D4.2 BEMO interoperability and communication architecture

Author: Luca Paoletti as representative of D'Appolonia project team
Contributor (s): -
Project Acronym: S4ECoB
Grant Agreement Number: 284628

Issue Date:	**M34**
Deliverable Number:	**D4.2**
Work Package Number:	**WP4**
Status:	**Final version**

DISEMINATION LEVEL	
X	**PU** = Public
	PP = Restricted to other programme participants (including the EC)
	RE = Restricted to a group specified by the consortium (including the EC)
	CO = Confidential, only for members of the consortium (including the EC)

Document History			
Version	**Date**	**Author**	**Description**
0.1	30.11.2014	Luca Paoletti as representative of D'Appolonia project team	ToC preparation
0.2	11.12.2014	Andrea Cavallaro	Reviewed ToC

		as representative of D'Appolonia project team	
1.0	12.02.2015	Luca Paoletti as representative of D'Appolonia project team	Draft version
1.1	15.03.2015	Andrea Cavallaro as representative of D'Appolonia project team	Review of draft version and input for next development
1.2	19.03.2015	Luca Paoletti as representative of D'Appolonia project team	Updated version
1.3	23.03.2015	wkattanek (IMMS)	QCC review
2	31.03.2015	Luca Paoletti as representative of D'Appolonia project team	Final version

Disclaimer

The information proposed in this document is provided as a generically explanation on the proposed topic. No guarantee or warranty is given that the information fits for any particular purpose. The user thereof must assume the sole risk and liability of this report practical implementation. The document reflects only the author's views and the whole work is not liable for any empirical use of the information contained therein.

CONTENTS

TERMINOLOGY AND ABBREVIATIONS

BEMO	building energy management system optimizer = overall S4ECoB system
BMS	building management system
BEMO server	central component of each BEMO installation (usually 1 BEMO server per building / site); consists of several software components (e.g. data base, data fusioning module, GUI and BMS interface, gateway to occupancy sensor network, internet remote access) running on a dedicated PC / server or on PC / server which is already part of the BMS
Occupancy sensor	audio front-end + APU
Audio front-end	microphones / microphone array + ASU
Microphone / microphone array	one or up to eight (array) acoustic-to-electrical transducers
Acoustic processing unit (APU)	embedded computer for processing audio data – received from microphones via ASU – i.e. calculating occupancy levels and detecting unknown sounds and sending resulting information via network to the APU gateway
Audio satellite unit (ASU)	electronic unit for converting analogue microphone signals to digital audio signals
APU gateway	(software) component connecting the occupancy sensor network with the BEMO server; usually runs on the BEMO server
APU Gateway Interface	(software) component connecting APU gateway with BEMO server
OS	Operating System
VM	Virtual Machine
AIS	Accounting Information System
OPC	OLE (object linking and embedding) for Process Control
BACnet	Building Automation Control Network
VPN	Virtual Private Network
VNC	Virtual Network Computing
AIS Gateway	(software) component connecting AIS with BEMO server
BMS Gateway	(software) component connecting BMS with BEMO server
BEMO Control and Management Platform	a web application running on the BEMO server, accessible through internet connection and a web browser, that comprises the communication platform, the optimization unit, a messaging system and four software communication modules

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This deliverable is associated to T4.2 and intends to design an interoperable architecture to gather and dispatch information from the different units composing the S4ECoB system.

The S4ECoB system consists of two main components:

- Occupancy sensors with integrated Acoustic Processing Unit (APU) and up to 3 Audio Satellite Units (ASU) each connected with up to 8 microphones
- Building Energy Management Optimizer server (BEMO server)

The BEMO server however needs to communicate also with other units like the BMS for monitoring the environment and controlling HVACL systems already present in the pilot buildings and the AIS for obtaining accounting data for the pilot site.

For designing the communication architecture, three software modules have been developed and will be described in the present document:

1. APU Gateway Interface: developed in C, handles the communication with the Occupancy Sensor helped by the APU gateway software module.
2. AIS Gateway: developed in Java using Super CSV, handles the communication with the AIS in Linate and Principe Pio pilot site. For Maremagnum the accounting data was already available online.
3. BMS Gateway: developed in Java, handles the communication with the BMS. For Linate this module is not available because Honeywell has not granted access rights to the BMS. For Principe Pio it uses BACnet, while for Maremagnum, OPC.

1 INTRODUCTION

1.1 Purpose of this document

.

The purpose of this deliverable is to define, design and describe the interoperability and communication architecture of the BEMO system, in order to gather and dispatch information from the different units that compose the system.

First, an introduction, underlying the purpose of the document, its structure, and the relationships to project objectives and to other deliverable and tasks, is presented.

The document proceeds with an overview of the whole S4ECoB system, defining the units that communicate with the BEMO server. The BEMO server is a VM that runs virtually on a physical workstation installed locally on each pilot site thanks to virtualization software, server VM, and is composed by a database and the BEMO Control and Management Platform, a web application that integrates different software modules.

Then the document provides a detailed description of the software modules developed in order to permit the communication between the BEMO server and the units previously defined that compose the S4ECoB system and the external systems on the pilot sites, underlying functionalities, technological choices, logical workflow for the three different pilot sites.

Three network configurations, one for each demonstrator, are proposed together with accessibility instructions and modality to their private network. Then the testing and validating procedures of the software modules developed are presented.

Finally a short recap of the work that has been done and the solutions developed concludes the deliverable.

1.2 Structure of the deliverable

This document is structured as follows:

- Chapter 2 presents an overview of the whole S4ECoB system, defining the modules that compose the system and the way these modules communicate with BEMO system. A description of the BEMO system is also provided.
- Chapter 3 is the main chapter of this document and it provides detailed information about the four modules functionalities, requirements, technological choices, logical workflow, communication and integration with the BEMO system; focusing on the differences found in the three project pilots and the solutions adopted.
- In Chapter 4 the different network configurations and accessibility procedures for the three project pilots are listed, together with local testing and validation on site procedures.
- Chapter 5 reports some key conclusions about the achieved results.

1.3 Relationship to the project objectives

This deliverable describes the architecture of the BEMO system (Objective 5) connecting together all the various units that compose the overall system, like the Occupancy sensor (Objective 2) composed by the APU's (Objective 1) and corresponding audio front-ends and the Building Management System (BMS).
The deliverable focuses also on the different scenarios encountered in the three operative demonstrators (Objective 7) and presents the dedicated solutions developed.

1.4 Relationship to other deliverables and tasks

The presented deliverable takes inputs from T2.2, T3.2 and T4.1 and the associated deliverables.
From T2.2 "BEM System and demonstration sites requirements" it takes the technical requirements and specifications specific to each pilot sites, like functional requirements related to the BMS installed in the buildings.
From T3.2 "Design and Implementation of the acoustic processing unit" the technical specifications for the APU.
The APU gateway implemented in T4.1 "Integration of occupancy sensor network with BEMO server" is one of the software modules that handles the communication between the BEMO server and the APUs.
This document will be used as starting point for the development activities belonging to task T4.3 and T5.5.
Within T4.3 "Development of the BEMO management platform" the web platform will be developed taking into account the architecture design presented in this deliverable.
The D5.3 "Web-monitoring system of BEM performance" describes in details one of the software module defined in this document: the optimization unit.

1.5 Contributions of Partners

D'Appolonia had the main responsibility to prepare this document.

2 OVERVIEW OF THE WHOLE S4ECOB SYSTEM

The S4ECoB system, or BEMO system, consists of two main components:

1. Occupancy sensors with integrated Acoustic Processing Unit (APU) and up to 3 Audio Satellite Units (ASU) each connected with up to 8 microphones (hereinafter called **Occupancy Sensor**);
2. Building Energy Management Optimizer server (**BEMO server**).

The BEMO server is the central kernel of the architecture and needs to communicate also with external systems already installed on the pilot:

1. The ***Accounting Information System (AIS)***;
2. The ***Sensors and Metering System already installed in the buildings;***
3. The ***Building Management System (BMS).***

The BEMO server handles input in the form of occupancy level and audio data from the Occupancy Sensor, but also data from two external systems: the energy data from the BMS and the accounting data like the flight database in Linate or the accounting data for the two Spanish shopping centers.
The sensors and metering system (only accessible in the two Spanish shopping centers) is directly managed by the BMS (please refer to Deliverable 6.1 "Mid Term demonstration report") to which the BEMO server connects to. In this way is not necessary for the BEMO server to communicate directly with the additional sensors and meters installed in the buildings.
Within the BEMO server, a software application, the BEMO Control and Management Platform, integrates all the software communication modules described in the following Chapter 3 "Communication Architecture" together with other software components:

- APU Gateway: this communication module was developed by IMMS in order to simplify the communication between APU's and BEMO server and described in details in Deliverable 4.1 "Integration of occupancy sensor network with BEMO server";
- APU Gateway Interface: this communication module provides an interface to the APU gateway, refining the communication between BEMO server and APU's, and it is described in details in subsection 3.1.3 "APU Gateway Interface" in the present document;
- AIS Gateway: this communication module is responsible for the communication between AIS and BEMO server and is described in details in subsection 3.2.1 "AIS Gateway" in the present document;
- BMS Gateway: this communication module is responsible for the communication between BMS and BEMO server and is described in details in subsection 3.3.1 "BMS Gateway" in the present document;
- Communication platform: this module is accessible remotely over the Internet through VPN described in Chapter 4 "Network accessibility and Testing" in the present document, authenticating with appropriate credentials;
- Optimization unit: this module takes for input the energy data provided by the BMS, implements the optimization algorithms and provides the optimized set points based on the occupancy level detected by the Occupancy Sensor;
- Messaging system: this module checks the system status and sends automatic alerting messages to pilot and system responsible in the case some fault was detected in the overall system.

Another important component of the BEMO server is the database. In the database different kind of data, like occupancy level, audio data, energy data, accounting data, pilot site descriptions, are inserted by the different software modules composing the BEMO Control and Management Platform and here stored and retrieved.
The database is divided in nine main blocks of tables:

1. Audio data block
2. APU block
3. Pilot area block
4. Pilot accounting data block
5. Pilot energy data block
6. Optimization data block
7. User data block
8. Error management data block
9. Configuration data block

For further details on the BEMO Control and Management Platform or on the database please refer to Deliverable 4.3 “BEMO control and management platform”.

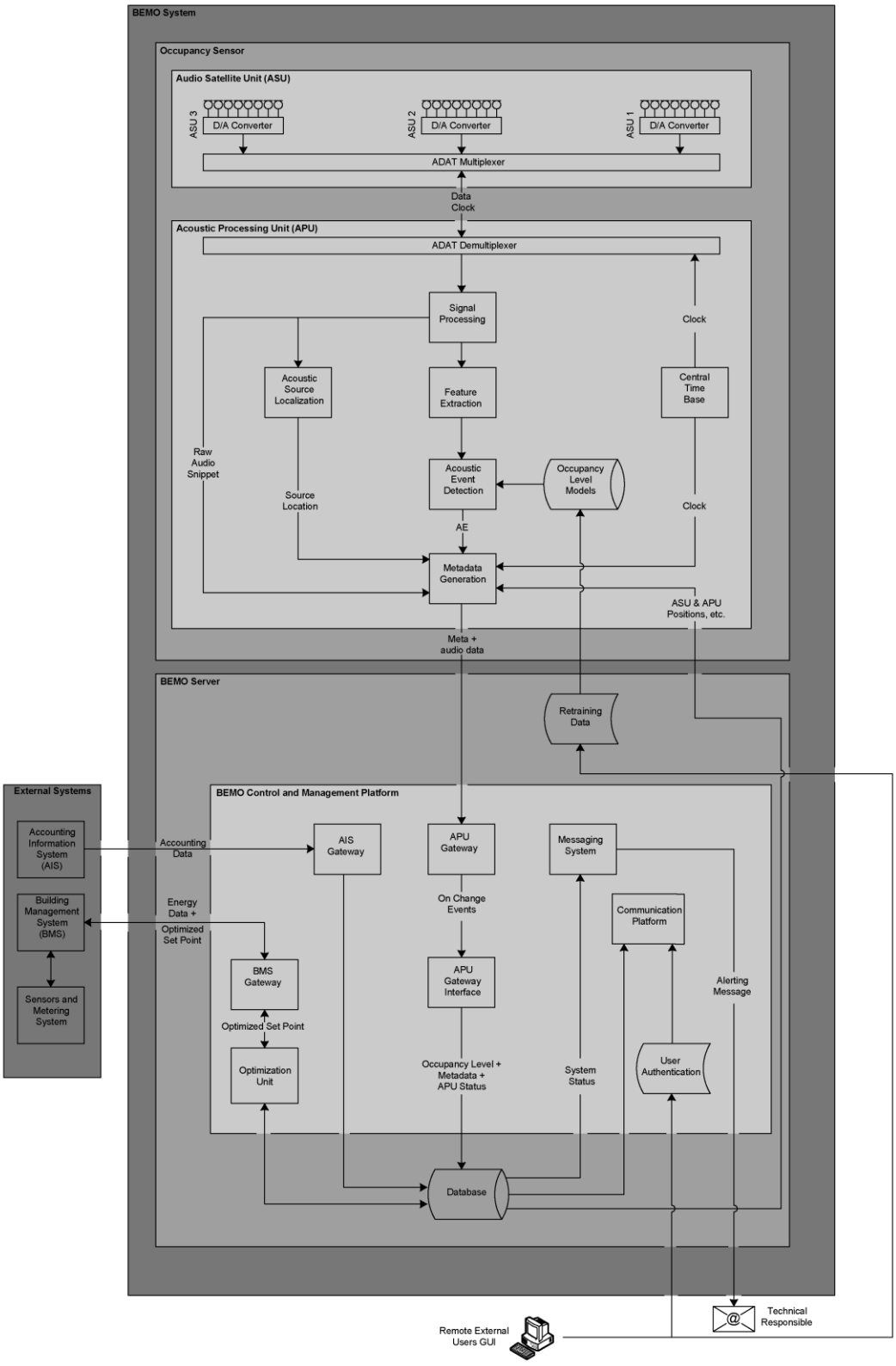


Figure 1 - Whole S4ECoB System architecture

2.1 Units Definition

Table 1 gives an overview of the four units composing the communication architecture presented and communicating with the BEMO server. The correct definition of these units is critical and very important for the development of the BEMO control and management platform that will be fully detailed in D4.3 "BEMO control and management platform".

Unit Name	System owner	Function	Input to the BEMO server
Occupancy Sensor	BEMO system	Processing of audio and acoustic data in a networked environment	Audio metadata, occupancy level, unknown and training sounds and status information of the Acoustic Satellite Units (ASUs) installed on the pilot site.
Accounting Information System	External system	Estimation of the number of people currently inside the building	Accounting data
Sensors and Metering System	External system	Sensing and metering of energy data	These units do not communicate directly with the BEMO but provide input to the BMS
BMS	External system	Monitoring the environment and controlling the HVAC system of the building	Energy data related to current and historical energy consumptions in the building related to the HVAC controlling system

Table 1 - S4ECoB architecture modules overview

2.2 BEMO Server

The BEMO server runs virtually on a physical workstation installed locally on each pilot site thanks to virtualization software, and is composed by the BEMO Control and Management Platform and a database. The BEMO Control and Management Platform is the web application that integrates different software modules developed in the framework of the S4ECoB communication architecture.

Workstation and BEMO server are the two main components of a virtual machine (VM) system: the host and the guest.[1] The physical workstation is the virtual machine host server; the underlying hardware that provides computing resources, such as processing power, memory, disk and network I/O, and so on. The BEMO server is the virtual machine guest: a completely separate and independent instance of operating system and application software, which is the virtual workload that resides on the host virtual machine and shares its computing resources.
The only requirement in this VM architecture is that the physical workstation must meet or exceed the minimum hardware requirements for the BEMO server.

The choice of the **host** machine has fallen on the Dell Precision T3600 [2] workstation for its characteristics that balance performance and scalability, offering great reliability and manageability, with the capability to handle a big processing workload.
The technical specifications for the workstation are summarized in Table 2.

Model	Dell Precision T3600
Processor	Intel Xeon CPU E5-1650 0 @ 3.20GHz
RAM	16GB
Operating System	Windows 7 Professional Service Pack 1
System type	64-bit
Disk drive	ATA ST1000DM003-1CH1 SCSI Disk Device
Display adapter	NVIDIA Quadro K600
Network adapter	Intel 82579LM Gigabit Network Connection

Table 2 - Workstation host technical specifications

This workstation provides a strong and reliable Microsoft Windows environment where it is possible to manage the BEMO server unix-like system and its applications and functionality properly designed and implemented for S4ECoB project.

On the physical workstation in the three pilot sites a virtualization software package from Oracle Corporation, **Oracle VirtualBox**[3] is installed as an application that allows additional guest operating systems (Guest OS) to load and run, each with its own virtual environment.
A package called Guest Additions was installed inside the BEMO server in order to optimize the guest operating system for better performance and usability.
Further details on guest VM configuration and installation are provided in D4.3 “BEMO control and management platform”.

The **guest** machine that has been configured is a Linux machine called BEMO server, hosted by the Dell Precision T3600 workstation. This guest virtual machine handles the occupancy level annotation and audio data recorded and processed by the APU through the APU Gateway Interface module and optimizes the energy data on the BMS through the optimization unit. On the BEMO server is installed the database of sounds.
A guest virtual machine (VM) is a software implementation of a machine that executes programs like a physical machine. Virtual machines are separated into two major classes, based on their use and degree of correspondence to any real machine:[4]

- A system virtual machine: provides a complete system platform which supports the execution of a complete operating system (OS) to run programs where the real hardware is not available for use.
- A process virtual machine: is designed to run a single program, which means that it supports a single process.

The BEMO server is a system virtual machine. This choice has clear advantages:

- Multiple OS environments can co-exist on the same primary hard drive, with a virtual partition that allows sharing of files generated in either the "host" operating system or "guest" virtual environment. Adjunct software installations, wireless connectivity, and remote replication can be generated in any of the guest or host operating systems. Regardless of the system, all files are stored on the hard drive of the host OS.

- Application provisioning, maintenance, high availability and disaster recovery are inherent in the virtual machine software selected.
- Can provide emulated hardware environments different from the host's instruction set architecture (ISA), through emulation or by using just-in-time compilation.

The technical specifications for the BEMO server are summarized in the Table 3.

Operative System	Linux Red Hat CentOS 6.5
CPU number	2
RAM	4GB
Storage	300GB
Network	bridge

Table 3 - BEMO server VM technical specifications

This configuration of the virtual machine has been chosen for satisfying S4ECoB requirements. The advantage to use a Linux OS is the strong stability for performing optimization calculations on open source software.

The BEMO Control and Management Platform is a web application running on the BEMO server.
Its design follows the 3-tiers layered architecture (Figure 2) to ensure encapsulation, maintainability, reuse and separation. The design covers the business logic layer, user interface layer and the data layer.

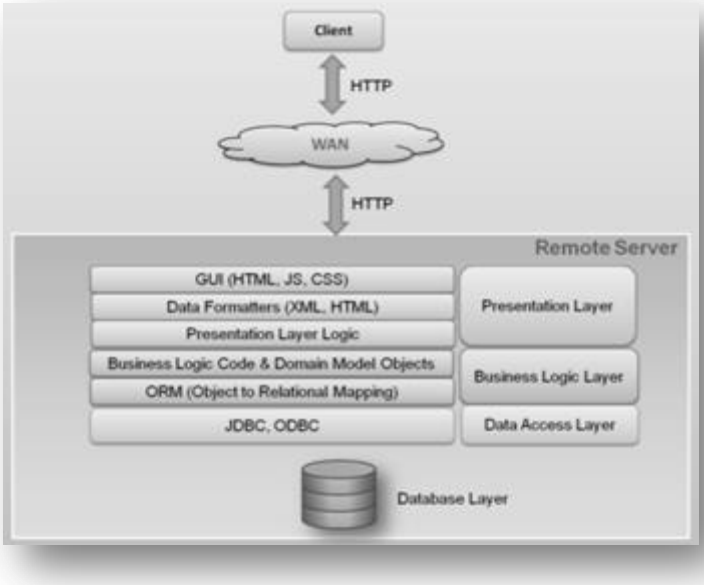


Figure 2 - Layered Architecture for BEMO Management Platform Web Application

For this kind of application with (relational) database requirements to persist the data, the three tier architecture was preferred to aid the separation of concerns in such a way that the business logic is kept separate from the display and Web pages. This enables future expansion of the software application to use a different display technology if required.
The web application is running on the BEMO server but is accessible remotely thanks to Virtual Private Network (see subsection 4.2.1 for further details).

For privacy, security and ethical issues data is kept locally in the pilot sites where each workstation has been installed directly. Secure remote access to the BEMO server is possible.

3 COMMUNICATION ARCHITECTURE

This section describes in more detail four units already presented in this document, underlying their functionalities, the technological choices, the logical workflow and their communication within the BEMO server.

3.1 Occupancy Sensor

Two software modules handle the communication between the BEMO server and the Occupancy Sensor (including APU's) installed on the pilot: the APU gateway and the APU Gateway Interface module.

3.1.1 APU

Due to the scalability of the S4ECoB system, it supports an arbitrary number of APUs on the level of the acoustic or occupancy sensor network (the number of APUs is only limited by the number of available IPv4 network addresses in the corresponding subnet). Each APU is connected with up to three ASUs on the level of the audio sensing network.
The fully detailed hardware and software APU specifications are given in D3.2 "Design and Implementation of the acoustic processing unit".

3.1.2 APU gateway

The APU gateway is an endpoint for all occupancy sensor communication; it integrates and connects the occupancy network with other components of the S4ECoB system thanks to the BEMO server.
The communication between APU and APU gateway is based on the TCP/IP protocol. This allows the use of existing IT infrastructure in the demo sites.
The APU gateway receives the data, occupancy level, unknown sounds and status information from the APUs in the occupancy sensor network. This information will be caught by the APU Gateway Interface module that will allow the data to be accessible from the BEMO server, behaving like an interface between BEMO server and sensors.
The S4ECoB APU gateway is realized as a software module, it is a flexible solution, adaptable for the different requirements of the demo sites and it is integrated on the BEMO Web Application without any extra hardware as a separated Java process launched from a shell.
The full detail on APU gateway development is given in D4.1 "Integration of occupancy sensor network with BEMO server".

3.1.3 APU Gateway Interface

For the acquisition of occupancy level annotations, sound files and audio metadata recorded and processed by APUs and sensors, a software module is necessary that enables the BEMO server to obtain data provided by the APU gateway and feeds the project database of sounds.
The APU Gateway Interface module will behave like an interface to the APU gateway and the BEMO server.
The software requirements of the APU Gateway Interface module are:

- Capability to connect to D-Bus and interact with the APU gateway.
- Capability to connect to database of sounds filling tables with appropriate data.

- Compatibility with BEMO server specifications (it should run on the BEMO server).
- Integrated into the BEMO control and management platform.

The technological choice for the implementation of the software module is C and integration with the Java based BEMO Web Application as a separate process launched from a shell.
The advantage of this choice is the easier integration with the APU Gateway on the D-Bus. The application exploits a C++ class automatically created through the command *qdbusxml2cpp* from the xml interface of the APU Gateway on D-Bus. The application is also dynamically configurable thanks to a configuration file in which is memorized the information related to database and pilot and supports error handling through a log file and filling the appropriate table on the database of sounds. The Gateway Interface connects to D-Bus and catches all the signal "on change" emitted by the APU Gateway and for each of these events writes a new record in the database of sounds. These events are:

1. apuStatusChanged
2. directionChanged
3. rmsChanged
4. annotationChanged
5. soundFile

The - Communication between APU Gateway and APU Gateway Interface modules is presented in Figure 3.

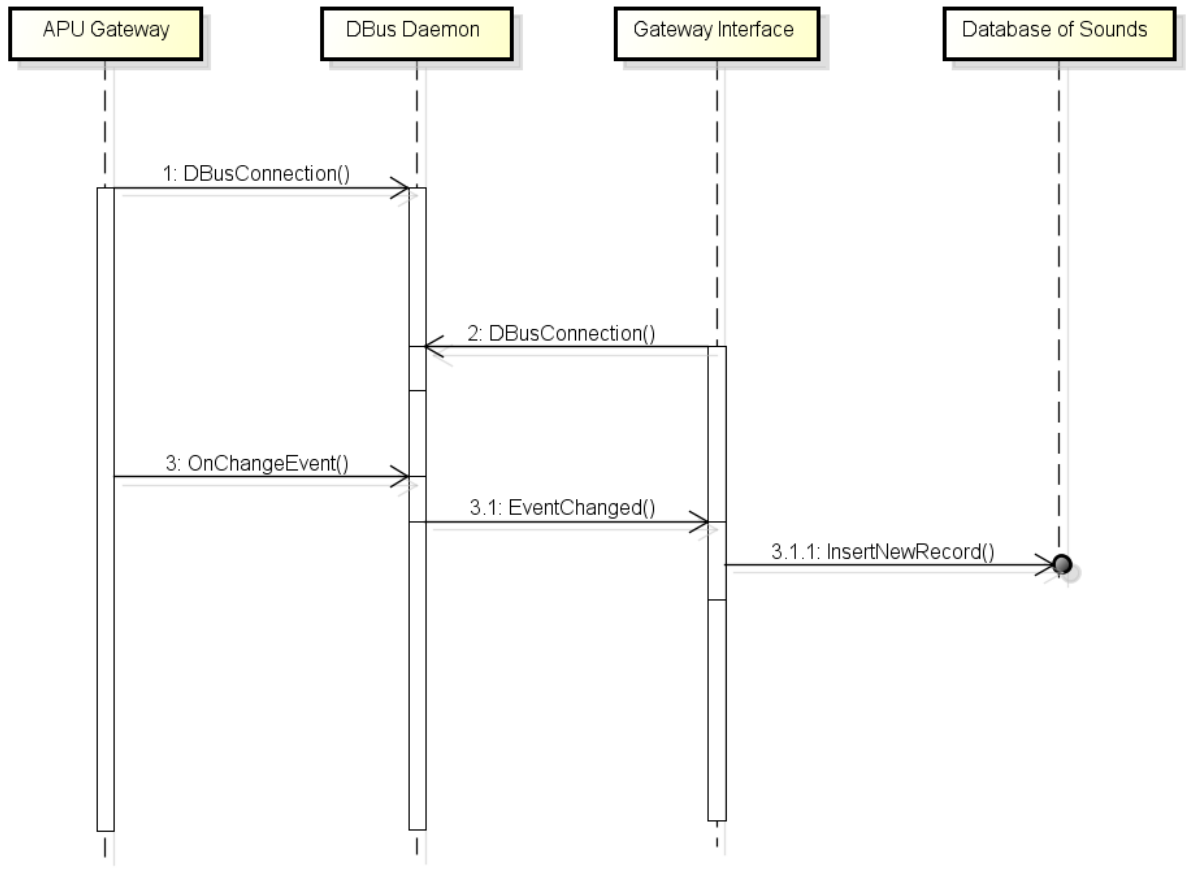


Figure 3 - Communication between APU Gateway and APU Gateway Interface modules

For correct usage the APU Gateway Interface application has to be launched after the APU gateway to successfully register to D-Bus and catch "on change" events from the APU gateway.

The two modules should connect to the system D-Bus instead to the session D-Bus in order to reach a correct integration within the BEMO Control and Management Platform.

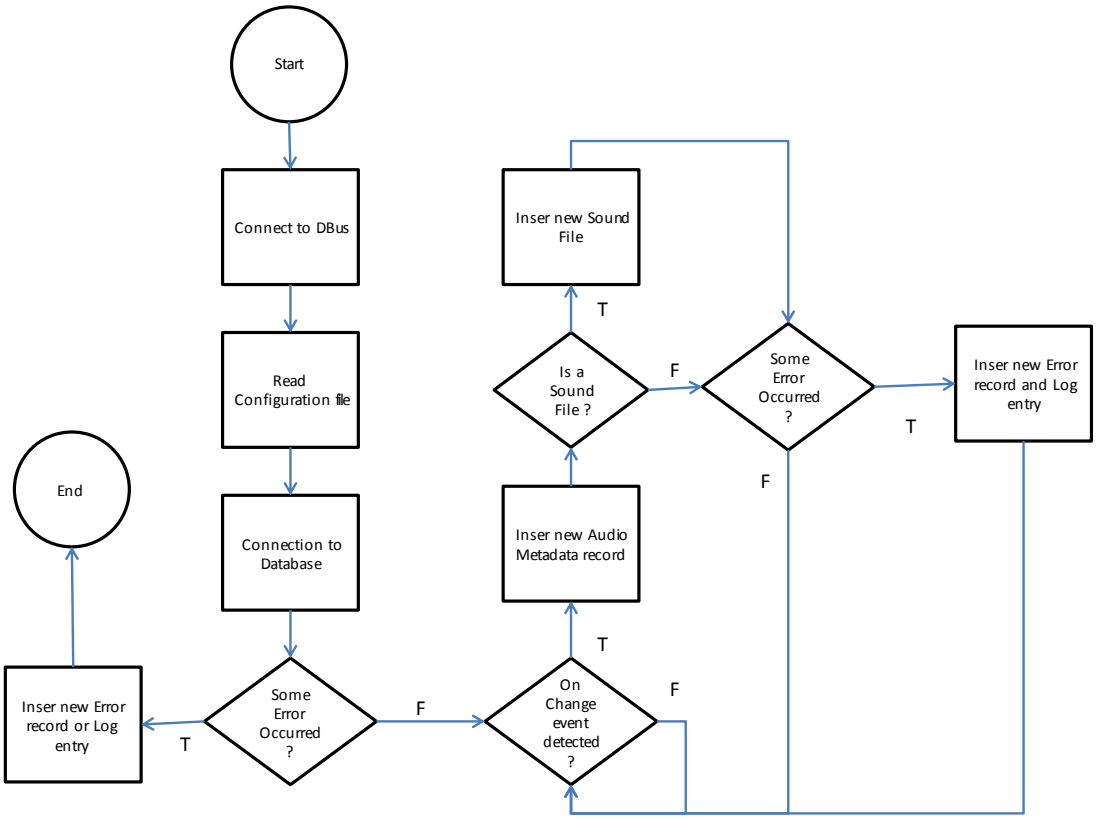


Figure 4 – APU Gateway Interface logical workflow

3.2 Accounting Information System

An **Accounting Information System** (**AIS**) [5] is a system of collecting, storing and processing financial and accounting data that is used by decision makers. An accounting information system is generally a computer-based method for tracking accounting activity in conjunction with information technology resources.

For Maremagnum shopping centre in Barcelona the accounting data is directly provided from a web platform in near real time at the URL https://www.centreanalyser.com, while for Milan Linate airport and Principe Pio shopping centre in Madrid, it is not possible to obtain real time data, but accounting data is available for previous days.

For this reason the technological choice adopted for Maremagnum is to access the data through web service, while for Linate and Principe Pio an automatic import procedure within the database has been developed in order to make the data accessible by the BEMO Web Application.

3.2.1 AIS Gateway

The AIS gateway has been developed as a software module with the following requirements:

- Dynamically configured folders “to processed” and “processed”.
- Capability to automatically process Comma Separated Files (CSV).
- Designed to be easily scalable for different CSV file header formats.

- Capability to connect to the project database inserting the accounting data.
- Compatibility with BEMO server specifications (it should run on the BEMO server).
- Integrated into the BEMO control and management platform.

The technology chosen for processing accounting data in CSV format is Super CSV[6] because it is the foremost, fastest, and most programmer-friendly, free CSV package for Java., which offers a list of features not found together in other CSV Java packages. These features are:

- POJO support
- Automatic CSV encoding
- Highly configurable
- Data conversion
- Constraint validation
- Stream-based I/O

In Figure 5 the logical workflow of the AIS gateway module is presented.

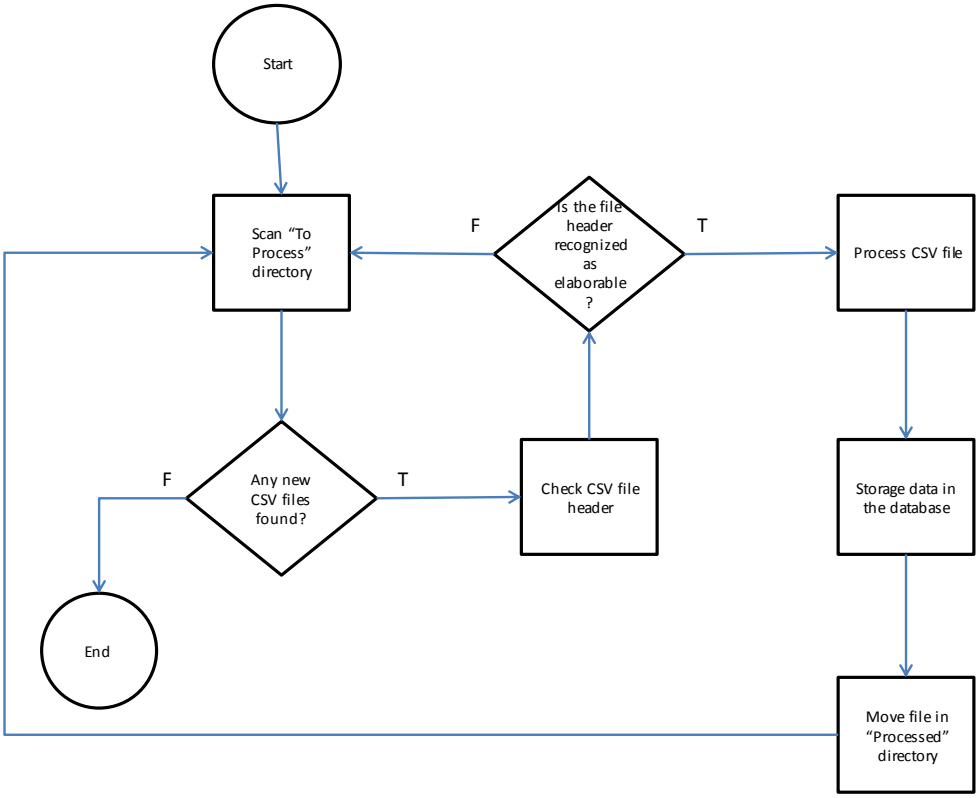


Figure 5 - AIS gateway module logical workflow

3.3 BMS

The Building Management System (BMS) is one of the units composing the S4ECoB system and handles the energy data provided by the sensors and metering system already installed in the buildings, and thus provides energy data to the BEMO system. For reaching the optimization objective of the project, connecting together the BEMO system with the local Building Management System (BMS) is crucial.

It is worth to remind that in the three pilot sites three different BMS are present (as described in D2.2 "BEM system and demo site buildings requirements") and briefly listed here:

1. in Milan Linate airport pilot, the system SymmetrE provided by Honeywell
2. in Principe Pio shopping mall pilot, the system Desigo Insight provided by Siemens
3. in Maremagnum shopping mall pilot, the system Tac Vista provided by Schneider

For this reason a module that handles the communication between BMS and BEMO server has to be developed: the BMS Gateway This module will be a software module in order to avoid further costs associated to a dedicated hardware.

3.3.1 BMS Gateway

The software requirements for the BMS Gateway are:

- Capability to connect to the BMS, read and write energy data.
- Designed to be easily scalable for different automation protocols and BMS.
- Compatibility with BEMO server specifications (it should run on the BEMO server).
- Integrated into the BEMO control and management platform.

For the three project pilots, a detailed study of automation communication protocols supported by the three different BMS and exploitable for S4ECoB purposes has been analyzed. In particular the study focused on OPC and BACnet protocols was done according to the specificity of the two BMS installed on the Spanish demonstration sites. In fact, the OPC protocol has been considered because the TAC Vista in Maremagnum has an OPC server installed, while the BACnet protocol instead is used in Principe Pio where the Desigo Insight Sx Open is a BACnet server.
It is worth to underline that for Linate, no interaction with Honeywell SymmetrE BMS system has been established due to the fact that Honeywell has not granted rights to access directly the BMS but the data will be available through an extraction procedure.

The data gathered from these systems is integrated in the database through an automatic import procedure, the same that has been described in detail in this document in the subsection "Accounting Information System".
The general BMS gateway is composed of three steps:

1. Initialization and connection to the BMS
2. Reading data from the BMS
3. Writing data to the BMS

The logical workflow of the BMS gateway is straightforward and presented in Figure 6.

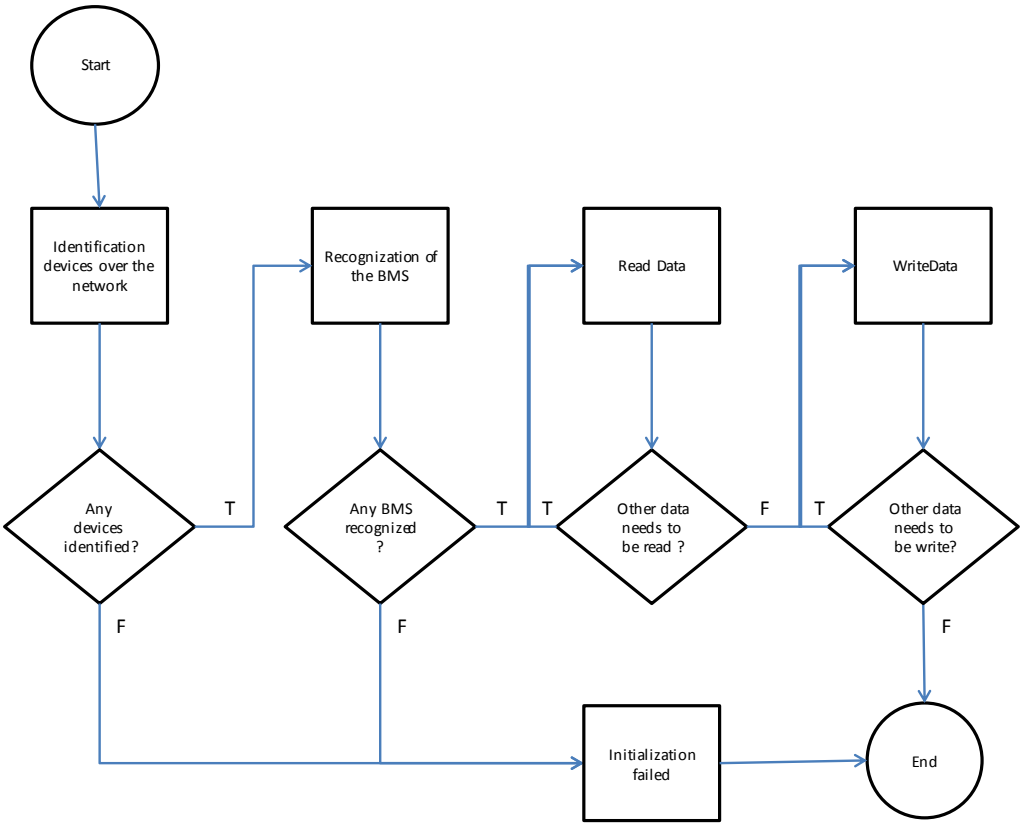


Figure 6 - Logical workflow of the BMS gateway

In the following subsections the detailed study for the different demonstrators is presented. After an introduction on the communication protocols analyzed and the description of the equipment installed in the pilots, the technological solution chosen and developed for BEMO – BMS communication is explained.

3.3.2 Principe Pio Connectivity

For the project pilot of Principe Pio, the Desigo Insight Sx Open installed on the BMS side is a BACnet server, so a detailed study the BACnet standard communication protocol was performed and the solution developed is presented in this subsection of the present document.

3.3.2.1 BACnet Communication Protocol

BACnet[7], a registered trademark of ASHRAE, stands for "Building Automation Control Network", is a standard data communication protocol for building automation and control networks and enabling interoperability between different building systems and devices. BACnet is both an international (ISO) and ANSI standard.

With BACnet every device contains a device object that defines certain device information.

Furthermore, BACnet divides the task of device interoperability into three distinct areas:

1. **Objects** (information): all information within an interoperable BACnet device is modeled in terms of one or more information objects. The BACnet standard defines 54 different standard object types. Each object is identified with a 32-bit binary number containing a code for the

object type and the object instance number, called object identifier. In addition, every object, has a collection of **properties** that define the object. Each property contains two pieces of information: a property identifier and the property's value. A property's purpose is to allow other BACnet devices to read and write information about the object containing the property.

2. **Services** (action requests): BACnet services are formal requests that one BACnet device sends to another BACnet device to ask it to do something. Services are grouped into five categories of functionality – object access, device management, alarm and event, file transfer, and virtual terminal.
3. **Transport systems** (internetworking, electronic messages): The model of objects and services is realized by encoding messages into a stream of numeric codes that represent the desired functions or services to be performed, following a common encoding language to all BACnet devices. BACnet defines seven network types, which serve as the transport for BACnet messages and encompass the physical and datalink layers of the protocol. A BACnet router, a device that links dissimilar network types, is used to join multiple network types.

3.3.2.2 Siemens Desigo Sx Open

Siemens Desigo SX Open [8], installed in Principe Pio pilot, is a configurable 3rd party system – BACnet/IP gateway. Using this gateway, both Desigo PX automation stations (peer-to-peer) and the Desigo Insight management station have access to integrated (3rd party) data points over the IP network. It is possible to define BACnet servers to map third party datapoints on BACnet objects including alarming (e.g. High/Low Limits) and processing like, trend logging and time scheduling. SX Open is ideal for the integration of any size of data points.
Together with the BACnet server is an OPC client integrated that supports DA v2.x and v3.0 specifications. Unfortunately in Principe Pio shopping centre it was not possible to install an OPC server.

3.3.2.3 Technological solution: BACnet integration

In Principe Pio, in order to connect with Siemens Desigo BMS, a software module integrated into the java based BEMO Management Platform Web Application has to be developed.
The communication based on the BACnet protocol could be managed with two technologies:

- BACnet Stack
- BACnet4J

The **BACnet Stack** [9] library provides a BACnet application layer, network layer and media access (MAC) layer communications services. It is an open source (GPL license), royalty-free library for an embedded system, Windows, Linux, or other operating system. The source code is written in C for portability.
BACnet4J [10] is a high-performance, open source implementation of the BACnet I/P and MS/TP protocol written for Java (minimum version 1.5) by Serotonin Software. BACnet4J supports all BACnet services and full message segregation and can be used for field devices or for control platforms.
The great advantage of BACnet4J is that it is 100% Java code making simpler the integration in the BEMO Management Platform Web Application.
The disadvantage of BACnet4J library is that unfortunately it does not support BACnet/IP Broadcast Management Device (BBMD)[11] so it will not possible to communicate with BACnet devices in a different network, like the Siemens dedicated network in Maremagnum directly from the BEMO server. The installation of a dedicated computer installed into Siemens dedicated network in Principe Pio will be required in order to overcome this disadvantage. In this way BACnet4J could run on the dedicated

computer and the BEMO could communicate with the micro-laptop through Java Remote Method Invocation (RMI), a Java API that performs the object-oriented equivalent of remote procedure calls (RPC).

3.3.3 Maremagnum Connectivity

For the project pilot of Maremagnum, the TAC Vista OPC server was installed on the BMS side, so a detailed study the OPC standard communication protocol was performed and the solution developed is presented in this subsection of the present document.

3.3.3.1 OPC Communication Protocol

OPC[12] is an interoperability standard for the secure and reliable exchange of data for industries and automation, developed and maintained by the OPC Foundation. The OPC standard is composed by a series of specifications that defines the interface between Clients and Servers, including access to real-time data, monitoring of alarms and events, access to historical data and other applications.
Initially OPC, that stands for OLE (object linking and embedding) for Process Control, was restricted to the Windows operating system (OPC DA specifications), then the OPC UA specifications was developed in order to provide a feature-rich technology open-platform architecture.
The underlying layer of the OPC Data Access (DA) specification[13] is based on Microsoft's COM/DCOM technology, which enables inter-process communication in a variety of programming languages for the Windows operating system. COM provides a communication interface layer that allows local and remote procedure calls between processes. DCOM (Distributed COM) is the natural extension of COM to support communication among objects on networked computers. For additional specifications on DCOM security, refer to the DCOM Technical Overview paper on the Microsoft TechNet website.
The data access server has three divisions:

- **Server**: Contains all of the group objects
- **Group**: Maintains information about itself and contains and organizes the OPC items
- **Item:** contains a unique identifier held within the group..

3.3.3.2 TAC VISTA OPC Server

Tac Vista OPC server [14], installed in Maremagnum is an add-on module to Tac Vista server, which enables third party OPC-compliant presentation systems to interface with TAC products. Vista OPC Server consists of two separate interfaces:

- OPC Data Access (DA)
- OPC Alarms and Events (AE) interface

All objects and signals in the TAC Vista Server database can be read and changed via the OPC Data Access interface. The OPC Alarms & Events interface makes it possible to subscribe to and acknowledge alarms from the TAC Vista OPC Server. TAC Vista OPC Server is implemented as a Windows service and can be started from the Computer Management console.
The TAC Vista OPC Server is compatible and supports the following specifications:

- OPC Data Access 1.0a
- OPC Data Access 2.05a
- OPC Alarms and Events 1.10

3.3.3.3 Technological solution: OPC client

In Maremagnum in order to connect to Tac Vista OPC server, it will be possible to develop an OPC client to connect to the BMS. Before managing to connect to the BMS, the TAC Vista OPC server has to be configured as a service on the BMS together with the associated user.
The best way to develop an OPC client integrated in the BEMO Management Platform Web Application is to write java code that is open source and that could run under Linux BEMO server OS. The Linux compatibility prevents DCOM configuration of client machine (not possible under Unix OS).
There are two open source Java based solutions for developing an OPC client:[15]

- JEasyOPC
- Utgard

Antonín Fischer's Java OPC Client, **JEasyOPC**,[16] full source code is hosted on SourceForge. It uses a JNI layer coded in Delphi and works only in Windows OS supporting both DA 2.0 and 3.0 specifications.
Utgard[17] is an open source project offering an OPC library written in pure native Java, with no dependency on JNI or other DLLs, licensed under the GPL and currently supporting DA 2.0 specifications for client development, with server enumeration now implemented. DA 3.0 client and DA server side are planned to follow. Utgard is a contributor to the larger openSCADA project, and uses j-Interop for DCOM interoperability. It is a 100% pure JAVA OPC Client API, that can be used independently from other openSCADA projects requiring no more than an up to date Java environment (Java 1.6.x+). This way, it is possible to connect to an OPC server running on Windows directly from your application, regardless of which operating system (Unix/Linux/Mac OS) your application is running on. The Utgard Project for now cannot help for running OPC servers on non-windows operating systems.
JEasyOPC has some disadvantages:

- The core library is not in Java but in Delphi
- Cannot run under Linux

It seems that Utgard is exactly what is needed for connecting to Tac Vista OPC server in Maremagnum from the BEMO Management Platform Web Application for S4ECoB purposes:

- It's 100% Java code
- It's open source
- Can run under Linux
- Supports DA 2.0 specifications
- It's independent from other openSCADA projects

4 NETWORK ACCESSIBILITY AND TESTING

4.1 Network Configuration

For the aim of the project the three project pilots of Linate in Milano, Principe Pio in Madrid and Maremagnum in Barcelona, have been configured with three different network configurations according to their already existing private network rules.
These network configurations are summarized in these subsections through tables.

4.1.1 Network configuration Linate

The network configuration in the Italian pilot of Linate is presented in Table 4.

APU IP address	10.1.155.2
Netmask	255.255.248.0
Network	10.1.155.0
Gateway	10.1.159.241
DNS-nameserver	10.1.88.1
BEMO-server IP address	10.1.155.1
Workstation IP address	10.1.155.7

Table 4 - Linate network configuration

4.1.2 Network configuration Principe Pio

The network configuration in the Spanish pilot of Principe Pio is presented in Table 5.

APU IP address	10.100.150.3
Netmask	255.255.255.0
Subnet	10.100.150.0/24
Gateway	10.100.150.1
BEMO-server IP address	10.100.150.2
Workstation IP address	10.100.150.7
BMS IP address	10.128.3.69

Table 5 - Principe Pio network configuration

4.1.3 Network configuration Maremagum

The network configuration in the Spanish pilot of Maremagnum is presented in Table 6:

APU IP address	192.168.10.3
Spare APU IP address	192.168.10.4
Submask	255.255.255.0
Gateway	192.168.10.1
BEMO-server IP address	192.168.10.2
Workstation IP address	192.168.10.7
BMS IP address	169.254.249.250

Table 6 - Maremagnum network configuration

4.2 Network Accessibility

One of the objectives of S4ECoB project is remote control of S4ECoB system through the BEMO server installed virtually on the workstation, physically present in each project pilot sites. To ensure this objective while taking care of network security issues, a Virtual Private Network (VPN) specific for each pilot site has been used for obtaining access to the pilot private network over the Internet. In addition a Virtual Network Computing (VNC) application software has been installed and configured on each BEMO server in order to provide direct access to it and enabling remote control of S4ECoB system.

4.2.1 Virtual Private Network

A virtual private network (VPN)[18] extends a private network across a public network, such as the Internet. It enables a computer or network-enabled device to send and receive data across shared or public networks as if it were directly connected to the private network, while benefiting from the functionality, security and management policies of the private network. A VPN is created by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryption.
A VPN connection across the Internet is similar to a wide area network (WAN) link between websites. From a user perspective, the extended network resources are accessed in the same way as resources available within the private network.
VPNs cannot make online connections completely anonymous, but they can usually increase privacy and security. To prevent disclosure of private information, VPNs typically allow only authenticated remote access and make use of encryption techniques.
VPNs provide security by the use of tunneling protocols and often through procedures such as encryption. The VPN security model provides:

- Confidentiality such that even if the network traffic is sniffed at the packet level (see network sniffer and Deep packet inspection), an attacker would only see encrypted data;
- Sender authentication to prevent unauthorized users from accessing the VPN;
- Message integrity to detect any instances of tampering with transmitted messages.

4.2.1.1 VPN Linate

The VPN for Linate is Firepass SSL VPN. The access for this VPN is web based and optimized for Mozilla Firefox web browser. Before accessing the VPN the corresponding certificate (extension .p12) has to be installed in the web browser. After the installation of the certificate the login form shown in Figure 7 will appear. Then it will be possible to access Linate private network thanks to the appositely created and provided credentials.

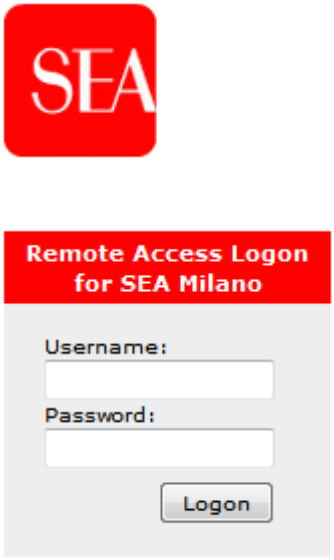


Figure 7 - Login to Linate VPN

The FirePass SSL (Secure Socket Layer)[19] VPN provides secure remote access to enterprise applications and data for users over any device or network. FirePass ensures easy access to applications by delivering outstanding performance, scalability, availability, policy management, and endpoint security. The result is unified security enforcement and access control that increases the agility and productivity of your workforce.

4.2.1.2 VPN Principe Pio

The VPN for Linate is accessible through Cisco VPN client. The access for this VPN is application based. After the application has been installed, it will be possible to create a new entry to which connect into. The new entry form is shown in Figure 8. It will be possible to access Principe Pio private network selecting the corresponding certificate (extension .pcf) and inserting username and password.

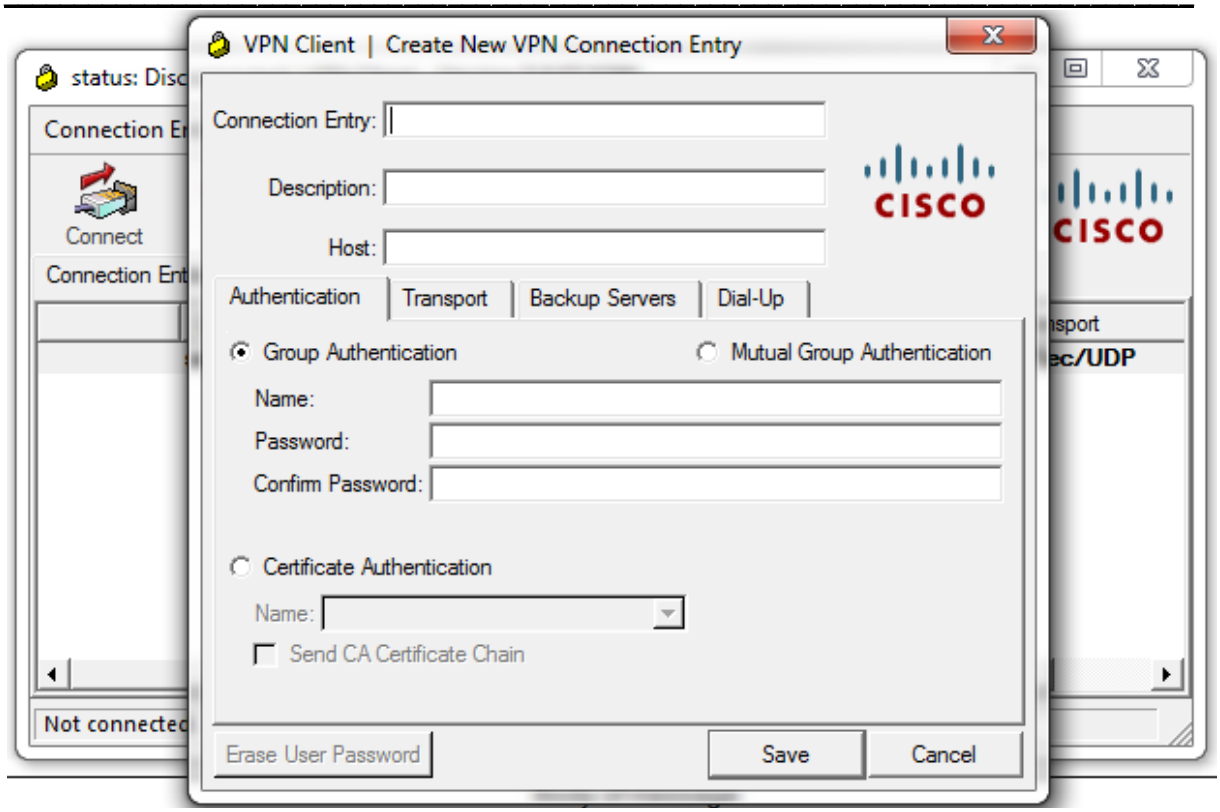


Figure 8 - Principe Pio VPN

4.2.1.3 VPN Maremagnum

The VPN for Maremagnum is Cyberoam SSL VPN. The access for this VPN is application based. After the application has been installed, the user has to select Server Settings and type the correct IP address for the server. Then the login form shown in Figure 9 will enable the access in Maremagnum private network after the insertion of the appropriate username and password.

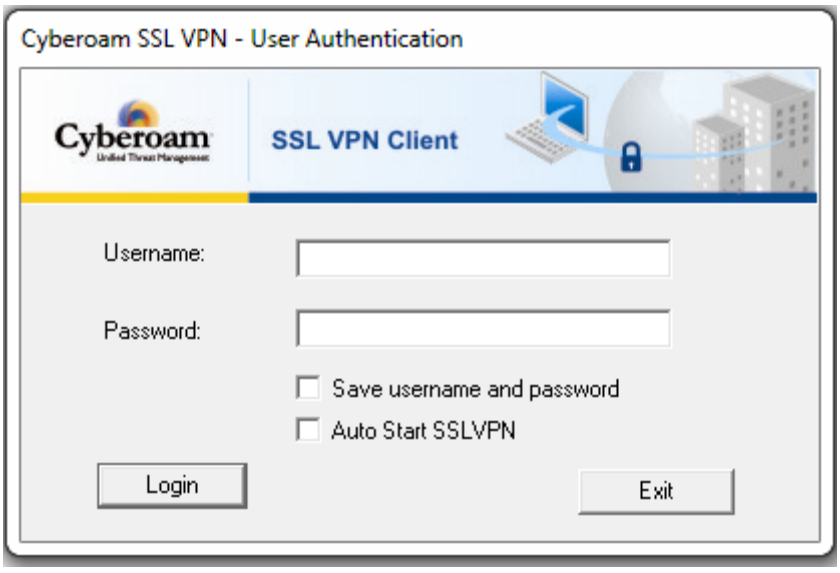


Figure 9 - Maremagnum VPN

Cyberoam SSL (Secure Socket Layer)[20] VPN provides simple-to-use, secure access for remote users to the corporate network from anywhere, anytime. It enables creation of point-to-point encrypted tunnels between remote user and company's internal network, requiring combination of SSL certificates and a username/password for authentication.

4.2.2 Virtual Network Computing

Virtual Network Computing (VNC)[21] is a platform-independent, graphical desktop sharing system to perform remote control of another computer using the Remote Frame Buffer protocol (RFB) as shown in Figure 10.

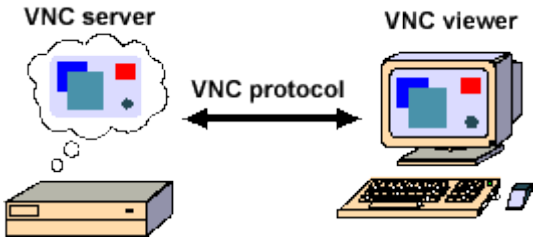


Figure 10 - VNC architecture

VNC by default uses TCP port 5900+N, where N is the display number (usually :0 for a physical display).
On the BEMO server VNC has been installed and the users are configured on port 5901 and 5902. So it is possible to perform remote control of BEMO server with VPN active.

4.3 Testing

The software modules presented in this document have been tested locally on our test environment and then validated directly on the three pilot sites.
All the functionalities that could be tested locally without real environment requirements, have been tested in the test environment thanks to three different instruments: JUnit, java logging system, and debugging mode.
JUnit is a unit testing framework for the Java programming language, which has been used in the test-driven development.
The **java.util.logging package** has provided the recording of the activities performed by the software, simplifying testing and validation phases.
In **debugging mode** has been possible to locally find and reduce the number of bugs, or defects in the software, making it behave as expected before performing validation phase directly on the three pilot sites.
All the code has been also sub versioned for optimizing the development and testing.

4.3.1 Local Testing

Due to the fact that the software modules developed in T4.2 and described in the present deliverable have the main function of handling the communication between BEMO system and the previously defined units belonging to the real environment of the demonstrator sites, the preparation of a local test environment that simulates the real one is necessary.
The local test environment was possible thanks to a virtual machine running on D'Appolonia server identically configured to the BEMO server installed on the three project pilot sites.

4.3.1.1 APU gateway simulator

On the local VM, the Gateway Interface software module has been developed and tested thanks to a simulator of the APU gateway software. This simulator presents exactly the same interface over the D-Bus of the APU gateway, simulating an APU network with a configurable number of APU which changes dynamically. All the real APU gateway events are also simulated and transmitted over the D-Bus interface using the respective signals. For further details on the APU gateway simulator, please refers to D4.1 "Integration of occupancy sensor network with BEMO server".

4.3.1.2 Principe Pio: RMI and BACnet server simulator

For the Spanish shopping centre demonstrator of Principe Pio in Madrid the BMS gateway has been tested in the local test environment, running a java Remote Method Invocation (RMI) software on another laptop in D'Appolonia network and one instance of a BACnet server simulator from the already mentioned BACnet stack library on a third machine in the same network. A BACnet server with the same object and properties of the Siemens Desigo BACnet server physically installed in the real environment of Principe Pio was simulated.

4.3.1.3 Maremagnum: OPC test server

For the Spanish shopping centre demonstrator of Maremagnum in Barcelona the BMS gateway has been tested in the local test environment, installing a simple OPC TOP server into one D'Appolonia server, after performing DCOM configuration on the machine, and configuring the OPC server with the appropriate groups and items, in a way to simulate the real Tac Vista OPC server physically installed in the real environment of Maremagnum.
After a successful detection of the simulated server, connection into it, reading and writing of the desired items, the software module was ready to the validation phase on Maremagnum real environment.

4.3.2 On Site Validation

When the software modules were completely working and optimized under the local test environment and all the issues and bugs reproducible locally have been solved, they are ready for the validation phase directly in the pilot sites in real environments.
During the validation phase any other issues need to be resolved before the final version could be released.

4.3.2.1 APU and APU gateway

The real environment in the three pilot sites has been set, configuring the APUs after installation, and installing and launching the APU gateway software on the BEMO server.
Some issues related to different annotations and version of the APU gateway software module have been solved in the real environment during the validation phase of the Gateway Interface software module that has been finally successfully completed and integrated in the BEMO Control and Management Platform.
Figure 11 shows a screenshot of the successful validation of Gateway Interface software module in real pilot environment with APU Gateway software.

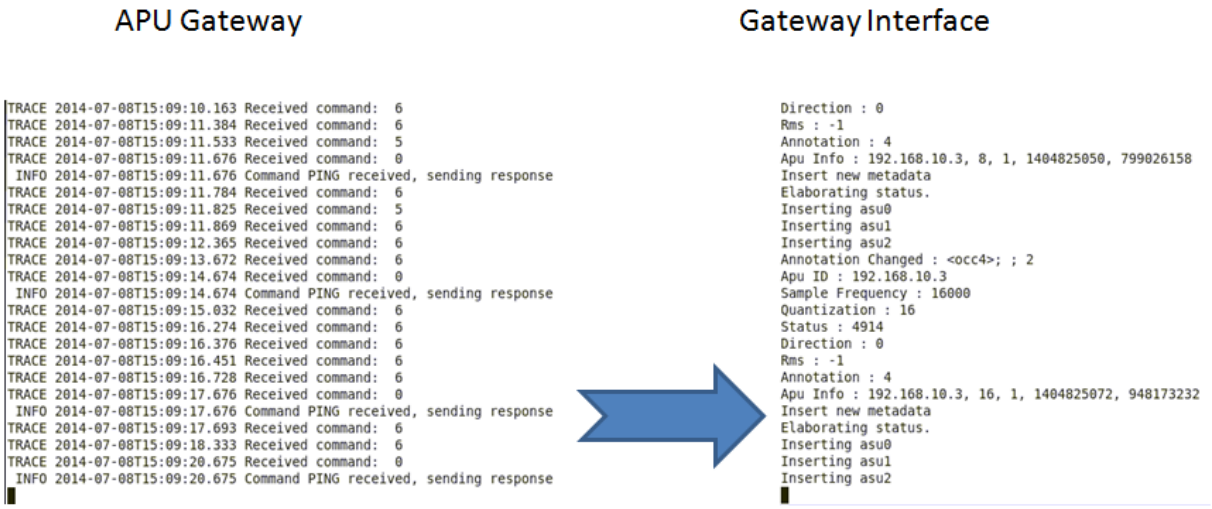


Figure 11 - Gateway Interface successful validation

4.3.2.2 Principe Pio: real BMS

In the real environment of Principe Pio demonstrator site, the real BMS has been studied directly on the field thanks to Innea BACnet Explorer software and the BMS gateway was successfully connected to the BMS, being able to read and write data from and into it, after resolving some networking issues.
It has been necessary to create a tunnel between S4ECoB dedicated network and Siemens dedicated network in order to give the possibility to the BEMO system to connect to the BMS because the two networks were not able to see each other. The ports 47808 and 47809 have been opened on the firewall in order to facilitate BACnet communication.
Figure 12 shows a screenshot of the successful validation of BMS gateway in Maremagnum pilot site.

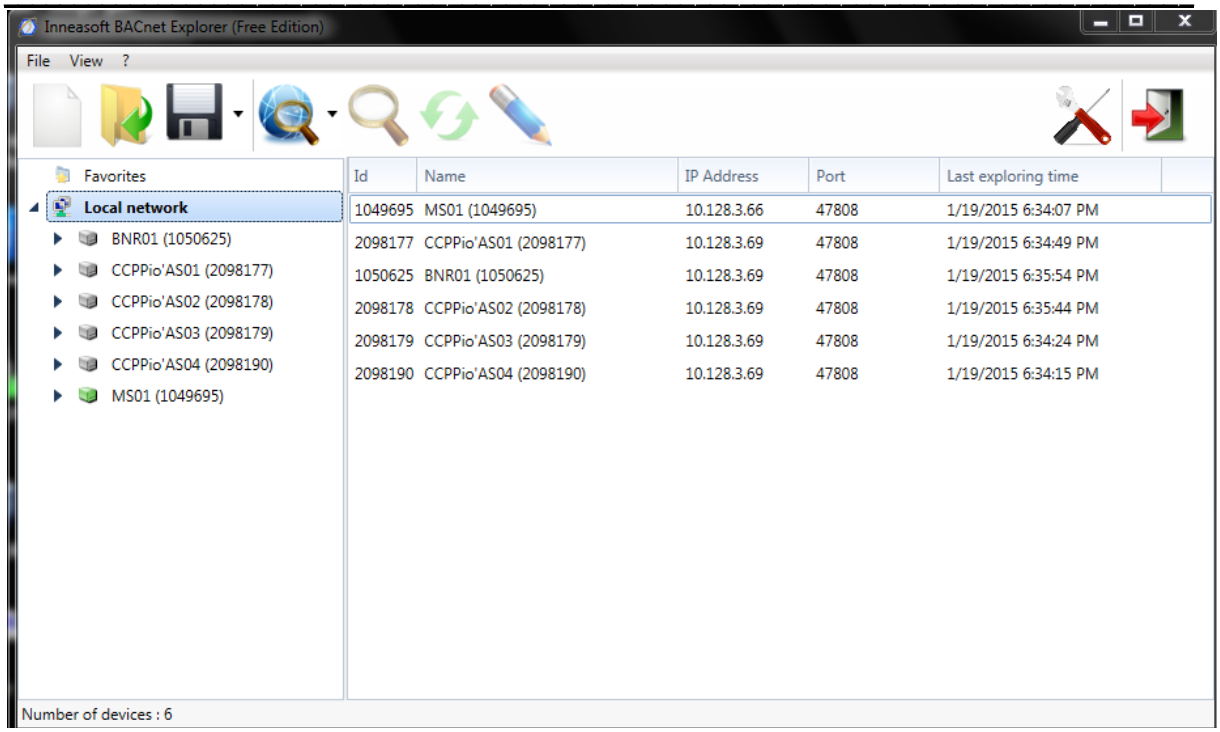


Figure 12 - Principe Pio BEMO - BMS validation

4.3.2.3 Maremagnum: real BMS

In the real environment of Maremagnum demonstrator site, the real BMS has been studied directly on the field thanks to Matrikon OPC Explorer software and the BMS gateway was successfully connected to the BMS, being able to read and write data from and into it, after resolving some configurations issues.

OPC Tac Vista server has been configured as a service on the BMS side and has been granted administrators access rights to the OPCTAC user.

Figure 13 shows a screenshot of the successful validation of BMS gateway in Maremagnum pilot site.

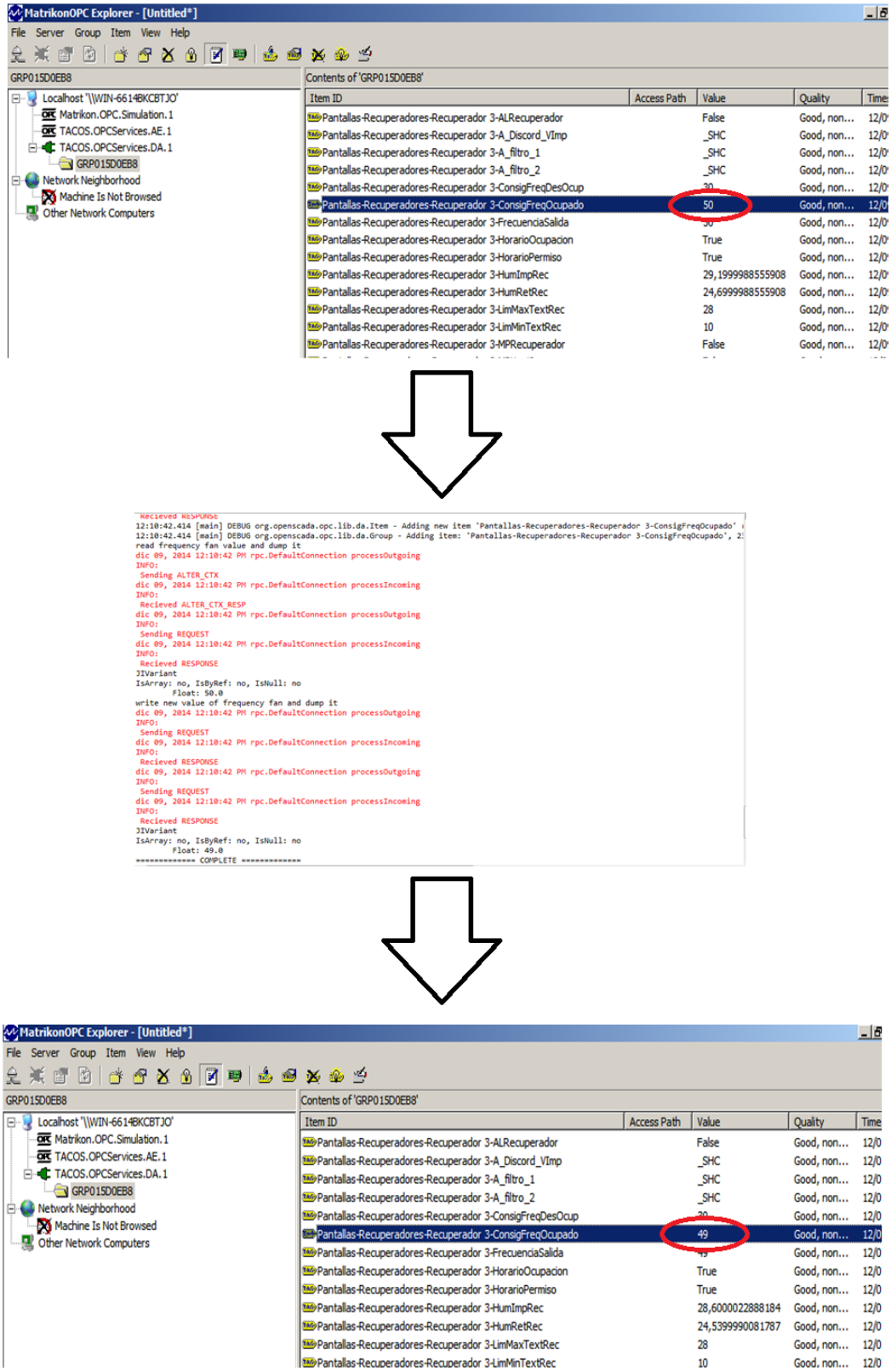


Figure 13 - Maremagnum BEMO BMS validation

5 CONCLUSIONS

The objective of T4.2 associated to this deliverable is to design an interoperable architecture to gather and dispatch information from the different units that compose the BEMO system.

The four units the BEMO server needs to communicate with are:

1. The Occupancy Sensor
2. The Accounting Information System
3. The sensors and meters installed in the buildings
4. The Building Management System (BMS)

For the Occupancy Sensor a software module APU Gateway Interface has been developed and integrated together with a second software module, the APU gateway in the BEMO Control and Management Platform.
An AIS gateway software module has been developed for accounting data integration into the BEMO system.
The BMS provides the energy data obtained from the sensors and meters installed in the building directly to the BEMO system, but a great effort has been done in the design and development of the BMS gateway software module.

Thanks to these software modules, the work performed within this task has led to the final interaction and communication architecture between the different units composing the BEMO system as well as the BEMO system and the external world via BEMO server.

6 REFERENCES

[1] Virtual Machine host and guest. Online: http://searchservervirtualization.techtarget.com/tip/Host-and-guest-virtual-machine-Definitions, March 19 2015.

[2] Dell Precision T3600. Online: http://www.in.tum.de/fileadmin/user_upload/RBG/Datenblaetter/Dell/Workstation/Dell-Precision-T3600.pdf, March 19 2015.

[3] VirtualBox. Online: http://en.wikipedia.org/wiki/VirtualBox, March 19 2015.

[4] Virtual Machine. Online: http://en.wikipedia.org/wiki/Virtual_machine, March 19 2015.

[5] Accounting Information System. Online: http://en.wikipedia.org/wiki/Accounting_information_system, March 19 2015.

[6] Super CSV. Online: http://super-csv.github.io/super-csv/index.html, March 19 2015.

[7] BACnet. Online: http://www.bacnetinternational.org/files/Homepage-Introduction%20to%20BACnet/BACnet%20Introduction%20-%20V3-1.pdf, March 19 2015.

[8] Siemens Desigo Sx Open. Online: www.siemens.com/download?A6V10445033, March 19 2015.

[9] BACnet stack library. Online: http://bacnet.sourceforge.net/, March 19 2015.

[10] BACnet4J. Online: http://sourceforge.net/projects/bacnet4j/, March 19 2015.

[11] BACnet/IP Broadcast Management Device. Online: http://www.ccontrols.com/pdf/bbmd.pdf, March 19 2015.

[12] OPC. Online: https://opcfoundation.org/about/what-is-opc/, March 19 2015.

[13] OPC Introduction. Online: http://www.ni.com/white-paper/7451/en/, March 19 2015.

[14] Tac Vista OPC server. Online: http://195.70.43.12/Vista/tacvistaserveropcserver101157.pdf, March 19 2015.

[15] OPC Programming with Java. Online: http://www.opcconnect.com/java.php, March 19 2015.

[16] JEasyOPC. Online: http://sourceforge.net/projects/jeasyopc/, March 19 2015.

[17] OpenSCADA Utgard. Online: http://openscada.org/projects/utgard/, March 19 2015.

[18] Virtual Private Network. Online: http://en.wikipedia.org/wiki/Virtual_private_network, March 19 2015.

[19] Firepass. Online: http://www.f5.com/pdf/products/firepass-overview.pdf, March 19 2015.

[20] Cyberoam. Online: http://kb.cyberoam.com/default.asp?id=2270, March 19 2015.

[21] Virtual Network Computing. Online: http://en.wikipedia.org/wiki/Virtual_Network_Computing, March 19

APPENDIX A: CONFIGURATION OF A VIRTUAL MACHINE FOR THE BEMO SERVER ON DAPP 1764

This appendix describes in detail the procedure followed to configure a Virtual Machine for the BEMO server on the workstations physically installed in the pilot. This procedure has been developed and tested and refined before on the D'Appolonia machine: DAPP 1764.

Creation of the Virtual Machine on DAPP1764

- Access to php Virtual box interface on http://10.10.5.50/phpvirtualbox/
- Login with username and password
- Click 

- In the wizard for the creation of a new virtual machine set the following parameters and click "Next"

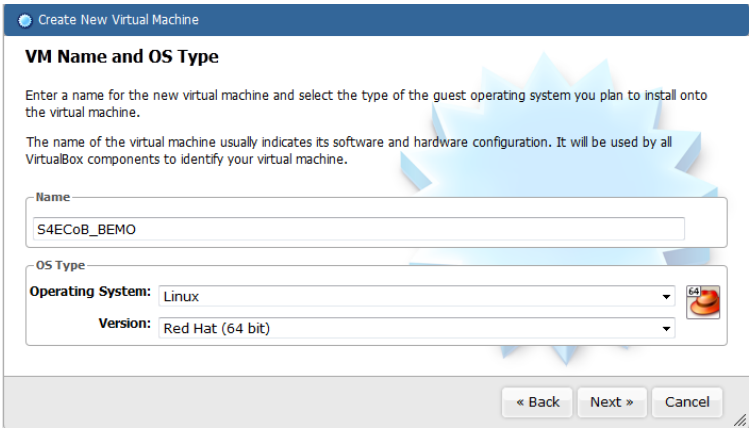


Figure 14 - VM name and OS type

- Select memory corresponding to 2048 MB and click "Next"

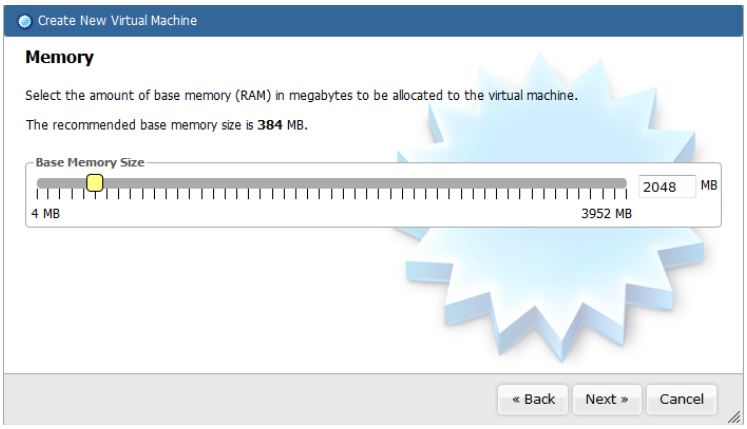


Figure 15 – Memory

- Create a new hard disk and click "Next"

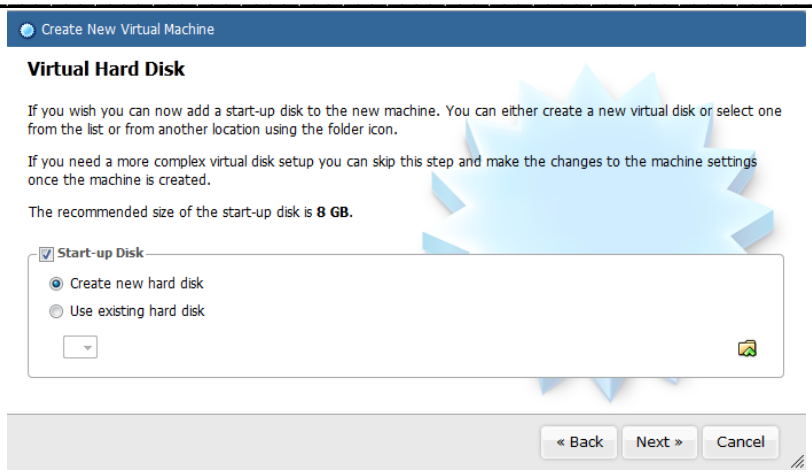


Figure 16 - Virtual hard disk

- Select the type of virtual disk corresponding to VDI and click "Next"

Figure 17 - Virtual disk creation wizard

- Create a fixed size hard disk of 120GB, click "Next" and then "Create"

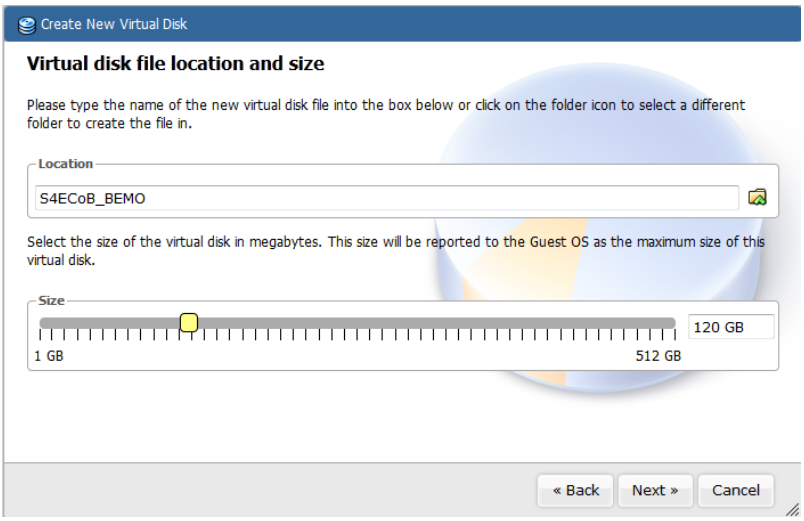


Figure 18 - Virtual disk file location and size

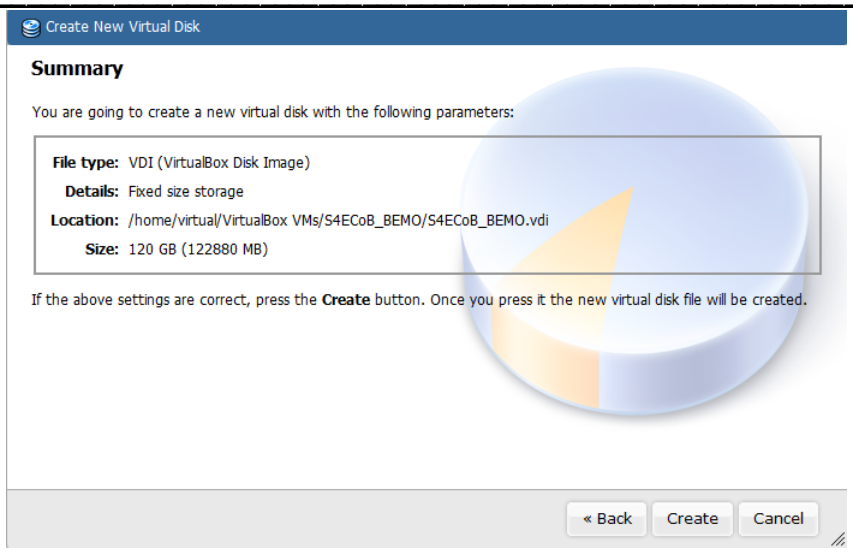


Figure 19 – Creation of new virtual disk summary

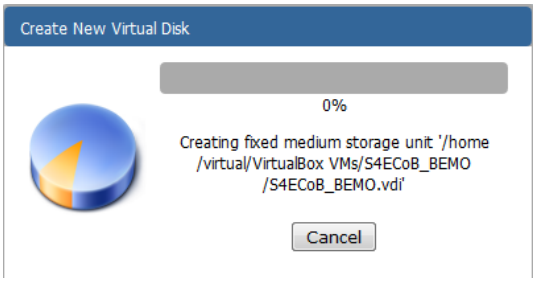


Figure 20 - Creation of the virtual disk

- Once the disk is created, create the virtual machine by clicking “Create”

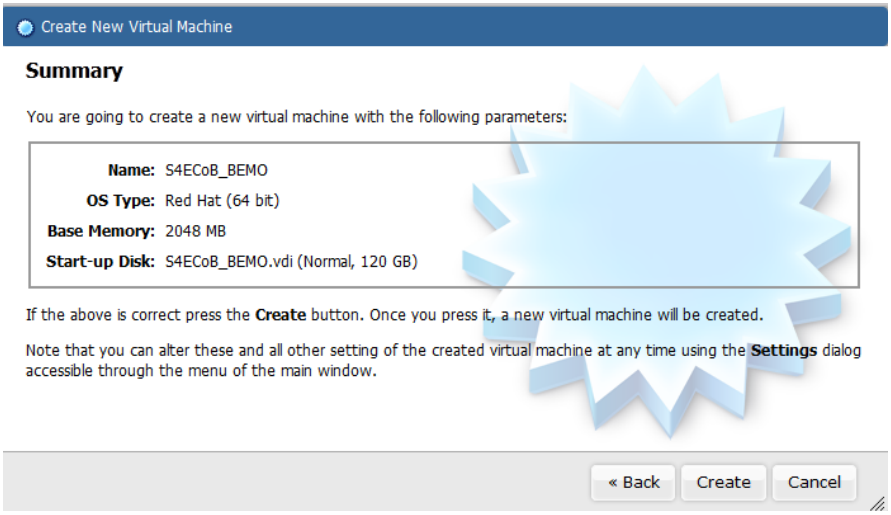


Figure 21 - creation of new virtual machine summary

- Now the virtual machine should show up in the list

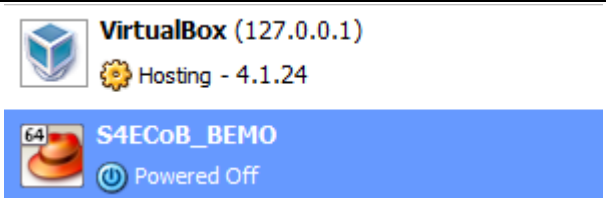


Figure 22 - List of VM

- In the Settings select 2cpus
-

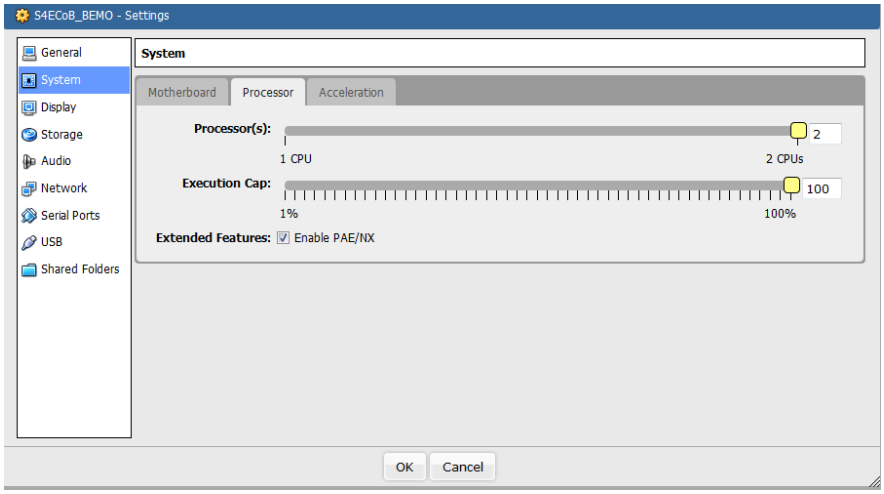


Figure 23 - System processor

- To allow the virtual machine network connection in Settings > Network perform the following settings

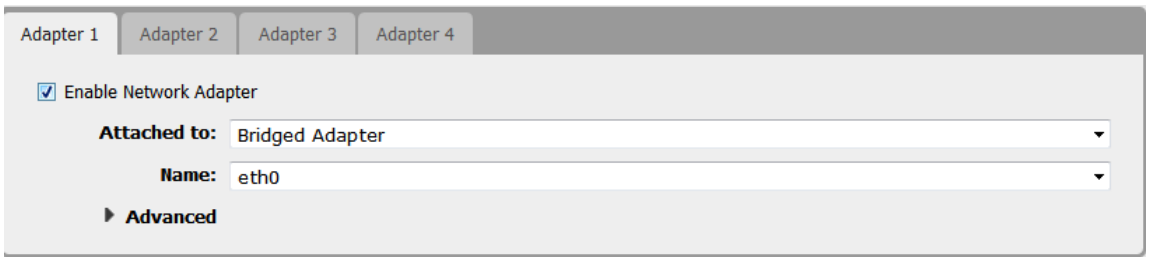


Figure 24 - Network adapter

Installation of CENTOS 6.3 on the Virtual Machine

- Copy the CentOS 6.3 DVD 1 named "CentOS-6.3-x86_64-bin-DVD1.iso" from J:\Proj\P\P-INN\eng\INN-ICT Group\Software\OS\CentOS 6.3 x86_64bit to the home folder of the user "virtual", for example by using WinSCP

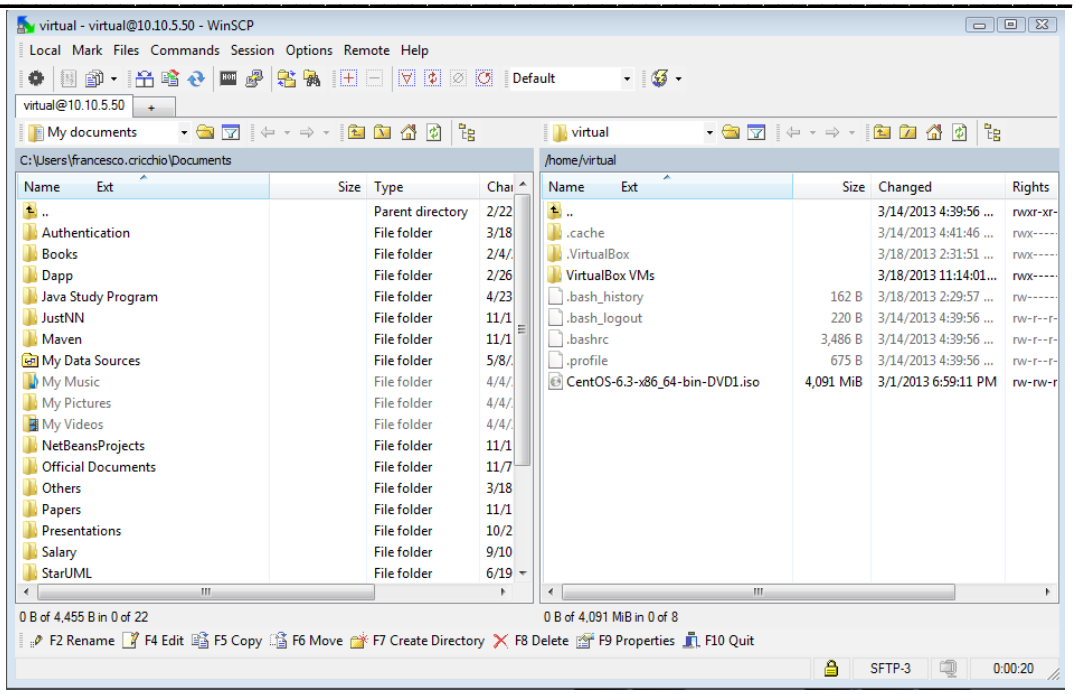


Figure 25 - Centos DVD WinSCP

- In Virtual Box set the CentOS 6.3 DVD as boot source in the virtual machine settings, select the Live CD/DVD checkbox

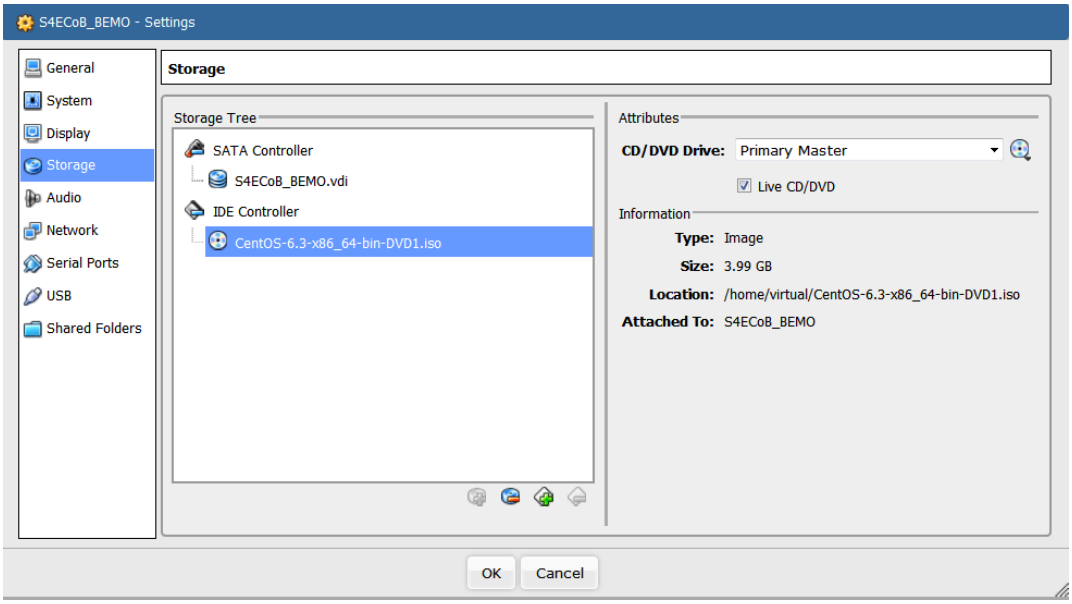


Figure 26 - Storage

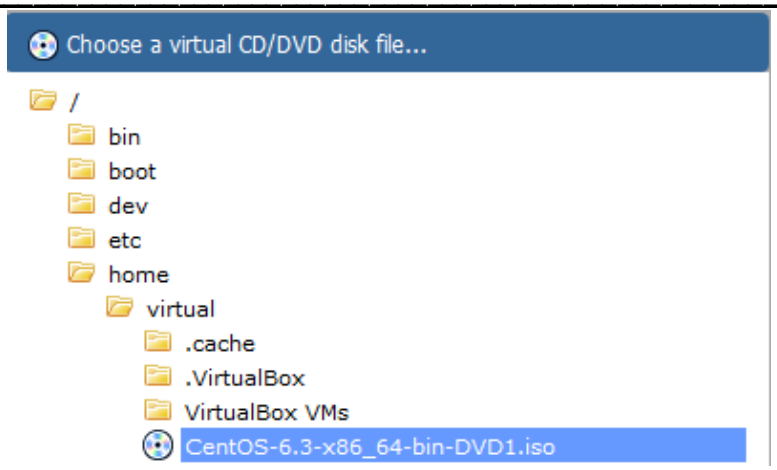
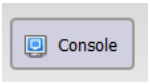


Figure 27 - Choose of a virtual CD/DVD disk file

- Run the virtual machine
- The CentOS 6.3 installation begins, interact with the machine through the remote desktop console 

- Adjust the resolution of the remote desktop console to 1440x900. To access the resolution menu disconnect the desktop once and then reconnect

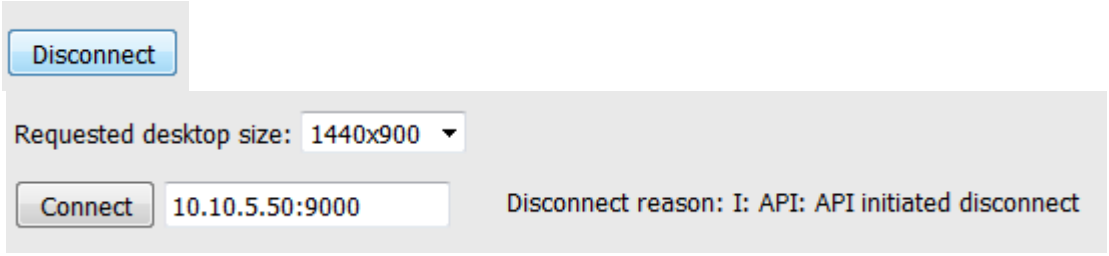


Figure 28 - Adjust resolution

- Once CentOS DVD has started select **Install or upgrade an existing system**
-

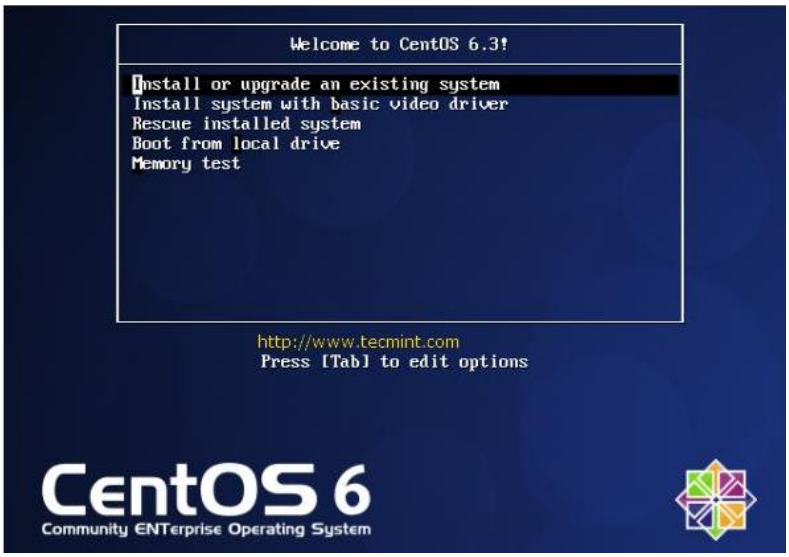


Figure 29 - Install or upgrade an existing system

- When prompted for disk checking **Skip** the procedure

Figure 30 - Skip media test

- CentOS 6.3 Welcome Screen press **Next**

Figure 31 - Centos 6 welcome screen

- Language Selection, select **English**, if problem with mouse use the Tab button

Figure 32 - Language selection

- Select **Italian keyboard**, if problem with mouse use the Tab button

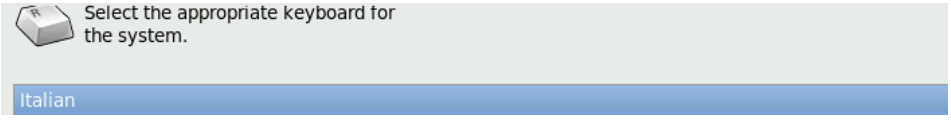


Figure 33 - Select keyboard

- Select **Basic Storage Device** (the hard drive is attached locally)

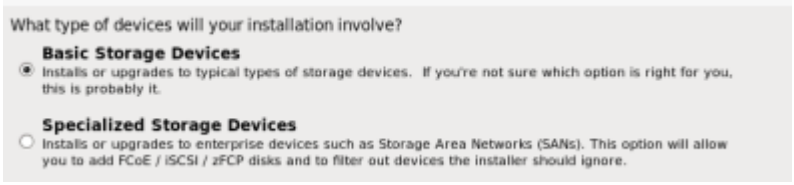


Figure 34 - Type of devices

- You may get **Storage Device** warning, you can click **Yes**, discard any data button to **Continue**.

Figure 35 - Storage device warning

- Give a hostname to the server and click, provide S4ECoB_BEMO, then click on **Configure Network**

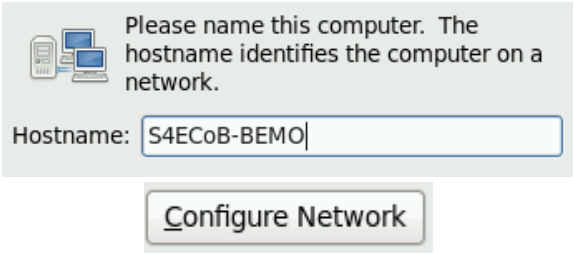


Figure 36 - Provide the hostname

- Set up a **wired connection** and rename it "eth0"

Figure 37 - Set up a wired connection

- Set up the **IPv4 Settings** for the wired connection as following

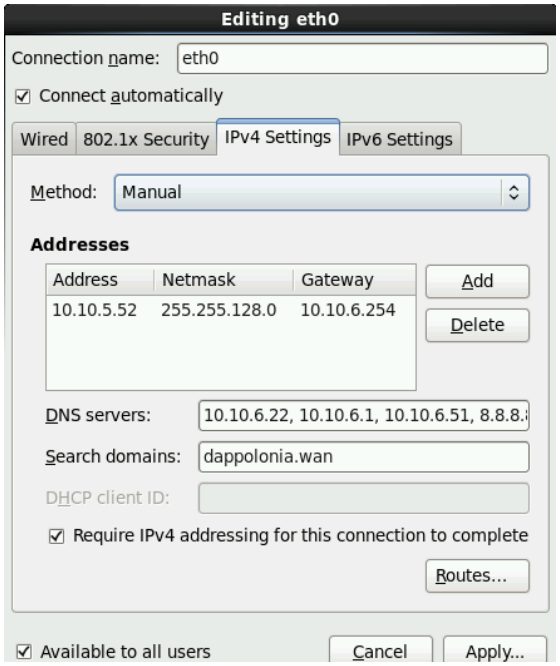


Figure 38 - Set up the IPv4 settings

- Set the time zone to **Europe/Rome**

Figure 39 - Set the time zone

- Set the root password

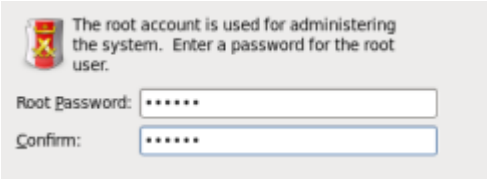


Figure 40 - Set the root password

- Select the type of partition; choose **Use All Space**

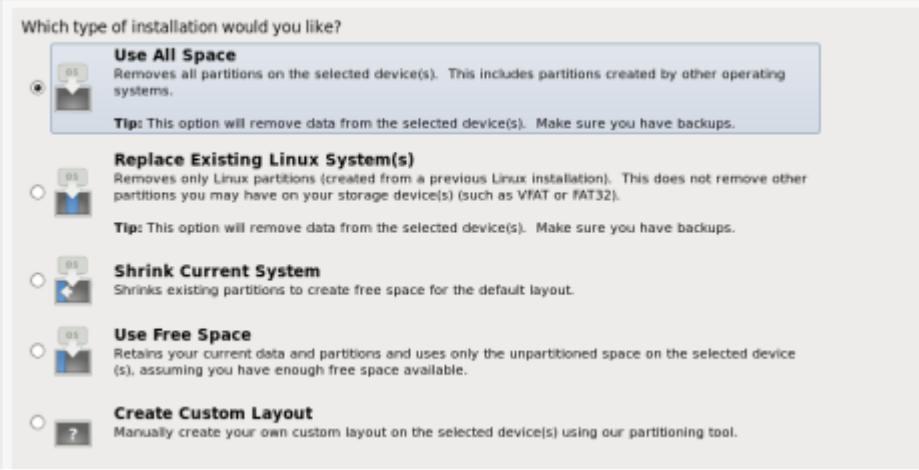


Figure 41 - Type of installation

- Select **Desktop Installation**

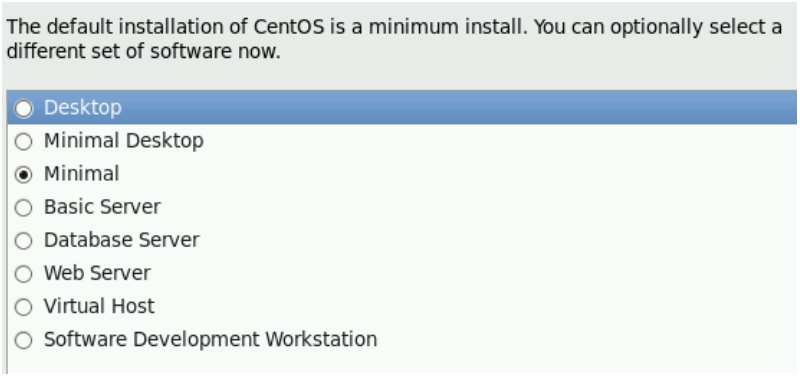


Figure 42 - Set software

- The installation will now be performed, wait until finished

Figure 43 - Installation

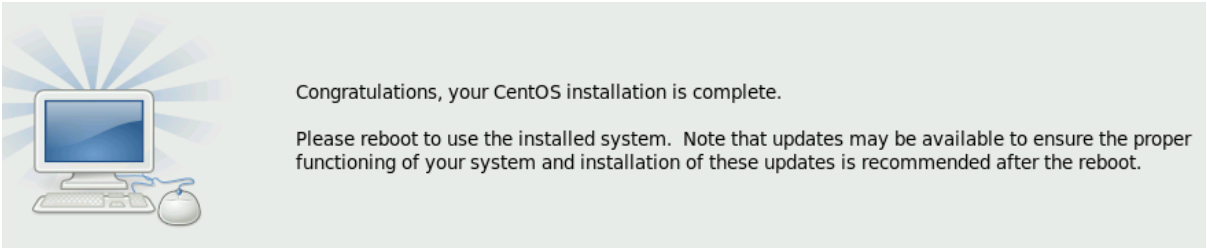


Figure 44 - Installation complete

- In Virtual Box move hard disk as boot source before CD/DVD-ROM

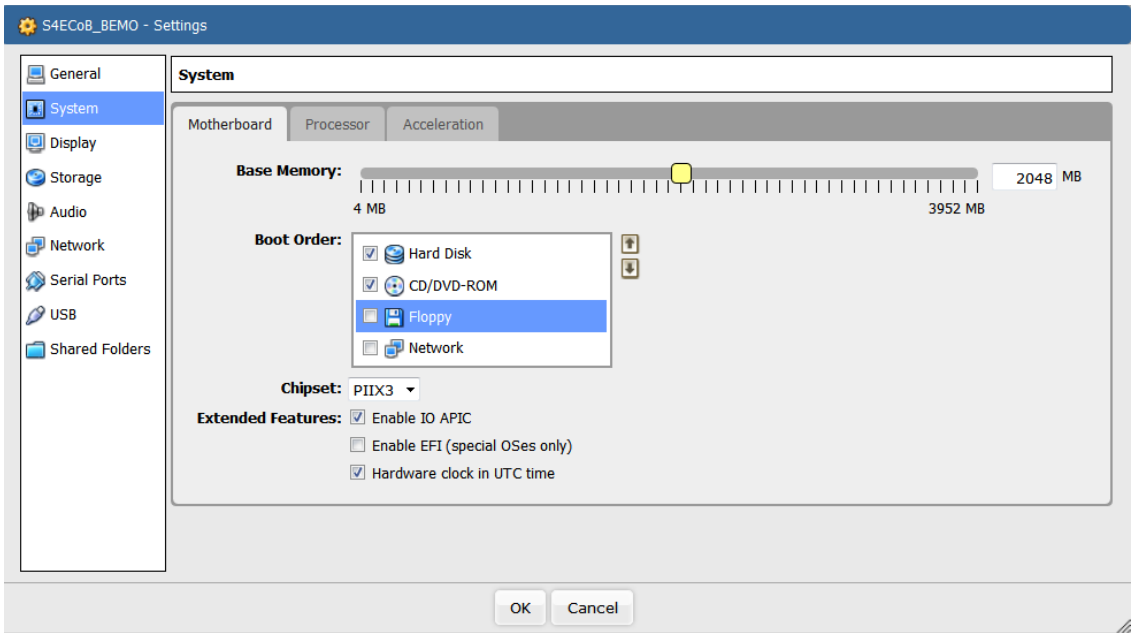


Figure 45 - Set boot order

- Reboot the system
- At first boot, when asked create the S4ECoB-user with a password

Figure 46 - User creation

- Set date and time of the system by selecting automatic configuration over the network

- Enable kdump that will reboot the system

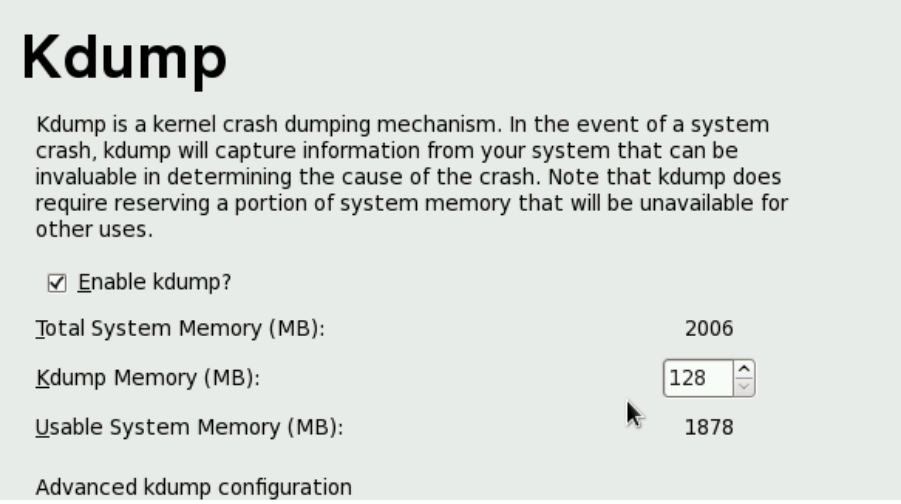


Figure 47 - Kdump

- In case of network problems, try to disconnect/reconnect the network eth0

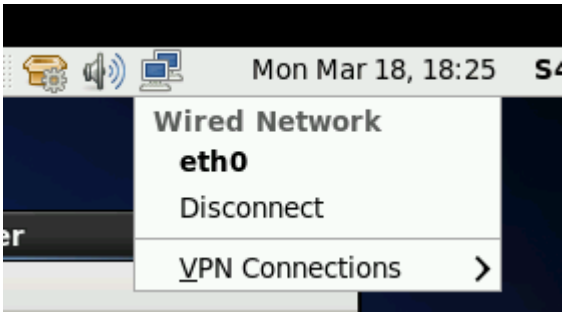


Figure 48 - Network problem

APPENDIX B: INSTALLATION OF REQUIRED SOFTWARE ON THE GUEST MACHINE

This appendix describes in detail the procedure to be followed to install the required software on the guest machine BEMO server.

Installation of openssh on the virtual machine

- # yum install openssh

Installation of Filezilla on the virtual machine

- # yum install filezilla

Installation of VNC on the virtual machine

- # yum -y install vnc-server
- Configure vncuser
  # su –
  useradd S4ECoB-user
  passwd S4ECoB-user
  Set password
- Edit VNC Configuration File:
  vi /etc/sysconfig/vncservers
  VNCSERVERS="1:S4ECoB-user" //Uncomment 2 last lines, and edit your VNC user
  VNCSERVERARGS[1]="-geometry 800×600" // Save and exit
- Edit hosts file:
  vi /etc/hosts
  and add the correct name of the host name, for example S4ECoB-BEMO
  127.0.0.1 localhost localdomain.localhost S4ECoB-BEMO
- Start VNC Server:
  # vncserver
  # chkconfig vncserver
- More information on http://wiki.centos.org/HowTos/VNC-Server

Installation and Configuration of MYSQL on the virtual machine

Installation

- Install MySQL
  # yum install mysql-server mysql php-mysql

Configuration

- Set the MySQL service to start on boot
  # chkconfigc --levels 235 mysqld on
- Start the MySQL service
  # service mysqld start
- Log into MySQL
  # mysql -u root
- Set the root user password for all local domains
  SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new-password');
  SET PASSWORD FOR 'root'@'127.0.0.1' = PASSWORD('new-password');
- Drop the Any user *(maybe later)*
  DROP USER ''@'localhost';
  DROP USER ''@'localhost.localdomain';
- Exit MySQL
  exit

Install JDK 7

- Let's start out by installing what we will need to have on the Linux box from it's own package repository. As root (or sudo) do:

  # sudo yum install jpackage-utils

- Now you can download the JDK here http://www.oracle.com/technetwork/java/javase/downloads/index.html and install it, for example

  #sudo yum localinstall jdk-7u51-linux-x64.rpm

- Now you should have JDK (and JRE) installed on your machine. Now we make the Oracle JDK as default by using the command `alternative`:

  # alternatives --install /usr/bin/javac javac /usr/java/latest/bin/javac 20000

  # alternatives --install /usr/bin/jar jar /usr/java/latest/bin/jar 20000

  # alternatives --install /usr/bin/java java /usr/java/jdk1.7.0_17/jre/bin/java 20000

  # alternatives --install /usr/bin/javaws javaws /usr/java/jdk1.7.0_17/jre/bin/javaws 20000

- Check that the Oracle JDK is set as default

  # /usr/sbin/alternatives --config java

/usr/sbin/alternatives --config javac

/usr/sbin/alternatives --config jar

/usr/sbin/alternatives --config jawaws

- Check that the Java version is correct

  # java –version

  should return

  # java version 1.7.0_51

- Set the JAVA_HOME path. This is where we installed the JDK above. To set the JAVA_HOME for users, we add this to the user ~/.bashrc or ~/.bash_profile of the user. We can also add it /etc/profile.d/java_home.sh such that is loaded at login for all users.

  # vi /etc/profile.d/java_home.sh

  JAVA_HOME=/usr/java/jdk1.7.0_51

  export JAVA_HOME

  PATH=$JAVA_HOME/bin:$PATH

  export PATH

- Once you have added the above to ~/.bash_profile or ~/.bashrc, you should log out, then log back in and check that the JAVA_HOME is set correctly.

  # echo $JAVA_HOME

  # /usr/java/jdk1.7.0_51

Install Glassfish 4 CentOS

Download and Install the GlassFish 4 Server

- You can download both the GlassFish Server Open Source Edition 4 and Oracle GlassFish Server 4 at http://glassfish.java.net/
- Once you have downloaded the desired file, move (mv) or copy (cp) the file to /usr/share/glassfish-4.zip (or /usr/share/ogs-4.zip for Oracle GlassFish).

  # mv glassfish-4.zip /usr/share/glassfish-4.zip

- Change to the /usr/share directory and unzip the file:

  # cd /usr/share
  # unzip -q glassfish-4.zip

  The unzip will create the following directory: /usr/share/glassfish4 Note: Both GlassFish editions will create the same directory when unzipped: glassfish4

Running GlassFish as a Service

To run GlassFish as a service and enable start up at boot, we'll now create a Start/Stop/Restart script.

- We'll create the script as /etc/init.d/glassfish, make the script executable, and then add our new glassfish service to chkconfig.
- Create our glassfish script:

```
# cd /etc/init.d
# vi glassfish
#!/bin/bash
# description: Glassfish Start Stop Restart
# processname: glassfish
# chkconfig: 234 20 80
  JAVA_HOME=/usr/java/jdk1.7.0_51
  export JAVA_HOME
  PATH=$JAVA_HOME/bin:$PATH
  export PATH
  GLASSFISH_HOME=/usr/share/glassfish4/glassfish

   case $1 in
   start)
   sh $GLASSFISH_HOME/bin/asadmin start-domain domain1
   ;;
   stop)
   sh $GLASSFISH_HOME/bin/asadmin stop-domain domain1
   ;;
   restart)
   sh $GLASSFISH_HOME/bin/asadmin stop-domain domain1
   sh $GLASSFISH_HOME/bin/asadmin start-domain domain1
   ;;
   esac
   exit 0
```

- If you do not set the JAVA_HOME and PATH in the GlassFish script, when you attempt to start the GlassFish server it will complain it cannot find Java with the following:

  error: /usr/share/glassfish4/glassfish/bin/asadmin: line 19: exec: java: not found

- Now, make the script executable and add it to our chkconfig so it starts at boot.

chmod 755 glassfish
chkconfig --add glassfish
chkconfig --level 234 glassfish on

- We should now be able to Start, Stop, and Restart GlassFish as a service.

  Start GlassFish:

  # service glassfish start
  Waiting for domain1 to start .....
  Successfully started the domain : domain1
  domain Location: /usr/share/glassfish3/glassfish/domains/domain1
  Log File: /usr/share/glassfish4/glassfish/domains/domain1/logs/server.log
  Admin Port: 4848
  Command start-domain executed successfully.

  Stop GlassFish:

  # service glassfish stop
Waiting for the domain to stop ....
Command stop-domain executed successfully.

Check Homepage

- You should now see the GlassFish default home page at http://yourdomain.com:8080 or http://YourIP:8080

Error 4848 port is already used

- If the error "4848 port is already used" probably there is an error with the hostname You should

  vi /etc/hosts

  and add the correct name of the host name, for example

  127.0.0.1 localhost localdomain.localhost S4ECoB-BEMO

APPENDIX C: APU CONFIGURATION

This appendix describes in detail the procedure to be followed to configure the APU on the BEMO server.

Configuration of APU IP Address from Host Machine DAPP1764 to be connected to virtual machine S4ECoB

1. Connect APU to DAPP1764 through a cross network cable

2. Set up the IP address of the Host machine DAPP1764 to 10.1.155.7 and set up the IP address of the virtual machine to 10.1.155.1.

   The two machines should have IP addresses that belong to the same network
3. Login to the APU
   # ssh root@10.1.155.3
   10.1.155.3 is the *stickerIP* physically applied on the APU
   This IP address should belong to the same network of the host and virtual machine.
   Type the password.
4. Set the APU IP address to 10.10.5.57 by using the following command

   root@apu:~# netconf2eeprom 0 <apu ip> <netmask> <network> <gateway> <dns> <bemo ip>

   For D'Appolonia network will be

   root@apu:~# netconf2eeprom 0 10.10.5.57 255.255.128.0 10.10.5.0 10.10.5.52 10.10.5.52 10.10.5.52

   On success the utility will just execute without further output. If a parameter error occurs please try again without the last IP address in the command line

5. Check if the settings change succeeded

   root@apu:~# eeprom2netconf

   The output of this command should look like this:

   auto eth0
   iface eth0 inet static
   address 10.10.5.57
   netmask 255.255.128.0
   network 10.10.5.0
   gateway 10.10.5.52
   dns-nameserver 10.10.5.52
   bemo-server 10.10.5.52

6. Reboot the APU by typing

   root@apu:~# reboot

   in order to enable the new network configuration

7. Reset the Host IP address to the original 10.10.5.50 and the S4ECoB_BEMO virtual machine with IP address 10.10.5.52. In this way the machine is inserted back into the network

8. Connect the APU through an ordinary network cable to the network

9. Try to ping the APU from the S4ECoB_BEMO virtual machine with IP address 10.10.5.52
   ping 10.10.5.57

Configuration of APU IP Address from Host Machine DAPP1764 to be connected to SEA network

The purpose is to set the IP address of the APU to one of the IP addresses of SEA network such that it can be connected to another machine of the SEA network. The parameters of the SEA network are the following

I.P. 10.1.155.1
I.P. 10.1.155.2
I.P. 10.1.155.3
I.P. 10.1.155.4
I.P. 10.1.155.5
I.P. 10.1.155.6
I.P. 10.1.155.7
I.P. 10.1.155.8
I.P. 10.1.155.9
I.P. 10.1.155.10

SUBNET: 255.255.248.0
GTW: 10.1.159.241
DNS: 10.1.88.1 / 10.3.88.1

Configuration

1. Connect APU to DAPP1764 through a cross network cable. The IP address of virtual machine should be 10.10.5.52

2. Login to the APU
   # ssh root@10.10.5.57
   Type the password.

3. Set the APU IP address to one of SEA network (10.1.155.3) by using the following command

   root@apu:~# netconf2eeprom 0 <apu ip> <netmask> <network> <gateway> <dns> <bemo ip>

   For SEA network this will be

   root@apu:~# netconf2eeprom 0 10.1.155.3 255.255.248.0 10.1.155.0 10.1.159.241 10.1.88.1 10.1.155.1

   On success the utility will just execute without further output. If a parameter error occurs please try again without the last IP address in the command line

4. Check if the settings change succeeded

   root@apu:~# eeprom2netconf

   The output of this command should look like this:

   auto eth0
   iface eth0 inet static
   address 10.1.155.3
   netmask 255.255.248.0
   network 10.1.155.0
   gateway 10.1.159.241
   dns-nameserver 10.1.88.1
   bemo-server 10.1.155.1

5. Reboot the APU by typing

   root@apu:~# reboot

   in order to enable the new network configuration

Test the settings

1. Connect the APU to DAPP1764 with a cross cable

2. Set up the IP address of the Host machine DAPP1764 to 10.1.155.7 and the IP address of virtual machine to 10.1.155.1

3. Login to the APU
   # ssh root@10.1.155.3
   Type the password.

4. If the connection is successful the test is positive

APPENDIX D: PROCEDURE TO INSTALL THE GATEWAY ON A CENTOS 6.3 VIRTUAL MACHINE

This appendix describes in detail the procedure to be followed to install the APU gateway software module on the CentOS Virtual Machine BEMO server.

Preliminary settings

1. Create a directory gateway
   # mkdir gateway
2. Copy the gateway and the gateway simulator on the machine by using for example WinSCP on the gateway directory

Installation of prerequisites

3. Check if dbus daemon is running
   # ps -ef | grep dbus | grep -v grep
4. Install the following packages:
   # yum install http://ftp-stud.hs-esslingen.de/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
   # yum install http://software.freivald.com/el/6/x86_64/os/software.freivald.com-2.0.0-0.el.noarch.rpm
   # yum update

Gateway Installation

5. Now (after the copy of the gateway on BEMO-server virtual machine) the Gateway RPMs can be installed (necessary dependencies should install automatically):
   # yum install apuGateway-0.8.4-x86_64.rpm and/or # yum install apuGatewayGui-0.8.4-x86_64.rpm

Enabling connection with APUs

6. For the APU to be able to establish a network connection to the Gateway the TCP port 6789 has to be enabled in the CentOs firewall: use the configuration utility to add the port: System>Administration>Firewall>Other Ports
7. The APU is currently configured for SEA Network and therefore to connect to the IP address 10.1.155.3 that is the *stickerIP* physically applied on the APU. You have to configure the network interface of the BEMO server the APU is connected to using this static address
8. Now the Gateway can be started from a terminal (as user) without or with GUI:

   # apuGateway
   or
   # apuGatewayGui

9. Connect the APU either directly or over a switch to the BEMO server using the single RJ45 port (on the same side of the APU as the power connector).
10. Now the APU can be connected to power. **Make sure to use the 5V power supply!**
11. After a short while the APU should be showing up in the Gateway. There are no audio services configured to start up automatically yet. So all you will be seeing are the heartbeat messages arriving from the APU. If it is possible for you the enable a remote access to this BEMO installation I may be able to login to the APU and configure some additional services so some more messages will be generated.
12. For testing the Gateways D-Bus interface you can of course still use the simulator. To compile the simulator and the C++ example you need some additional dev packages:

    # yum install gcc-c++ qt-devel

    Then follow the instructions in the readme file.

If qmake command not found, set QDIR environment variable. For doing this you can modify ~/.bash_profile of the user:

```
# vi ~/.bash_profile

QDIR=/usr/lib64/qt4

export QDIR

PATH=$QDIR/bin:$PATH

export PATH
```

Once you have added the above to ~/.bash_profile, you should log out, then log back in and check that the QDIR is set correctly.

```
# echo $QDIR

/usr/lib64/qt4
```

13. And for the Python example to run:

    # yum install dbus-python pygobject2

The examples are able to connect to the Gateway or the simulator (which of course can not run simultaneously).