. Hence, the privilege to gain access will be granted from FI-Lab to specific users.

2.2.6.2 Main interactions

- Interactions to provide historical results

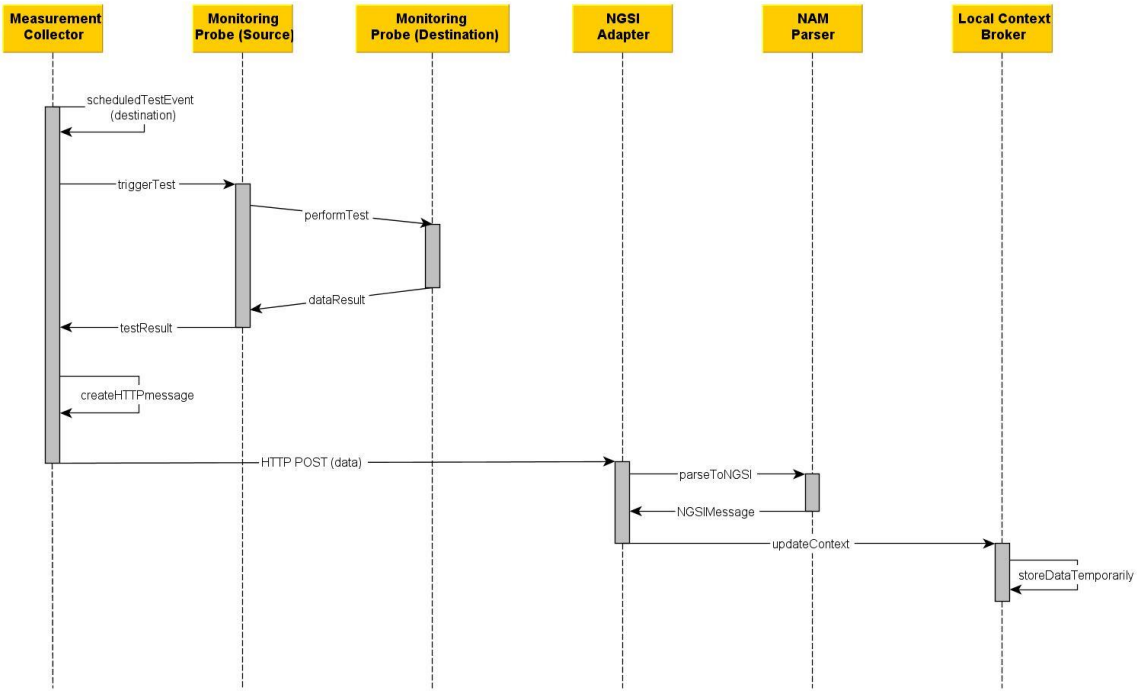

Figure 10: Sequence Diagram – Historical Results

- Interactions to check on-demand tests

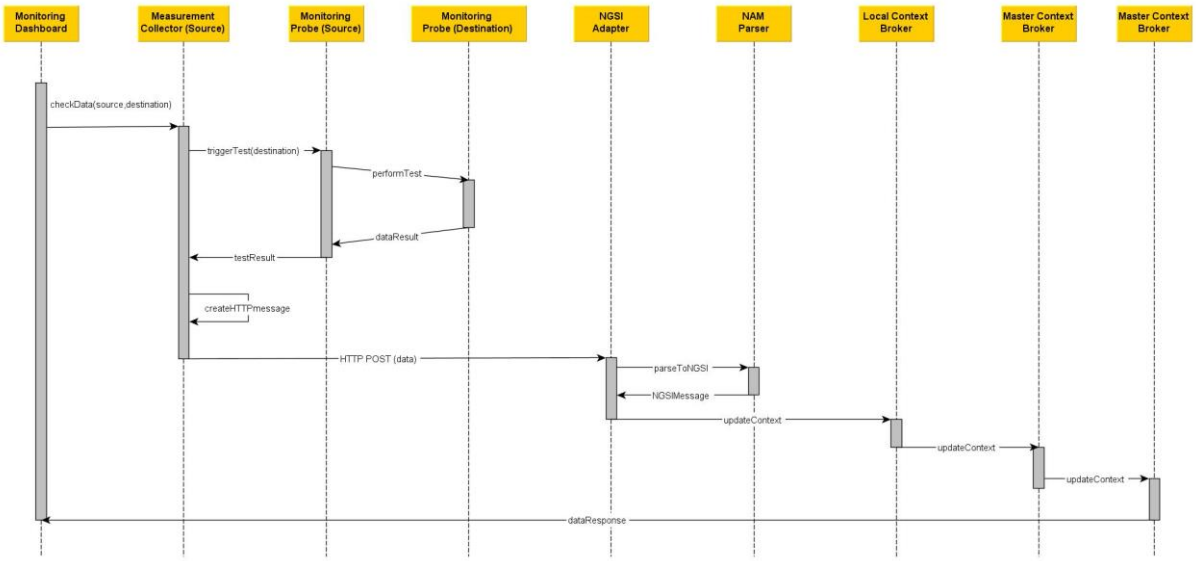

Figure 11: Sequence Diagram - On-demand Test

2.2.7 Release Plan

Version ID	Milestone	User Stories
v0.1	M3	1, 2, 3
v1.0	M6	4, 5
v2.0	M9	6, 7
v2.1	M12	8, 9
v3.0	M15	10, 11, 12, 13
v3.1	M18	14, 15, 16

Table 11: NAM Adapter Release Plan

2.2.8 Test Cases

To check the software component, several unit tests have been packed in the file *NAMadapter_test.js* located at: https://xifisvn.res.eng.it/wp3/software/NAM_Adapter/Trunk/Test/.

To run such batch of tests, type the following commands:

```
root@{directory}/NAM_Adapter/Trunk#npm install mocha -g
root@{directory}/NAM_Adapter/Trunk#mocha Test/NAMadapter_test.js
```

If everything worked properly, the message below shall be displayed:

```
   ......
   10 passing (34s)
root@{directory}/NAM_Adapter/Trunk#
```

2.2.9 Installation Manual

This section aims to guide the reader through the properly configuration of the NAM Adapter.

2.2.9.1 Requirements

- A single NAM Adapter instance will be integrated in the target infrastructure to be federated. Such adapter is expected to be deployed in a host with monitoring purposes (also known as Monitoring Node) within the cloud-based environment settled by the Infrastructure Owner.
- NAM Adapter is a light-weight software package; hence it does not require special requirements in terms of hardware. If such installation is meant for testing, the adapter can be installed in any available resource.
- The reader shall bear in mind that additional instances may be deployed to offer a more fine-grained picture by replicating the procedure to follow in this documentation.
- A physical host is recommended to deploy the adapter. However, the installation can be performed within a VM.
- The standard operating system is Ubuntu 12.04 LTS (64-bits). Tests have also been fulfilled with CentOS.
- In order to engage the NAM Adapter with the rest XIFI Monitoring Architecture, it is required that both NGSI Adapter and Context Broker instances are installed previously in a resource of the infrastructure. NAM instance will require pointing such components during the configuration.
- Prior the installation of the instance, it is required to install some additional software packages (if not yet present in the host) by typing the following command lines:

Installation of Node.js

```
$ sudo apt-get update
$ sudo apt-get install -y python-software-properties python g++
make
$ sudo add-apt-repository -y ppa:chris-lea/node.js
$ sudo apt-get update
$ sudo apt-get install nodejs
```

Installation of MongoDB
`$ sudo apt-get install mongodb`

Installation of Iperf
`$ sudo apt-get install iperf`

Installation of NTP
`$ sudo apt-get install ntp`

2.2.9.2 Software Repository

- NAM Adapter: https://xifisvn.res.eng.it/wp3/software/NAM_Adapter/Trunk/

2.2.9.3 Setup Guidelines

Since the NAM Adapter software package is also located in the NPM Package Manager [15], the component can be simply installed in the selected host by typing:

```
$ sudo npm install nam_adapter
```

If required, in order to uninstall the instance, proceed by typing:

```
$ sudo npm uninstall nam_adapter
```

If the user credentials are required to be reset as well, the internal database must be deleted:

```
$ mongo
> use NAM_adapter
> db.dropDatabase();
```

2.2.10 User Manual

2.2.10.1 API Specification

NAM Adapter API specification is fully described in http://docs.namapi.apiary.io/.

2.2.10.2 Handbook

These guidelines will support the configuration of the NAM Adapter as a standalone software package.

Running the software:

- In order to run the component, move to the right directory:

```
$ cd node_modules/nam_adapter
```

- The following command line will start the component:

```
$ sudo ./NAMadapter start -d
```

- The command line to stop the component (if needed) is:

```
$ sudo ./NAMadapter stop
```

Configuration of general attributes:

Once all the required software has been installed, it is time to configure the adapter properly.

- The component offers a Web-based configuration user interface. To access, type in the Web browser:

```
http://{IPaddress}:3000/config
```

  - *IPaddress* refers to the host's address where the NAM is installed.
  - Default credentials can be changed to avoid further security issues in forthcoming steps:
    - User: userxifi
    - Password: xifiMaster2014

- Configuration of NAM Server Parameters:

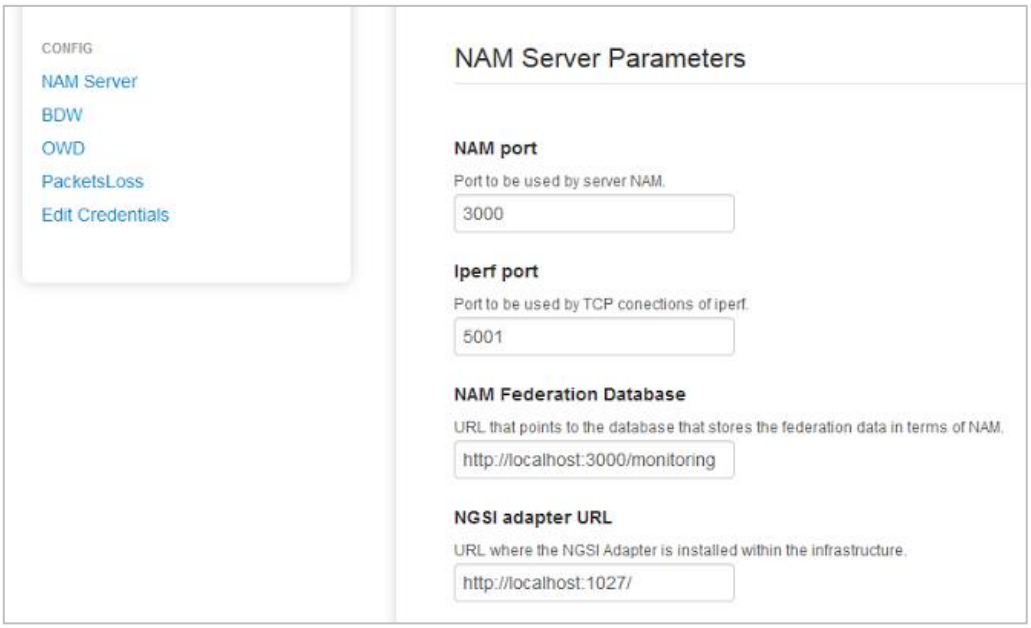

Figure 12: Configuration of NAM Server Parameters

Figure 12 illustrated the configurations of the server parameters that are described as follow:

- **NAM + Iperf Ports**: Default ports to provide the NAM services. Such ports can be modified if needed, but this would imply the notification to the federation administrator.
- **NAM Federation Database**: URL that points at the database that stores the federation data in terms of NAM instances. It is used to inform that a new instance is being federated. This endpoint will be provided by a contact point from the federation.
  - By default, such URL points to *localhost*. This means that the NAM instance refers to its local database. Hence, the attribute must be changed to the correct value.
- **NGSI Adapter URL**: URL where the NGSI Adapter is installed within the cloud environment. Such adapter is required because is the link with the overall monitoring architecture.

- Configuration of Host Parameters:

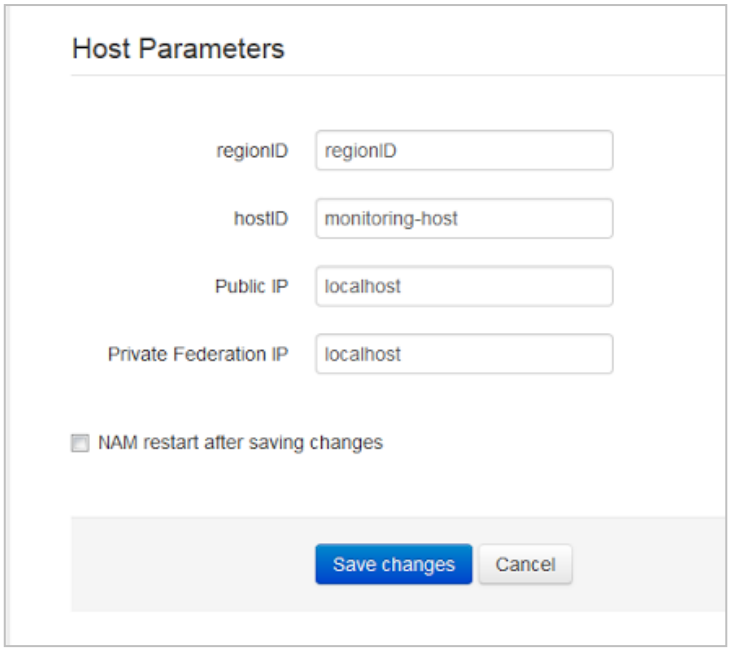

Figure 13: Configuration of NAM Host Parameters

Figure 13 shows the configuration of host parameters that are described as follow:

- **regionID**: Attribute which identifies the node within the federation. Such identifier will be provided by a contact point from the federation.
  - The expected value shall be the name of the town where the node is, e.g. "Berlin"
- **hostID**: Attribute which identifies univocally the host where the NAM instance has been installed.
  - The expected value here is the "Hostname" attribute of such host, which shall be unique within the domain. Given that, the Infrastructure Owner should be familiar with this environment
  - During the installation, the NAM package retrieves the value of this attribute and includes it in the configuration file. Hence, the value displayed in this configuration page should coincide
- **Public IP**: Public IP address assigned to the host where the NAM instance is installed.
- **Private Federation IP**: Private IP address assigned to the host according to the MD-VPN plan.

- After configuring the attributes, click on "*NAM restart after saving changes*" and finally "*Save changes*".

Configuration of general attributes:

NAM Adapter provides 2 different services, BDW (Bandwidth) and OWD (One-way Delay), that are supported both in real-time and scheduled modes. The following attributes will set the scheduling plan for BDW (Figure 14) and OWD services, configuring the NAM instance to request tests periodically to another specific NAM instance (or a group of them) of the federation. Such settings will provide historical results between the NAM instance and other ones with special relevance.

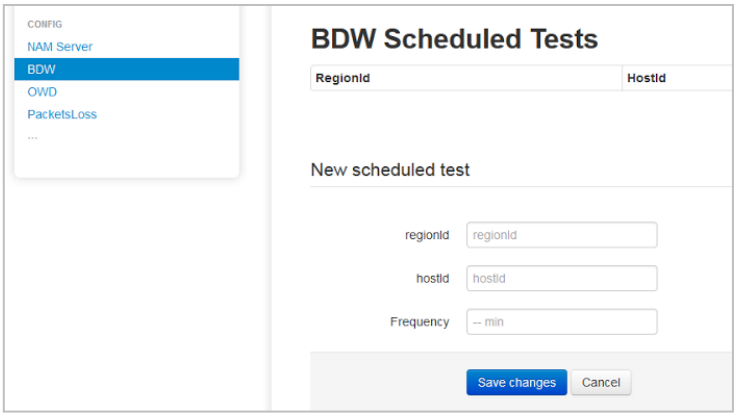

Figure 14: BDW Scheduled Tests

- Default Tests: since the connectivity to the Master Node is a relevant value, the instance's configuration file includes a pair of pre-configured scheduled tests (both BDW and OWD) that check Master Node's NAM instance.

RegionId	HostId	Frequency
Trento	monitoring	480

Figure 15: Default values for NAM Scheduled Tests

- To configure a new Scheduled Test:
    - **regionID**: Identifier of the region to test periodically
    - **hostID**: Identifier of the host to test periodically (REMINDER: the pair regionID + hostID identifies a host within the federation univocally)
    - **Frequency**: Value (expressed in units of minutes) which states how often such test shall be triggered. Default value is "*480*" minutes, which means three times per day.

2.3 Datacenter & Enabler Monitoring-DEM Adapter

2.3.1 Summary

Datacenter and Enablers Monitoring (DEM) Adapter is responsible for performing the monitoring – along with the necessary adaptation mechanisms– of the Virtual Machines and the Generic Enablers deployed within the XIFI federation.

The DEM Adapter provides the following features:

- An abstraction and adaptation layer that caters for gathering monitoring data related to the Virtual Machines (VMs) and the Generic Enablers (GEs) deployment across XIFI federation.
- Integration with the pertinent FIWARE components (NGSI Adapter/ Orion Context Broker), providing raw data to the NGSI Adapter which in turn translates the data into context-based format understood by the Context Broker.
- An instantiation mechanism for DEM Adapter to XIFI nodes (through a deployment process via GE images).
- An auto-registration service for newly deployed VMs/GEs in XIFI federation environment.
- A lightweight version that caters for specific GE deployment restrictions, such as deployment in a XIFI node using a private IP address, being also compatible with the networking architecture of FIWARE by offering isolation among users/tenants deployment.
- Supporting several operating systems, such as Ubuntu and CentOS.
- A configuration file for the application developer to customize monitoring options.

In order to be compliant with all issues related to installation, networking and management issues from the infrastructure owners viewpoint, DEM is offered in two versions, depending on the specific needs of an application developer. The first version of DEM (called active DEM) is an adapter that offers the capability to the application developer to gather monitoring data from the VM or GE, deployed within the XIFI federation, on demand. On the other hand, given the limitations of IPv4 addressing plan, another version of DEM component (called passive DEM) has been developed, tailored for VM/GE deployments where a public IP address is not mandatory or available. Although the latter version does not provide the capability to query on demand and retrieve monitoring data from the VM/GE, it offers several advantages related to the XIFI/FIWARE networking architecture (deployment of GEs in a private network), public addressing limitations, providing a lightweight but powerful solution for application developers and infrastructure owners.

Figure 16 depicts the context where DEM adapter is placed in the architecture of a XIFI Node.

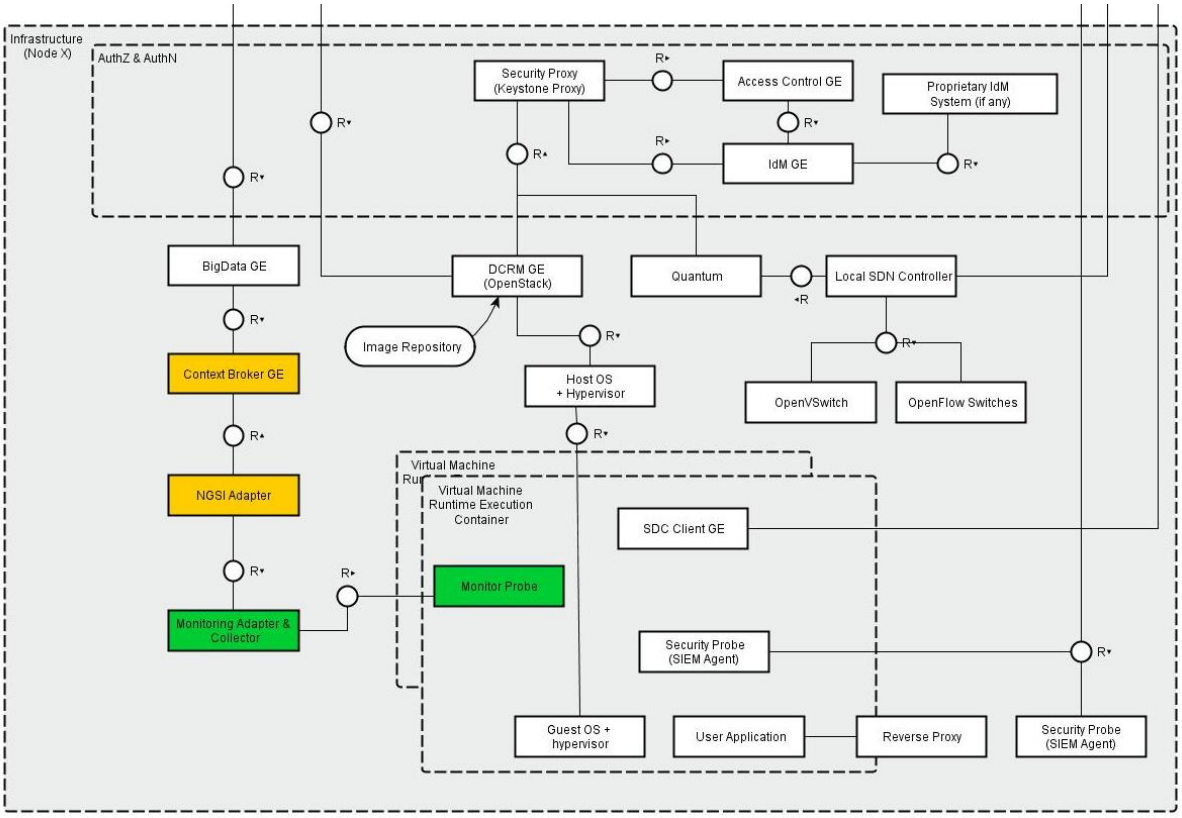

Figure 16: DEM Adapter within the Architecture of a XIFI Node

Tables below provide specific details with regard to the component:

Reference Scenarios	UC-5 – Network and Data Centre operations
Reference Stakeholders	This component aims at providing monitoring data from VMs and GEs deployed within XIFI federation.
Type of ownership	Extension/New component
Original tool	Nagios/None
Planned OS license	Apache License version 2.0
Reference OS community	Nagios

Table 12: DEM Adapter Context Details

Consist of	• DEM active
	  ○ Nagios Core, nrpe, plugins
	  ○ Auto Registration Service (ARS) client on VMs
	  ○ Auto Registration Service (ARS) server installed alongside Nagios Core
	  ○ DEM adapter on individual VM
	• DEM passive
	  ○ Nagios plugins
	  ○ DEM adapter on individual VM

Depends on	• DEM active
	  ○ NGSI Adapter
	• DEM passive
	  ○ OpenStack

Table 13: DEM Adapter Dependencies Summary

2.3.2 Component Responsible

Developer	Contact	Partner
P. Trakadas	ptrak@synelixis.com	SYNELIXIS
P. Karkazis	pkarkazis@synelixis.com	SYNELIXIS

Table 14: DEM Adapter Reference Details

2.3.3 Motivation

Monitoring of virtual machines is a critical part in every cloud environment, allowing both infrastructure owners and application developers to collect data and services. In a federated cloud environment, such as XIFI, state-of-the-art solutions, as will be discussed later, are not directly applicable at the moment. This primarily stems from the fact that FI-PPP has been built around the notion of Generic Enablers (GEs) that are not inherently supported by current cloud environments, such as OpenStack. Moreover, federation of pan-European cloud facilities adds extra complexity, requiring special care in terms of monitoring and interoperable solutions spanning from supporting different operating systems to networking architectures followed within the XIFI nodes. In this context, one has to determine the requirements and carefully select and develop tools and adapters needed to support monitoring of several resources from virtual machines collected by heterogeneous cloud environments.

DEM adapter provides many advantages:

- it is compliant with the FIWARE-based developments, such as NGSI adapter and Orion Context Broker [23]
- it includes monitoring solutions already installed on the majority of cloud infrastructures
- it is easily extensible to other monitoring tools that may be present in the future, targeting sustainability of XIFI

- it is tailored for scalability needs of XIFI
- it offers different software versions of DEM adapter suited to different operating systems
- it is customizable according to user needs
- it is suitable to any networking solutions within a XIFI node
- it provides isolation among several users/tenants within private networks

2.3.4 User Stories Backlog

Id	User Story Name	Actors	Description
1	Install Nagios NRPE	Component developer	Nagios NRPE is deployed on physical server and compatibility with DEM is verified
2	Initial set of measurements in a single node (physical server)	Component developer	DEM can retrieve an initial set of measurements from a single node
3	Install Nagios NRPE on VM	Component developer	Nagios NRPE is deployed on VM and compatibility with DEM is verified
4	Initial set of measurements in a VM	Component developer	DEM can retrieve an initial set of measurements from a VM
5	Initial set of measurements in a GE	Component developer	DEM can retrieve an initial set of measurements from a VM, where a GE is deployed
6	Integration with NGSI adapter	Component developer	Compliant with the XIMM architecture and the inclusion of NGSI adapter.
7	Architecture refinement	Component developer	Alignment with Monitoring Data Model and APIs.
8	Measurements from multiple VMs/GEs	Component developer	DEM retrieves information from multiple VMs/GEs
9	Measurements from multiple VMs/GEs in multiple nodes	Component developer	DEM retrieves information from multiple VMs/GEs in multiple nodes
10	Integration with Federation Monitoring	Component developer	Integration of software elements with the Federation Monitoring is ready through the NGSI adapter.

Id	User Story Name	Actors	Description
11	Final set of documentation ready	Component developer	Release a stable and enhanced version of manuals and procedures.

Table 15: DEM Adapter User Stories Backlog

2.3.5 State of the art

The objective of this section is to analyse the requirements arising from a federated cloud infrastructure standpoint, such as XIFI, and come up with a promising solution that goes beyond the state-of-the-art.

- The most probable case in a federated service cloud environment is the one in which each infrastructure owner is already using his own monitoring system. Background solutions mainly include Nagios [13], Zabbix [33] and OpenNMS [20] as the most representative tools. Moreover, taking for granted that new nodes will join the XIFI federation, it is likely that the aforementioned list of monitoring tools will be extended. Thus, the first requirement that has to be faced is the definition of an abstract approach towards a monitoring adapter that will 'homogenise' data collected by different monitoring tools and present them in a consistent and compatible fashion.
- Apart from the requirements stemming from the heterogeneity of monitoring tools, as described above, there is also a need to store monitoring data both on node and federation level. As an additional requirement, the proposed solution must be consistent with the overall XIFI architecture.
- Furthermore, the proposed solution must also provide means for the proper operations on these data, such as aggregation, filtering, etc. Thus, the monitoring solution will be capable of being connected with other modules developed within XIFI, such as Scalability Manager, Federation Monitoring and SLA Management.
- An additional requirement is set by the demand for accessing both real-time and off-line data. In this perspective, the DEM solution must provide mechanisms for retrieving such data in an easy way.
- Generic Enablers and Specific Enablers (services developed and deployed by the use case projects/developers) are based on different operating systems. Thus, in order to cover all application developers' requirements, DEM adapter must support said operating systems to be installed in, such as Ubuntu and CentOS.
- Finally, there is a need for VM/GE isolation among users/tenants in a federated cloud environment. Solutions regarding monitoring have to take into consideration such circumstances and provide solutions tailored to application developers and infrastructure owners' needs.

As XIFI is part of the FI-PPP programme and is closely cooperating with other projects, it was firstly assessed whether already developed tools, adapters or FIWARE GEs could fulfil such requirements. In this perspective, NGSI Adapter described in section 2.1 fulfils the requirements set above and has been selected as a solution for XIMM adapters, being a key component to allow incorporating monitoring and metering mechanisms in order to be able to constantly check the performance of the VMs/GEs, providing a simple architecture that can be easily extended to collect data for other required needs.

2.3.6 Architecture

As stated earlier, DEM is the component responsible for collecting and publishing monitored data from VMs and GEs. DEM adapter is valuable for XIFI interests since it enables to check resources, such as processor load, RAM utilization, disk usage and number of running processes. Besides, service checking may be oriented to determine the actual performance of those FIWARE GE instances deployed, providing the federation with service information.

2.3.6.1 DEM Active Adapter

The DEM Active Adapter leverages on the data obtained from already deployed monitoring tools in charge of running the actual monitoring processes. There are several popular systems that can be taken into consideration for this purpose, such as the case of Nagios. Most of them rely on standalone programs, called 'plugins', to perform the measurements. The collection of the monitored data is not useful until an entity defines where the data (coming from the different sources of information, e.g. VMs) can be consumed by interested users.

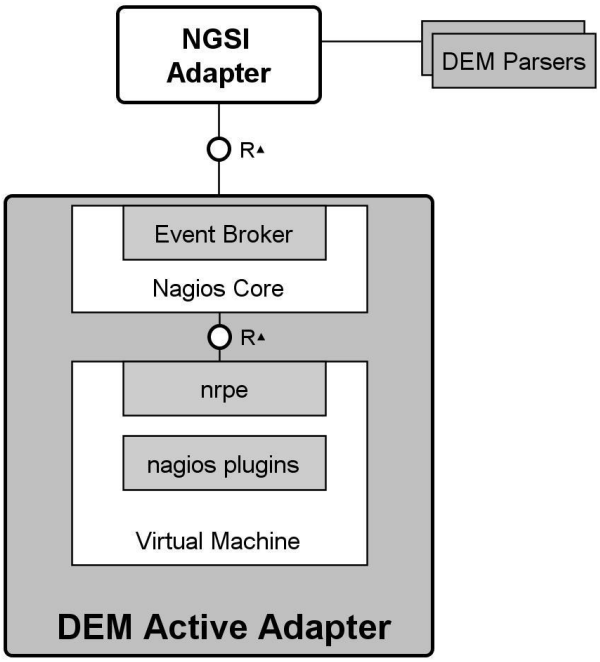

Figure 17: DEM Active Adapter General Architecture

The DEM Adapter stands as the 'glue' between the source of monitoring information (plugins on virtual servers) and the NGSI Adapter that publishes the data to the Context Broker, providing homogeneous representation of the data (based on an already defined data model) and being compatible with the NGSI-10 API [7] provided by FIWARE.

2.3.6.2 DEM Passive Adapter

There are numerous occasions where a GE or a service related to a GE service must not use a public IP address, but, instead, being installed in a private network within a XIFI node. Under these circumstances, the application developer will not have the ability to query on demand the DEM adapter, but, on the other hand, wants to be able to check the status of his VM/GE. DEM Passive Adapter has been developed in order to cope with such restrictions, allowing the application developer

to monitor his deployed VMs/GEs as well as being able to configure DEM functionality according to his needs. Moreover, installation of passive DEM comes within the GE image and thus no human intervention is needed.

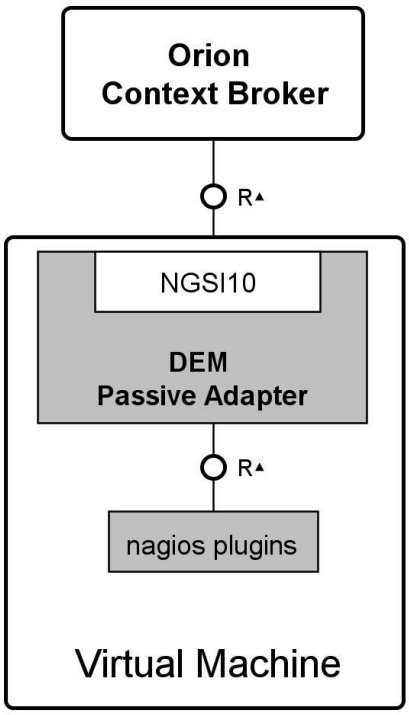

Figure 18: DEM Passive Adapter General Architecture

Additionally, DEM passive adapter bypasses the NGSI Adapter, sending monitoring data directly to the Orion Context Broker. Finally, DEM Passive Adapter can be installed in Ubuntu and CentOS based applications covering all GEs developed so far in FIWARE, where monitoring services can be extended without the need for respective parsers.

2.3.6.3 Main interactions

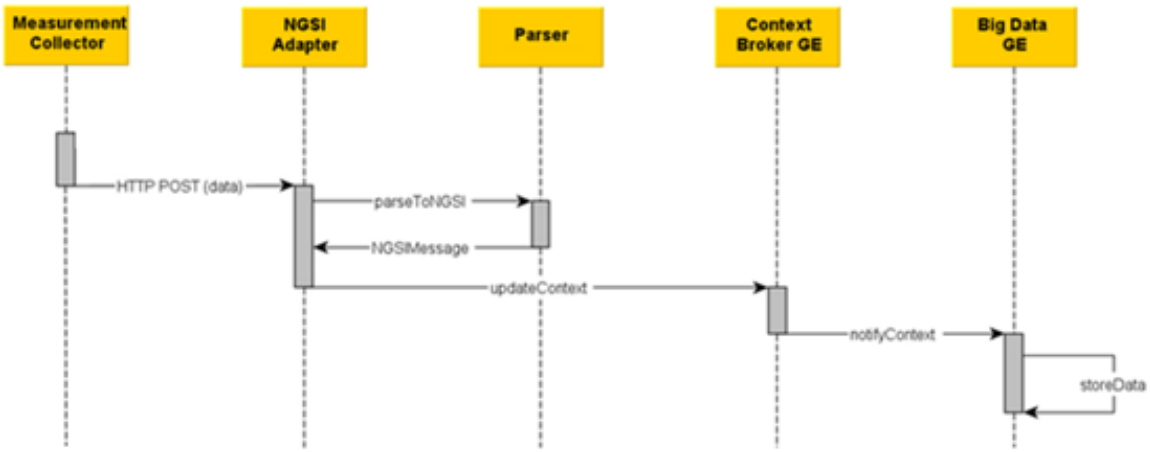

Figure 19: DEM Adapter - Sequence Diagram

2.3.7 Release Plan

Version ID	Milestone	User Stories
0.1	31.08.2013	1,2
1.0	31.12.2013	3,4,5
1.5	31.03.2014	6,7,8,9
2.0	30.09.2014	10,11

Table 16: DEM Adapter Release Plan

2.3.8 Test Cases

This section includes detailed functional verifications related to DEM Active Adapter.

2.3.8.1 Unit Test 1 - Verification of Nagios Core installation

Nagios Core is responsible for gathering monitoring data from remote hosts (VMs and physical) through already installed nrpe daemon.

In this test, we assume that Nagios core is installed and running in a server named NCoreSrv and we are going to check the status of two services: Apache server and Nagios Core.

- **Check Apache service**. In order to check that Apache server is running, type the following command. The second line indicates the response.

```
root@NCoreSrv:~# ps aux | grep apache2
root  5125  0.0  0.2  24860 6880 ? Ss   19:09   0:00
/usr/sbin/apache2 -k start
```

If there is no response, this means that Apache server is not running and you must start the service manually by typing:

```
root@NCoreSrv:~# service apache2 start
```

- **Check Nagios Core service**

```
root@NCoreSrv:~# ps aux | grep nagios
 nagios 32758  0.0  0.0 832824 2008 ? Ssl  Nov14  30:11
/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

If there is no response, this means that Nagios Core is not running and you must start the service manually by typing:

```
root@NCoreSrv:~# service nagios start
```

- **Check Nagios Web interface.** While the two aforementioned services are up and running, we must be able to access the Nagios web interface at NCoreSrv URL (http://localhost/nagios/).

2.3.8.2 Unit Test 2 - Verification of Nagios plugin correct installation

All monitored nodes must have a tool library responsible for gathering monitored data from each host. Nagios plugins are placed in /usr/local/nagios/libexec/. It is noted that a user is able to develop and add custom tools as well. In this unit test, we will check the existence and functionality of Nagios plugins locally.

We check one of the plugins which returns the number of the login users in the system.

```
root@ubu-0:~#  /usr/local/nagios/libexec/check_users -w 5 -c 10
 USERS OK - 2 users currently logged in |users=2;5;10;0
```

2.3.8.3 Unit Test 3 - Verification of nrpe connectivity to Nagios Core

In this unit test, we check the connection between the local host (IP=192.168.1.231), where Nagios Core is installed and a remote host (IP=192.168.1.244), e.g. a VM. In order to collect data from remote host, we use nrpe add-on which is a daemon running in all remote hosts, allowing access to local resources.

- **Verify proper operation of nrpe in a remote host.** In a remote node, we check that nrpe daemon is running under xinetd server.

```
root@192.168.1.244:~# netstat -at | grep nrpe
 tcp    0      0      *:nrpe       *:*    LISTEN
```

Next, make sure that nrpe daemon is functioning properly. Run check_nrpe plugin which returns the version of installed nrpe.

```
root@192.168.1.244:~# /usr/local/nagios/libexec/check_nrpe -H
localhost
 NRPE v2.15
```

- **Remote test.** nrpe daemon is installed asa service under xinetd server. In order to connect to nrpe from a monitoring host, it is deemed mandatory to define host's IP address to xinetd server. So, in remote node, open /etc/xinetd.d/nrpe file and check if monitoring host's IP is defined in only_from property (in our example, monitoring host has an IP = 192.168.1.231).

```
 only_from = 127.0.0.1 192.168.1.231
```

Keep in mind that if you make any changes in nrpe file, you must restart xinetd server.

```
service xinetd restart
```

Now, from monitoring host (Nagios Core) get current user status through nrpe. In this example, we request data for the logged users, cpu load and number of processes running.

```
root@192.168.1.231:~# /usr/local/nagios/libexec/check_nrpe -H
192.168.1.244 -c check_users
USERS OK - 2 users currently logged in |users=2;5;10;0

root@192.168.1.231:~# /usr/local/nagios/libexec/check_nrpe -H
192.168.1.244 -c check_load
OK - load average: 0.00, 0.01, 0.05|load1=0.000;15.000;30.000;0;
load5=0.010;10.000;25.000;0; load15=0.050;5.000;20.000;0;

root@192.168.1.231:~# /usr/local/nagios/libexec/check_nrpe -H
192.168.1.244 -c check_total_procs
PROCS OK: 61 processes
```

2.3.8.4 Unit Test 4 - Verification of Nagios Event Broker functionality

In this unit test, we check the proper installation and functionality of livestatus unixsocket which is used for communication between DEM adapter and NagiosCore Server.

- In order check whether unix socket has been initialized properly, restart Nagios service in monitoring host.

```
root@192.168.1.231:service nagios restart
```

Then, check /usr/local/nagios/var/nagios.log file for any errors. If unixsocket is set up successfully, you must see something like this in the abovementioned log.

```
root@192.168.1.231:/usr/local/nagios/var# cat nagios.log | grep
'Event broker module'
[1386869864] Event broker module '/usr/local/lib/mk-
livestatus/livestatus.o' initialized successfully.
```

Next, we execute a test query requesting from Nagios service the alias and IP addresses of all the remote hosts. Create a file named testQ with the following context:

```
GET hosts
Columns: alias address
```

Execute the query (testQ):

```
root@192.168.1.231:/usr/local/nagios# unixcat < testQ
/var/lib/nagios/rw/live
Ubuntu 10.04 LTS;192.168.1.65
localhost;127.0.0.1
Ubuntu 12.04;192.168.1.244
```

2.3.8.5 Unit Test 5 - Verification of connectivity between DEM and ORION Context Broker

In this unit test, we check the configuration and the update process of Orion Context Broker from DEM side.

- First, make sure that DEM component has been configured properly. Open DEM's configuration file located at /usr/local/nagios/nagiosNGSi/nagiosNGSi.cfg. These settings must be appropriately configured (e.g. conRegistration_url and regionid).

```
polling_interval=5000
conUpdate_url=http://192.168.1.242:1026/NGSI10/updateContext
conRegistration_url=http://192.168.1.242:1026/NGSI9/registerContext
unixSocket=/var/lib/nagios/rw/live
logfile=log/nagiosNGSi.log
regionid=Trento
debug=true
```

Start DEM service:

```
root@192.168.1.231:/usr/local/nagios/nagiosNGSi# ./nagiosNGSi.sh
start
Starting nagiosNGSi ...
nagiosNGSi started …
```

Next, open and check /usr/local/nagios/nagiosNGSi/log/nagiosNGSi.log for errors. If all gone well, you must see the following output.

```
INFO: GET Services...
Dec 10, 2013 10:51:43 AM nagiosngsi.XML_File buildXMLupdate
INFO: File update_192.168.1.65.xml build completed.
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Post: 0 Web Response status code: 200
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Orion's Response: 200 OK
Dec 10, 2013 10:51:43 AM nagiosngsi.XML_File buildXMLupdate
INFO: File update_127.0.0.1.xml build completed.
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Post: 1 Web Response status code: 200
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Orion's Response: 200 OK
Dec 10, 2013 10:51:43 AM nagiosngsi.XML_File buildXMLupdate
INFO: File update_192.168.1.244.xml build completed.
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Post: 2 Web Response status code: 200
```

```
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Orion's Response: 200 OK
```

In every operation cycle, DEM adapter performs the following action steps:

- Collects data from all monitoring nodes via Nagios core (GET Services).
- Builds an updated xml file (File update_192.168.1.65.xml build completed) per monitoring host.
- Posts the xml to Context Broker (Post: 0 Web Response status code: 200).
- Gets status response from CB (Post: 0 Web Response status code: 200).

2.3.8.6 Unit Test 6 - Verification of Registration and ContextUpdate (NGSI9/10) API calls

In this unit test, we check the node's registration and procedures updating from CB's side. In this example, a remote host (IP address 192.168.1.244) has been registered in CB (IP address 192.168.1.242).

- Get registered services for host 192.168.1.244

```
root@192.168.1.231:# curl
192.168.1.242:1026/ngsi10/contextEntities/192.168.1.244 -s -S --
header 'Content-Type: application/xml' | xmllint --format -<?xml
version="1.0"?>
<contextElementResponse>
 <contextElement>
    <entityId type="" isPattern="false">
     <id>192.168.1.244</id>
    </entityId>
    <contextAttributeList>
     <contextAttribute>
       <name>Region ID</name>
       <type>String</type>
       <contextValue>Trento</contextValue>
     </contextAttribute>
     <contextAttribute>
       <name>CPU Load</name>
       <type>String</type>
       <contextValue>0.00</contextValue>
     </contextAttribute>
     <contextAttribute>
       <name>Current Users</name>
       <type>String</type>
       <contextValue>1</contextValue>
     </contextAttribute>
     <contextAttribute>
       <name>Memory Swap</name>
       <type>String</type>
       <contextValue>100</contextValue>
     </contextAttribute>
     <contextAttribute>
       <name>RAM Memory</name>
       <type>String</type>
```

```
          <contextValue>28</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>Total Proccess</name>
          <type>String</type>
          <contextValue>61</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>VHDD Free space</name>
          <type>String</type>
          <contextValue>15826</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>Zombie Processes</name>
          <type>String</type>
          <contextValue>0</contextValue>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
    <statusCode>
      <code>200</code>
      <reasonPhrase>OK</reasonPhrase>
    </statusCode>
  </contextElementResponse>
```


2.3.9 Installation Manual

This section provides step-by-step guidelines for the installation of the required modules and components in order to setup the monitoring of the virtual servers of a particular node in the XIFI federation.

In particular, these manuals are foreseen for:

- **DEM active adapter**
  - Installation of Nagios Core on OpenStack controller (or individual VM)
  - Installation of Nagios nrpe and plugins in VMs to be monitored
  - Installation of Auto-registration service
  - Installation of DEM parsers in NGSI adapter
- **DEM passive adapter**
  - Installation of DEM in VMs to be monitored

2.3.9.1 Requirements

Both DEM versions require the latest version of Java to be installed in the operating system.

2.3.9.2 Software Repository

The source code for installing active and passive DEM adapters can be downloaded from the following repository: https://xifisvn.res.eng.it/wp3/software/DEM_Adapter/

2.3.9.3 Setup Guidelines

Installation Nagios Core (active DEM)

This section describes the steps to install Nagios on the Monitoring node. In our case, the monitoring node is considered to be the controller node of the OpenStack Cloud but any other PC can be used. These steps require administrator privileges.

Prerequisites

- Ubuntu 12.04 LTS is already installed.
- OpenStack (Grizzly) is already installed.

Installation steps:

- Requirements

  The server hosting monitoring services (monitoring host) must include the following packages: Apache 2, PHP, GCC compiler and development libraries, GD development libraries.

```
sudo apt-get install apache2
sudo apt-get install libapache2-mod-php5
sudo apt-get install build-essential
sudo apt-get install libgd2-xpm-dev sudo
apt-get install libssl-dev
```

- Create Nagios Account:

```
sudo -s
/usr/sbin/useradd -m -s /bin/bash nagios
passwd nagios
```

  Create a new *nagcmd* group for allowing external commands to be submitted through the web interface. Add both the Nagios user and the apache user to the group.

```
/usr/sbin/groupadd nagcmd
/usr/sbin/usermod -a -G nagcmd nagios
/usr/sbin/usermod -a -G nagcmd www-data
```

- Download Nagios Packages:

```
mkdir ~/downloads
cd ~/downloads
wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-
3.4.1.tar.gz
wget https://www.nagios-plugins.org/download/nagios-plugins-
1.4.16.tar.gz
wget http://sourceforge.net/projects/nagios/files/nrpe-2.x/nrpe-
```

```
2.15/nrpe-2.15.tar.gz

cd ~/downloads
tar xzf nagios-3.4.1.tar.gz
tar xzf nagios-plugins-1.4.16.tar.gz
tar xzf nrpe-2.15.tar.gz
```

- Nagios Core installation

  Compile and installation of *nagios-3.4.1* package:

```
cd nagios
./configure --with-command-group=nagcmd
make all
make install
make install-init
make install-config
make install-commandmode
```

  Configure Nagios Web interface. Install the Nagios web *config* file in the Apache *conf.d* directory:

```
make install-webconf
```

  Create a *nagiosadmin* account for logging into the Nagios Web interface. Remember the password you assign to this account – you will need it later.

```
htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

  Restart Apache to make the new settings take effect:

```
/etc/init.d/apache2 reload
```

- Nagios plugins installation:

```
cd ~/downloads/nagios-plugins-1.4.16
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
make install
```

- Start Nagios

  Configure Nagios to automatically start when the system boots:

```
ln -s /etc/init.d/nagios /etc/rcS.d/S99nagios
```

Verify the sample Nagios configuration files:

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

If there are no errors, start Nagios:

```
/etc/init.d/nagios start
```

- Login to the Web Interface

  You should now be able to access the Nagios Web interface at the URL below. You will be prompted for the username (*nagiosadmin*) and password you specified above.

```
http://localhost/nagios/
```

- Install *check_nrpe* plugin:

```
cd ~/downloads/nrpe-2.15.tar.gz
./configure
make all
make install-plugin
```

Note: If configuration process cannot find ssl libs

```
checking for type of socket size... size_t
checking for SSL headers... SSL headers found in /usr
checking for SSL libraries... configure: error: Cannot find ssl
libraries
```

They shall be found manually:

```
apt-file search libssl | grep libssl-dev
libssl-dev: /usr/lib/x86_64-linux-gnu/libssl.a
libssl-dev: /usr/lib/x86_64-linux-gnu/libssl.so
libssl-dev: /usr/lib/x86_64-linux-gnu/pkgconfig/libssl.pc
libssl-dev: /usr/share/doc/libssl-dev/changelog.Debian.gz
libssl-dev: /usr/share/doc/libssl-dev/changelog.gz
libssl-dev: /usr/share/doc/libssl-dev/copyright
```

Reconfigure the plugin and continue with the compilation/installation:

```
./configure --with-ssl=/usr/bin/openssl --with-ssl-
lib=/usr/lib/x86_64-linux-gnu<br />
```

- Create a Nagios Command definition for using the *check_nrpe* plugin

In order Nagios Core to use the *check_nrpe* plugin a command definition must be created in Nagios object configuration files. To do so, open commands.cfg file and add the following definition: *(/usr/local/nagios/etc/objects/commands.cfg)*

```
define command {
command_name  check_nrpe
command_line  $USER1$/check_nrpe -H $HOSTADDRESS$ -C $ARG1$
}
```

- Create host and service definitions

At this point, we must define the remote host and its services to Nagios Core. It is strongly recommended that these definitions are to be placed in a separate configuration file per host. Create a new file (e.g. *miniUbuntu-0.cfg*) with the following content:

```
define host{
   use                        linux-server
   host_name                  miniUbuntu-0
   alias                      Ubuntu 12.04
   address                    192.168.1.244
   }

define service{
   use                        generic-service
   host_name                  miniUbuntu-0
   service_description        CPU Load
   check_command              check_nrpe!check_load
   }

define service{
   use                        generic-service
   host_name                  miniUbuntu-0
   service_description                Current Users
   check_command                      check_nrpe!check_users
   }

define service{
   use                        generic-service
   host_name                  miniUbuntu-0
   service_description                VHDD Free space
   check_command                      check_nrpe!check_hda1
   }

define service{
   use                        generic-service
   host_name                  miniUbuntu-0
   service_description                Total Proccess
   check_command                      check_nrpe!check_total_procs
   }

define service{
   use                        generic-service
   host_name                  miniUbuntu-0
   service_description                Zombie Processes
   check_command                      check_nrpe!check_zombie_procs
   }
```

```
define service{
        use                     generic-service
        host_name               miniUbuntu-0
        service_description RAM Memory
        check_command                   check_nrpe!check_memory
        }

define service{
        use                             generic-service
        host_name                       miniUbuntu-0
        service_description             Memory Swap
        check_command                   check_nrpe!check_swap
        }
```

Next, in */usr/local/nagios/etc/nagios.cfg* file add the following line:

```
cfg_file=/usr/local/nagios/etc/objects/miniUbuntu-0.cfg
```

and proceed with verification of Nagios Configuration file:

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

If there is no error, we can restart Nagios:

```
service nagios restart
```

At this point we are not able to get any measurements from any remote host because we have not installed anything there, but we must be able to see remote host's name and services from Nagios Web interface.

Installation Nagios nrpe and plugins

This section describes the steps to install Nagios on a remote host. In our case, as remote host is considered a VM in a compute node of the OpenStack Cloud.

<u>Prerequisites</u>

- Ubuntu 12.04 LTS is already installed in a VM.
- OpenStack (Grizzly, compute) is already installed on Compute node.

<u>Installation steps:</u>

- Requirements

  In the remote host, the following packages must have been already installed: gcc, make, libssl-dev.

- Create Nagios Account:

```
sudo -i
/usr/sbin/useradd nagios
passwd nagios
```

- Download Nagios Packages:

```
mkdir ~/downloads

cd ~/downloads
wget https://www.nagios-plugins.org/download/nagios-plugins-
1.4.16.tar.gz
wget http://sourceforge.net/projects/nagios/files/nrpe-2.x/nrpe-
2.15/nrpe-2.15.tar.gz

cd ~/downloads
tar xzf nagios-plugins-1.4.16.tar.gz
tar xzf nrpe-2.15.tar.gz
```

- Install Nagios plugins:

```
cd ~/downloads/nagios-plugins-1.4.16
./configure
make
make install
```

Set right permissions to plugin directory:

```
chown nagios.nagios /usr/local/nagios
chown -R nagios.nagios /usr/local/nagios/libexec
```

- Install xinetd

If xinetd is not already installed with Ubuntu:

```
apt-get install xinetd
```

- Install nrpe daemon:

```
cd ~/downloads/nrpe-2.15
./configure
make all
make install-plugin
make install-daemon
make install-daemon-config
```

Install the NRPE daemon as a service under xinetd:

```
make install-xinetd
```

Set Monitoring Host IP address (or where Nagios Core is installed) in */etc/xinetd.d/nrpe* file (apart from loopback address, you can add as many addresses related to the Nagios Core installations, if many, accessing the nrpe) only_from - 127.0.0.1 <nagiosCore_ip_address> Add the following entry for the NRPE daemon to the */etc/services* file:

```
nrpe        5666/tcp        # NRPE
```

Restart xinetd service:

```
service xinetd restart
```

- Test NRPE daemon Locally

First, we check if nrpe daemon is started:

```
netstat -at | grep nrpe
tcp           0      0*:nrpe       *:*           LISTEN
```

and then if it works properly:

```
/usr/local/nagios/libexec/check_nrpe -H localhost
NRPE v2.15
```

- Customize NRPE commands

Make sure that all services are defined in */usr/local/nagios/etc/nrpe.cfg* file:

```
command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c
  10
command[check_load]=/usr/local/nagios/libexec/check_load -w
  15,10,5 -c 30,25,20
command[check_hda1]=/usr/local/nagios/libexec/check_disk -w 20% -c
  10% -p /dev/vda1
command[check_zombie_procs]=/usr/local/nagios/libexec/check_procs
  -w 5 -c 10 -s Z
command[check_total_procs]=/usr/local/nagios/libexec/check_procs -
  w 150 -c 200
command[check_memory]=/usr/local/nagios/libexec/check_memory.sh -w
  70 -c 90
command[check_swap]=/usr/local/nagios/libexec/check_swap -w 20% -c
  10%
```

Add a new file *check_memory.sh* in */usr/local/nagios/libexec* with the following context:

```
#!/bin/bash
#
# Script to check memory usage on Linux. Ignores memory used by
  disk cache.
#
# Requires the bc command
#
print_help() {
    echo 'Usage:'
    echo '[-w] Warning level as a percentage'
    echo '[-c] Critical level as a percentage'
    exit 0
}

while test -n '$1'; do
    case '$1' in
        --help|-h)
            print_help
            exit 0
            ;;
        -w)
            warn_level=$2
            shift
            ;;
        -c)
            critical_level=$2
            shift
            ;;
        *)
            echo 'Unknown Argument: $1'
            print_help
            exit 3
            ;;
    esac
    shift
done

if [ '$warn_level' == '' ]; then
    echo 'No Warning Level Specified'
    print_help
    exit 3;
fi

if [ '$critical_level' == '' ]; then
    echo 'No Critical Level Specified'
    print_help
    exit 3;
fi

free=`free -m | grep 'Mem:' | awk '{print $4}'`
used=` free -m | grep 'Mem:' | awk '{print $3}'`

total=$(($free+$used))

result=$(echo '$used / $total * 100' |bc -l|cut -c -4)
result=$(echo '$result' | awk '{printf('%d\n',$1 + 0.5)}')

if [ '$result' -lt '$warn_level' ]; then
```

```
    echo 'RAM Memory OK. $result% used.'
    exit 0;
elif [ '$result' -ge '$warn_level' ] &&
     [ '$result' -le '$critical_level' ]; then
    echo 'RAM Memory WARNING. $result% used.'
    exit 1;
elif [ '$result' -gt '$critical_level' ]; then
    echo 'RAM Memory CRITICAL. $result% used.'
    exit 2;
fi
```

Make *check_memory.sh* executable:

```
chmod 777 check_memory.sh
```

- Check nrpe from monitoring host

  Now, everything is setup and we can check the nrpe daemon from the monitoring host side, by executing the following command (make sure that local firewall in remote host allows connections on port 5666):

```
/usr/local/nagios/libexec/check_nrpe -H <Remote_Host_ip> -c
check_users
```

  The response must be something like that:

```
USERS OK - 2 users currently logged in |users=2;5;10;0
```

- Access remote host from Nagios Web Interface.

  Installation is complete and you can access Nagios services from Web Interface in monitoring host in the URL *http://<monitoring_host_ip>/nagios*.

Installation DEM Adapter

Here it is described the steps to install DEM Adapter on the same server where Nagios core is installed.

<u>Prerequisites</u>

The following packages must have been already installed: g++, make, libc6-dev, libstdc++6-dev, java7, Nagios Core.

<u>Installation steps:</u>

- Install Unix Domain Sockets for Java (junixsocket)

  Download junixsocket tarball from https://code.google.com/p/junixsocket/downloads

Extract and copy the following files in */opt/newsclub/lib-native*:

```
libjunixsocket-linux-1.5-amd64.so Linux Intel 64bit
libjunixsocket-linux-1.5-i386.so Linux Intel 32bit
```

- Download and Install livestatus plugin:

```
cd ~/downloads
wget 'http://www.mathias-kettner.de/download/mk-livestatus-
1.2.2p2.tar.gz'

tar xzf mk-livestatus-1.2.2p2.tar.
cd mk-livestatus-1.2.2p2
./configure
make
make install
```

Create path for UNIX socket and change mode:

```
mkdir -p /var/lib/nagios/rw
chmod 777 /var/lib/nagios/rw -R
```

Add the following lines in nagios.cfg in order to configure Nagios to operate with livestatus broker:

```
broker_module=/usr/local/lib/mk-livestatus/livestatus.o
var/lib/nagios/rw/live event_broker_options=-1
```

Restart Nagios to activate the changes:

```
service nagios restart
```

Verify livestatus installation:

- Open Nagios log file. You must read:

```
[1256144866] livestatus: Version 1.1.6p1 initializing. Socket
path: '/var/lib
/nagios/rw/live'
[1256144866] livestatus: Created UNIX control socket at
/var/lib/nagios/rw/
live
[1256144866] livestatus: Opened UNIX socket
/var/lib/nagios/rw/live
[1256144866] livestatus: successfully finished initialization
[1256144866] Event broker module '/usr/local/lib/mk-
livestatus/livestatus.o' initialized successfully.
[1256144866] Finished daemonizing... (New PID=5363)
[1256144866] livestatus: Starting 10 client threads
```

```
[1256144866] livestatus: Entering main loop, listening on UNIX
socket
```

- You must be able to get host list from Nagios using unixcat:

```
echo 'GET hosts' | unixcat /var/lib/nagios/rw/live
```

- Install Nagios NGSI Adapter

Download the latest version of NGSI Adapter from www.synelixis.com/xifi.

```
Username: xifi

Password:xifi#2013#
```

Or from https://xifisvn.res.eng.it/wp3/software/DEM_Adapter/

Extract *nagiosNGSi* directory and copy it to */usr/local/nagios*.

Open configuration file */usr/local/nagios/nagiosNGSi/nagiosNGSi.cfg* and set context brokers *url*, *polling time interval* and *regionid*:

```
polling_interval=5000
conUpdate_url=http://192.168.1.242:1026/NGSI10/updateContext
conRegistration_url=http://192.168.1.242:1026/NGSI9/registerContext
regionid={your regionid}
regionid should be one of the following strings according to the
location of the infrastructure: Trento, Berlin, Brittany, Sevilla,
Waterford
```

Start NGSI Adapter:

```
./nagiosNGSi.sh start
```

- Check log file for errors:

```
tail -f log/nagiosNGSi.logs
```

Installation Orion

- Install the following packages (requirements for the given Orion RPMs) on a CentOS VM:

```
# yum update && yum install boost-filesystem boost-thread
libmicrohttpd-devel python python-flask curl libxml2 nc mongodb
```

- Install the Orion RPMs from https://forge.fi-ware.eu/frs/?group_id=7

- When the RPMs are installed, the mongo server is not installed (the mongo binary installed refers to the mongo interactive shell). Therefore, the following command should be issued:

```
# yum install mongodb-server python-pymongo
```

- The Context Broker assumes that a database called 'orion' exists in mongodb, as well as an administrator user, also called 'orion', with password 'orion'. The db actually exists, but the user does not. Therefore, we should create it:

```
# mongo
> use orion
> db.addUser( { user: 'orion', pwd: 'orion', roles: [ 'readWrite',
'dbAdmin' ] } )
> exit
#
```

- Since iptables is blocking access to port 1026 (default for contextBroker), we should open it through the iptables conf file */etc/sysconfig/iptables*

  Add the line -A INPUT -m state --state NEW -m tcp -p tcp --dport 1026 -j ACCEPT in iptables conf file, right below the line enabling port 22 (-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT)

  Restart iptables:

```
# service iptables restart
```

Installation manual (passive DEM on Remote Host -VM Ubuntu 12.04)

- Requirements

  Prior to this installation java 7 must be installed and OpenStack metadata API service must be enabled.

- Java installation

```
sudo -i
apt-get update
apt-get install openjdk-7-jdk
```

- Check open stack metadata API.

```
curl http://169.254.169.254/openstack/latest/meta_data.json
```

Output:


```
{"random_seed":"qmsKI/YXTisafsEEueV8J4HZdKPu0yXH74G8siSpkZclTpFFnFI
4SxoZWUhwEYMOrNHqm8x9KH9rKTbSxYPvCsuWBQevg5hDzL8lrrtmthO7IF1ZxRadCC
888pCt3ywqiuMKTBhEZZKWWsrfZPpO+WSb5spIKbqDhRgCDQ0RcJmKI3iYGYk5p0gwE
SimB4yBPMwJ1qNL1mBLAJMIRmh/Upx+Y6F0aW6k+UVDursHXiMHgtkLxXc/ui8ZOXeI
KlpifZ2Ak3yR9Odzb+tYHZYSLSAiMGtygdoTjPM+jYBHIPv6VIYyHxq6KIk2k+bHIZi
FomyCez8fYZEzlrOpUbDfiLSULzUD056ugdgk4pcCxORvGp6WGciu5lY8MS847SWTf/
wcs29au8hls30iBaJdhOy1fOcockLS/pYSQKZvRA89CnrhV5bs9X4erNGkjkX/M9eBF
YIxgE53ulOh/bNlLcF2Smu4NS81b0QC4ZBLlGtenp4dr+O8nX/B3DBSO0ezHkYMCNM+
STy5otevgYauLvI1RVQewMdJG8rx1ZlYVT0sDSuV9feLy30/uH1CG9iFQi0s/fNTi2p
byV538BFKuI3NRoiR3BoCgaX6SjdCASoJU1bf90D1602eg501mOigcrotoIBIvvXRWR
aFQv1IBQDMBu82NgDnXltJ/yNLbPo=",
   "uuid":"8cc96345-ac14-498a-8e8d-62cf2c474aa0",
   "availability_zone":"nova",
   "hostname":"myVM.novalocal",
   "launch_index":0,
   "meta":{"region":"RegionOne"},
   "public_keys":{
   "cri_keypair":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDR4PMadpOYGnpYMf3HnXqsJmJt2o5q685aftL
l//EvhCi/e+BeqWtcwCQ1dUcx91148u0DRpYsdg3n4Qab3/6eBJOFQFDknn+JhGjU8w
zsYtVzaFvUnCn1EA04GjzodDO+z8sAeCDiwqhlLMI7gvCA+cyIX0mYwVV87M/m7f51Z
//QT0M72z9JZDlc7j1lWSrf2BsbXln/BFj39PCtRxEbpZcwqaPldDX5Aq5/Dwdrn/G4
XsBlgI8vApDG7HRDiV5l/O5UWbqJHtDo5q+TuacBnI9UcEg6TYVmI1kU6gkPV5POp53
Lqy+q710y3NK0e2vQvDJKnsh7yzbdse+yUQeV Generated by Nova\n"},
   "name":"myVM"
}
```


- Installation Procedure

  - Download the s/w from xifi svn

```
wget
https://xifisvn.res.eng.it/wp3/software/DEM_Adapter/PDEMv1.0/Ub12.0
4_vmMonitor_v1.0.tar.gz
```

- Install the s/w

```
tar -zxvf Ub12.04_vmMonitor_v1.0.tar.gz
cd Ub12.04_vmMonitor_v1.0
./configure.sh
```

Output:

```
INSTALL_PATH is set to '/usr/local'
/usr/local/vmMonitor
Coping files..
Configuration Completed.
```

- Configuration

```
nano cd /usr/local/vmMonitor/monitoring.cfg
```

Set properly the following attributes:

- Context broker url:

```
cb_server = http://192.168.1.245:1026 (modify it by the Orion
Context Broker ip address)
```

- Update time interval in seconds:

```
time_int = 30
```

- Location of the application's logfile:

```
logfile = log/monitoring.log
```

- Openstack's metadata API url:

```
metadata_url =
http://169.254.169.254/openstack/latest/meta_data.json
```

- Enable Service
  - Start Service

```
service vmonitorS start
```

Output:

```
Starting NG Service Update ...
Update service started!
```

- Check log file

```
tail -f /usr/local/vmMonitor/log/monitoring.log
```

Output:

```
Aug 25, 2014 12:29:04 PM com.synelixis.xifi.Logger.LogFile add
INFO: Passive DEM (2014-08-25 12:29:04.899): Monitoring service
started...
Aug 25, 2014 12:29:07 PM com.synelixis.xifi.Logger.LogFile add
INFO: Passive DEM (2014-08-25 12:29:07.354):
{"contextElements":[{"type":"vm","isPattern":"false","id":"RegionOn
e:8cc96345-ac14-498a-8e8d-62cf2c474aa0","attributes":
[{"name":"_timestamp","type":"long","value":"1408969747353"},{"name
":"users","type":"string","value":"1"},{"name":"cpuLoadPct","type":
"string","value":"0.00"},{"name":"freeSpacePct","type":"string","va
lue":"95"},{"name":"z_procs","type":"string","value":"0"},{"name":"
procs","type":"string","value":"71"},{"name":"usedMemPct","type":"s
tring","value":"5"},{"name":"swap","type":"string","value":"100"}]}
],"updateAction": "APPEND"}
Aug 25, 2014 12:29:07 PM com.synelixis.xifi.Logger.LogFile add
INFO: Passive DEM (2014-08-25 12:29:07.403): POST RESP:200
...
```

2.3.10 User Manual

2.3.10.1 API Specification

DEM adapter provides the methods described in NGSI-9/10 specifications ([8], [7]), to be compliant with the Orion Context Broker.

2.3.10.2 Handbook

The primary objective of DEM Active Adapter is to retrieve monitoring data from Nagios, adapt them to NGSI-9/10 API and post them to a Context Broker. In each execution cycle, DEM Adapter performs the following steps:

- Collects data from Nagios Core, regarding all monitored hosts and their respective services.
- Processes information per node and per service.
- Builds a ContextUpdate xml for each host, based in NGSI 10 API.
- Posts the ContextUpdate xmls to Context Broker

This document guides the user to configure and properly set up the DEM active Adapter.

System Configuration

DEM configuration procedure is organized in two main parts. First, we must define DEM settings for the communication between Nagios monitoring system and Context Broker. Second, we describe the configuration procedure in order to add/remove nodes or monitored services.

First, we must define DEM settings for the communication between Nagios monitoring system and Context Broker.

- Service Settings

All DEM's service settings are defined in nagiosNGSi.cfg file. These settings are:

  - Nagios Polling interval in milliseconds: defines how frequently DEM service asks Nagios for update.
  - Context Broker ContextUpdate URL: defines the CB update URL based on NGSI-10 API.
  - Context Broker Registration URL: defines the CB registration URL based on NGSI-9 interface.
  - Nagios Event Broker: defines the path of unixsocket that is used by DEM in order to communicate with Nagios.
  - Log File: defines the filename and the location of DEM log file.
  - Region id: defines the geographical region of the federated infrastructure.
  - Debug: if true, log files are kept.

```
polling_interval=5000
conUpdate_url=http://192.168.1.242:1026/NGSI10/updateContext
conRegistration_url=http://192.168.1.242:1026/NGSI9/registerContext
unixSocket=/var/lib/nagios/rw/live
logfile=log/nagiosNGSi.log
regionid=Trento
debug=true
```

- Add new host

In order to add new hosts (i.e. VMs) we need to create a new object definition per each new host. It is strongly recommended that these definitions are placed in a separate file per new node. Assuming that we need to add a new node (ubu-0 with IP address 192.168.1.244), create a file (*/usr/local/nagios/etc/ubu-0.cfg*) with the following:

```
define host{
use            linux-box
host_name      ubu-0
alias Ubuntu 12.04LTS
address        192.168.1.244
}
```

Next, we must also define monitoring data (services) that are going to be collected by the new host. We assume that we have already installed *check_nrpe*, plugins and *nrpe* addon packages in ubu-0. From all the services that plugins package offers, we choose to retrieve the number of total process, CPU usage and free hard disk space. So, we proceed by defining these services in ubu-0.cfg by adding the following lines:

```
define service{
use generic-service
host_name ubu-0
service_description CPU Load
check_command check_nrpe!check_load
}
define service{
use generic-service
host_name ubu-0
service_description /dev/hda1 Free Space
check_command check_nrpe!check_hda1
}
define service{
use generic-service
host_name ubu-0
service_description Total Processes
check_command check_nrpe!check_total_procs
}
```

Finally, in order to inform Nagios Core for the introduction of a new host and services, we add the following lines to *nagios.cfg* file:

```
# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/ubu-0.cfg
```

Verify your Nagios configuration files:

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

If everything is ok, restart Nagios:

```
service nagios restart
```

You should see the host created in the Nagios web interface.

- Add new monitoring service

In this section we define and test the implementation of a new, custom monitoring metric (service), e.g. RAM memory usage.

In remote host, create a new file named check_memory.sh in */usr/local/nagios/libexec* with the following context:

```
#!/bin/bash
#
# Script to check memory usage on Linux. Ignores memory used by
disk cache.
#
# Requires the bc command
#
print_help() {
  echo 'Usage:'
  echo '[-w] Warning level as a percentage'
  echo '[-c] Critical level as a percentage'
  exit 0
}
while test -n '$1'; do
  case '$1' in
  --help|-h)
  print_help
  exit 0
  ;;
  -w)
  warn_level=$2
  shift
  ;;
  -c)
  critical_level=$2
  shift
  ;;
  *)
  echo 'Unknown Argument: $1'
  print_help
  exit 3
  ;;
  esac
  shift
done

if [ '$warn_level' == '' ]; then
  echo 'No Warning Level Specified'
  print_help
  exit 3;#i
if [ '$critical_level' == '' ]; then
  echo 'No Critical Level Specified'
  print_help
  exit 3;
fi
free=`free -m | grep 'Mem:' | awk '{print $4}'`
used=` free -m | grep 'Mem:' | awk '{print $3}'`
total=$(($free+$used))
result=$(echo '$used / $total * 100' |bc -l|cut -c -4)
result=$(echo '$result' | awk '{printf('%d\n',$1 + 0.5)}')
```

```
if [ '$result' -lt '$warn_level' ]; then
  echo 'RAM Memory OK. $result% used.'
  exit 0;
elif [ '$result' -ge '$warn_level' ] &&
  [ '$result' -le '$critical_level' ]; then
  echo 'RAM Memory WARNING. $result% used.'
  exit 1;
elif [ '$result' -gt '$critical_level' ]; then
  echo 'RAM Memory CRITICAL. $result% used.'
  exit 2;
fi
```

Make check_memory.sh executable:

```
root@192.168.1.244:~# chmod 777 check_memory.sh
```

Test metric locally:

```
root@192.168.1.244:~# /usr/local/nagios/libexec/check_memory.sh -w
70 -c 90
RAM Memory OK. 29% used. (18921)
```

In monitoring node, define new metric in nrpe configuration file. Open */usr/local/nagios/etc/nrpe.cfg* and add the following line:

```
command[check_memory]=/usr/local/nagios/libexec/check_memory.sh -w
70 -c 90
```

Test metric remotely:

```
root@192.168.1.231:~# /usr/local/nagios/libexec/check_nrpe -H
192.168.1.244 -c check_memory
RAM Memory OK. 29% used. (18500)
```

Enable active DEM Service

After configuration procedure is completed, we are ready to start DEM service:

```
root@192.168.1.231:/usr/local/nagios/nagiosNGSi# ./nagiosNGSi.sh
start
Starting nagiosNGSi ...
nagiosNGSi started …
```

In order to be sure that everything is working properly we open and check /usr/local/nagio/nagiosNGSi/log/nagiosNGSi.log for errors. If all gone well you must see the following output:

```
INFO: GET Services...
Dec 10, 2013 10:51:43 AM nagiosngsi.XML_File buildXMLupdate
INFO: File update_192.168.1.65.xml build completed.
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Post: 0 Web Response status code: 200
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Orion's Response: 200 OK
Dec 10, 2013 10:51:43 AM nagiosngsi.XML_File buildXMLupdate
INFO: File update_127.0.0.1.xml build completed.
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Post: 1 Web Response status code: 200
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Orion's Response: 200 OK
Dec 10, 2013 10:51:43 AM nagiosngsi.XML_File buildXMLupdate
INFO: File update_192.168.1.244.xml build completed.
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Post: 2 Web Response status code: 200
Dec 10, 2013 10:51:43 AM nagiosngsi.WebClient postXML
INFO: Orion's Response: 200 OK
```

Context Broker

- Registration Status

Get registered services for host 192.168.1.244:

```
root@192.168.1.231:#curl
192.168.1.242:1026/ngsi9/contextEntities/192.168.1.244 -s -S --
header 'Content-Type: application/xml' | xmllint --format -
<?xml version='1.0'?>
<discoverContextAvailabilityResponse>
  <contextRegistrationResponseList>
<contextRegistrationResponse>
 <contextRegistration>
    <entityIdList>
      <entityId type='VRTMachine' isPattern='false'>
       <id>192.168.1.244</id>
      </entityId>
   </entityIdList>
        <contextRegistrationAttributeList>
      <contextRegistrationAttribute>
        <name>Region ID</name>
        <type>String</type>
            <isDomain>false</isDomain>
      </contextRegistrationAttribute>
      <contextRegistrationAttribute>
        <name>CPU Load</name>
        <type>String</type>
            <isDomain>false</isDomain>
      </contextRegistrationAttribute>
      <contextRegistrationAttribute>
        <name>Current Users</name>
        <type>String</type>
            <isDomain>false</isDomain>
     </contextRegistrationAttribute>
      <contextRegistrationAttribute>
```

```
          <name>Memory Swap</name>
          <type>String</type>
              <isDomain>false</isDomain>
        </contextRegistrationAttribute>
        <contextRegistrationAttribute>
          <name>RAM Memory</name>
          <type>String</type>
              <isDomain>false</isDomain>
        </contextRegistrationAttribute>
        <contextRegistrationAttribute>
          <name>Total Proccess</name>
          <type>String</type>
              <isDomain>false</isDomain>
        </contextRegistrationAttribute>
        <contextRegistrationAttribute>
          <name>VHDD Free space</name>
          <type>String</type>
              <isDomain>false</isDomain>
        </contextRegistrationAttribute>
       <contextRegistrationAttribute>
          <name>Zombie Processes</name>
          <type>String</type>
              <isDomain>false</isDomain>
        </contextRegistrationAttribute>
      </contextRegistrationAttributeList>
<providingApplication>http://randomURL.com/VRTMachine</providingAp
plication>
  </contextRegistration>
</contextRegistrationResponse>
  </contextRegistrationResponseList>
</discoverContextAvailabilityResponse>
```

- Update Status

Get last update values of 192.168.1.244 services:

```
root@192.168.1.231:# curl
192.168.1.242:1026/ngsi10/contextEntities/192.168.1.244 -s -S --
header 'Content-Type: application/xml' | xmllint --format -<?xml
version='1.0'?>
<contextElementResponse>
  <contextElement>
<entityId type='' isPattern='false'>
 <id>192.168.1.244</id>
</entityId>
<contextAttributeList>
  <contextAttribute>
    <name>Region ID</name>
    <type>String</type>
        <contextValue>Trento</contextValue>
  </contextAttribute>
  <contextAttribute>
    <name>CPU Load</name>
    <type>String</type>
        <contextValue>0.00</contextValue>
  </contextAttribute>
  <contextAttribute>
    <name>Current Users</name>
```

```
      <type>String</type>
          <contextValue>1</contextValue>
    </contextAttribute>
    <contextAttribute>
      <name>Memory Swap</name>
      <type>String</type>
          <contextValue>100</contextValue>
    </contextAttribute>
    <contextAttribute>
      <name>RAM Memory</name>
      <type>String</type>
          <contextValue>28</contextValue>
    </contextAttribute>
    <contextAttribute>
      <name>Total Proccess</name>
      <type>String</type>
          <contextValue>61</contextValue>
    </contextAttribute>
    <contextAttribute>
      <name>VHDD Free space</name>
      <type>String</type>
          <contextValue>15826</contextValue>
    </contextAttribute>
    <contextAttribute>
      <name>Zombie Processes</name>
      <type>String</type>
          <contextValue>0</contextValue>
    </contextAttribute>
  </contextAttributeList>
    </contextElement>
    <statusCode>
  <code>200</code>
  <reasonPhrase>OK</reasonPhrase>
    </statusCode>
  </contextElementResponse>
```


2.4 OpenStack Data Collector-ODC

2.4.1 Summary

OpenStack Data Collector aims at collecting information from an OpenStack installation. Information collected is capacity data as number of virtual machine deployed, number of cores available, size of RAM and size of disk, number of users/tenders registered. Information is collected using the OpenStack keystone and nova APIs and/or the command line: unfortunately not all the data requested is available through an API and for this reason the command line should be used.

As depicted in Figure 20, the context of this component is highlighted: OpenStack Data Collector itself is part of the yellow block (*Monitoring Adapter & Collector*).

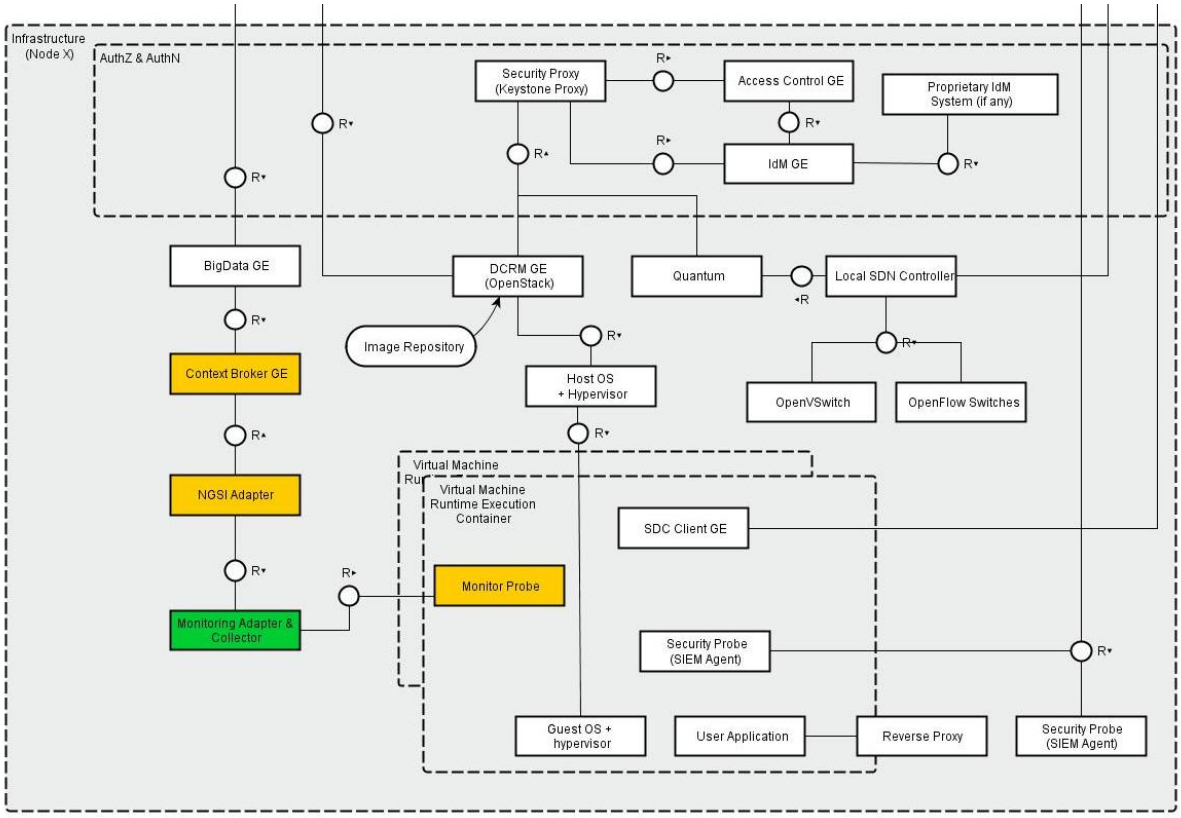

Figure 20: OpenStack Data Collector within the Architecture of a XIFI Node

Tables below provide specific details with regard to the component:

Reference Scenarios	UC-5 - Network and Data Centre operations
Reference Stakeholders	This component aims to provide capacity information useful to Infrastructure Owners and FI Developers
Type of ownership	New component
Original tool	None
Planned OS license	Apache License Version 2.0
Reference OS community	None at the moment

Table 17: OpenStack Data Collector Context Details

Consist of	• OpenStackDataCollector: component collecting OpenStack information from OpenStack APIs or command line
Depends on	• NGSI Adapter • OpenStack Grizzly version [22]

Table 18: OpenStack Data Collector Dependencies Summary

2.4.2 Component Responsible

Developer	Contact	Partner
Attilio Broglio	attilio.broglio@create-net.org	CREATE-NET
Silvio Cretti	silvio.cretti@create-net.org	CREATE-NET

Table 19: OpenStack Data Collector Reference Details

2.4.3 Motivation

As said, this component aims at collecting capacity information from OpenStack installation like number of cores installed, size of RAM and disk, number of virtual machines deployed, number of users/tenant registered. This component is needed in the XIFI landscape in order to provide capacity data that otherwise is not gathered from the other monitoring adapters components and on the other hand needed in order to monitor the capacity of the federation. This information is showed on the *Infographics and Status Pages Component* (D4.4 [31]) using infographics representation.

2.4.4 User Stories Backlog

Id	User Story Name	Actors	Description
1	Collect info on capacity size available on a given region	Infographics and Status Pages Component (see D4.4)	Capacity data as number of cores, amount of RAM, amount of DISK, number of users, number of VM should be collected from OpenStack (either APIs if possible or command line).
2	Get details about VMs	Deployment and Configuration Adapter (see D3.3 [30])	List of available VM images and list of active VM should be collected from OpenStack.

Table 20: OpenStack Data Collector User Stories Backlog

2.4.5 State of the art

At the moment we are not aware of any "off-the-shelf" component that can provide the functionality of OpenStack Data Collector. A possible alternative could be to use OpenStack Ceilometer [21] but in the version of OpenStack currently used in XIFI (Grizzly) it is not in a stable and mature version.

2.4.6 Architecture

Figure 21 provides the design model of the OpenStack Data Collector component. The OpenStack Data Collector gathers capacity information on the resources available from OpenStack APIs or OpenStack command line and sends them to the NGSI Adapter (described in Section 2.1).

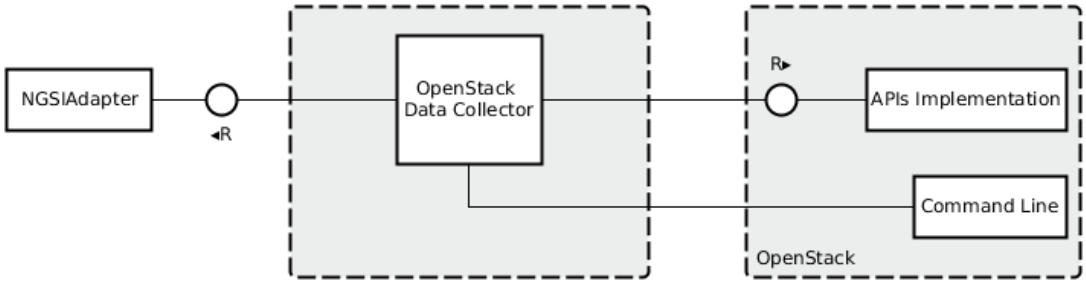

Figure 21: OpenStack Data Collector Architecture

2.4.7 Release Plan

Version ID	Milestone	User Stories
1.0	M9	1
2.0	M18	2

Table 21: OpenStack Data Collector Release Plan

2.4.8 Test Cases

- **Prerequisites**
  - Install OpenStack Grizzly
  - Install OpenStack data collector
- **Execution.** Run OpenStack data collector

```
python openstackdatacollector.py dump
```

- **Verification.** Check the file *results.dumped* and verify if the information about #cores, #RAM, #Disk, #VM, images, VM list and timestamp is present and is the same as the one obtained through the *nova client* command line.

2.4.9 Installation Manual

2.4.9.1 Requirements

- Ubuntu 12.04 as operating system
- OpenStack Grizzly installed
- Python installed (2.6+)

2.4.9.2 Software Repository

- OpenStack Data Collector: https://xifisvn.res.eng.it/wp3/software/Openstack_Data_Collector/Trunk/OpenstackDataCollector/

2.4.9.3 Setup Guidelines

- Install NGSI Adapter (section 2.1.8) and properly configure it in order to register to the Orion Context Broker [23].
- Get the *region.js* file from https://xifisvn.res.eng.it/wp3/software/Openstack_Data_Collector/Trunk/OpenstackDataCollector/ and copy it in the $NGSI_ADAPTER/lib/parsers (where NGSI_ADAPTER is the installation directory of the NGSI Adapter).
- Get the *openstackdatacollector.py* file from https://xifisvn.res.eng.it/wp3/software/Openstack_Data_Collector/Trunk/OpenstackDataCollector/
- Copy the *openstackdatacollector.py* file on a OpenStack controller machine in a directory of your choice (suggestion */usr/local/openstackdatacollector*)
- Edit the file *openstackdatacollector.py* and customize these parameters:

```
##Configure these parameters##
username='admin'
password='put here admin password'
tenant_name='admin'
auth_url='put here authorization api url'
regionId='put here the region id'
regionName='put here the region name'
location='put here country in ISO 3166-1 alpha2 standard'
latitude='put here GPS latitude'
longitude='put here GPS longitude'
agentUrl='endpoint of the NGSI Adapter';
timeInterval=refresh time in seconds;
```

2.4.10 User Manual

2.4.10.1 API Specification

Not applicable.

2.4.10.2 Handbook

In order to run the component:

- Run NGSI Adapter
- Open a terminal on the machine where the component has been installed
- Go to the directory where the component has been installed
- Run: *python openstackdatacollector.py* (you can also consider to run it as a daemon).

2.5 Network Passive Monitoring-NPM Adapter

2.5.1 Summary

Network Passive Monitoring Adapter is the component in charge of handling performance data from network devices located within the domain of the infrastructures. By leveraging on a Nagios system, this adapter collects network-based metrics via the SNMP protocol and publishes them to the NGSI Adapter, where the specific parser will standardize the format, as illustrated in Figure 22.

This document does not provide further details with regard to the description already available in deliverable D3.2 [26]. The component has not evolved since the release of such documentation. Therefore, the reader is referred to check D3.2 to find a proper specification.

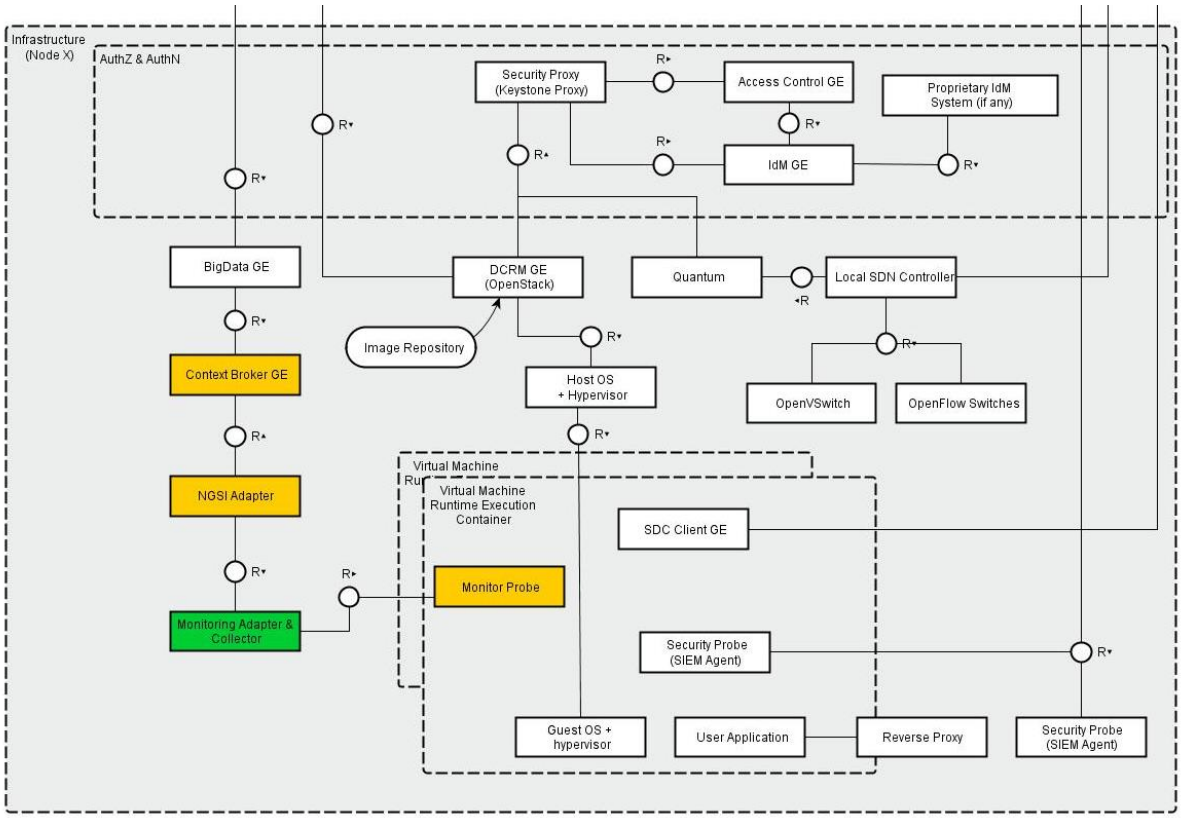

Figure 22: NPM Adapter within the Architecture of a generic XIFI Node

Tables below provide specific details with regard to the component:

Reference Scenarios	UC-5 - Network and Data Centre operations
Reference Stakeholders	• **Developers**: those who want a) to be aware of the network performance in real time for experimentation b) to install a private XIFI node • **Infrastructure owners**: those who want to join XIFI federation or build a private XIFI node
Type of ownership	Development and Extension
Original tool	• Nagios [13] • Nagira [14]
Planned OS license	--
Reference OS community	Nagios Community

Table 22: NPM Adapter Context Details

Consist of	• A script for the creation of MySQL DB: *xifi_npm_db_v1.0.txt* • A Nagira-based API (the so called *xifi-npm-DBfeed*), between Nagios and MySQL DB • A northbound REST API (*xifi-npm-restserver*)
Depends on	• NGSI Adapter (WP3) • XIFI Federation Monitoring (WP2) • XIFI Monitoring Dashboard (WP4) • OS Ubuntu 12.04 LTS Server, 64 bits • Nagios Core 3.4.1 • nagios-plugins 1.4.1 • Nagira 0.3.1 for internal APIs (between Nagios and MySQL DB) • MySQL 5.5 • RESTLET 2.1 for external APIs (of NPM Adapter) • Apache httpcomponents-client-4.3.1 • MySQL Connector/J 5.1.27

Table 23: NPM Adapter Dependencies Summary

2.5.2 Component Responsible

Developer	Contact	Partner
Giuseppe Ricucci	giuseppe.ricucci@telecomitalia.it	TI

Table 24: NPM Adapter Reference Details

3 CONCLUSIONS

This deliverable has tackled the updated specification of those monitoring adapters in charge of feeding the Federation Monitoring with infrastructure-related performance metrics, both in terms of network and computing resources.

With the deployment of this abstraction layer, XIFI infrastructure nodes are able to provide standard information concerning availability and status of those resources they intend to federate. Hence, this is an extremely important functional block, not only relevant for Infrastructure Owners. It also concerns other XIFI branches, especially those associated to high-level features. Several WP4-driven components will leverage on this data to provide enhanced services, such as the Monitoring Dashboard and the Recommendation Tool that supports developers in selecting the node that fits their needs in the best manner. Without a common approach, it would not have been possible to handle the whole picture of the status of the federation.

The outcomes enclosed in this document define the basis of the XIFI monitoring architecture, and represent the main reference of WP3's T3.2 - Infrastructure Monitoring Adapters, that comes to an end with the release of this M18 deliverable. Attending the recommendation M12.3 posed by the official reviewers at XIFI's second Review Report, which encourages the stabilisation of the architecture to foster uptake and usability, deliverable D3.5 provides the mature specification of the XIFI Infrastructure Monitoring Middleware (XIMM). Future upgrades and related work will take this documentation as starting point.

REFERENCES

[1] Apache License, Version 2.0. http://www.apache.org/licenses/LICENSE-2.0.html

[2] BWCTL. http://software.internet2.edu/bwctl/

[3] cURL. http://curl.haxx.se/

[4] FIWARE Big Data GE Architecture Description. https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Data.BigData

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

[6] FIWARE Monitoring GEi components. https://github.com/Fiware/fiware-monitoring

[7] FIWARE 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

[8] FIWARE 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

[9] FIWARE PEP OAuth Authentication Proxy. https://github.com/ging/fi-ware-pep-proxy

[10] GNU Wget. http://www.gnu.org/software/wget/

[11] Iperf. https://github.com/esnet/iperf

[12] Joyent Documentation: “Installing Node.js via package manager”. https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager

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

[14] Nagira - Nagios RESTful API. http://exchange.nagios.org/directory/Addons/APIs/Nagira--2D-Nagios-RESTful-API/details

[15] NAM Adapter’s NPM Package Manager. https://www.npmjs.org/package/nam_adapter

[16] Node.js. http://nodejs.org/

[17] NPM Package Manager. https://www.npmjs.org/

[18] OAuth 2.0 Authorization Framework. http://tools.ietf.org/html/rfc6749

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

[20] OpenNMS. http://www.opennms.org/

[21] OpenStack Ceilometer. https://wiki.openstack.org/wiki/Ceilometer

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

[23] Orion Context Broker. http://catalogue.fi-ware.org/enablers/publishsubscribe-context-broker-orion-context-broker

[24] OWAMP. http://software.internet2.edu/owamp/

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

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

[27] XIFI Deliverable D1.5 - Federated Platform Architecture v2

[28] XIFI Deliverable D2.5 - APIs and Tools for Infrastructure Federation v2

[29] XIFI Deliverable D3.2 - Infrastructures monitoring and interoperability adaptation components toolkit and API. https://bscw.fi-xifi.eu/pub/bscw.cgi/d58608/XIFI-D3.2-Infrastructures_monitoring_and_interoperability_adaptation_components_toolkit_and_API.pdf

[30] XIFI Deliverable D3.3 - Infrastructures management toolkit API. https://bscw.fi-xifi.eu/pub/bscw.cgi/d58587/XIFI-D3.3-Infrastructures_Management_Toolkit_API.pdf

[31] XIFI Deliverable D4.4 - Baseline Tools v2

[32] XIFI Federated Identity Management. http://wiki.fi-xifi.eu/Public:Federated_Identity_Management

[33] Zabbix. http://www.zabbix.com/