BaaS

Building as a Service

FP7-ICT-2011-6: ICT Systems for Energy Efficiency
Small or Medium-scale Focused Research Project
Grant Agreement No. 288409

Deliverable D3.6:

Development guide

Deliverable Version:	D3.6, v.1.1
Document Identifier:	baas_wp3_d3.6_developmentguide_1.1
Preparation Date:	October 31, 2013
Document Status:	Final
Author(s):	José L. Hernández and Susana Martín (CARTIF), Martin Floeck and Mischa Schmidt (NEC), Kai Mo (UCC)
Dissemination Level:	PU - Public

Project funded by the European Community in the 7th Framework Programme

ICT for Sustainable Growth

Deliverable Summary Sheet

Deliverable Details

Type of Document:	Deliverable
Document Reference #:	D3.6
Title:	Development guide
Version Number:	1.1
Preparation Date:	October 31, 2013
Delivery Date:	October 31, 2013
Author(s):	José L. Hernández and Susana Martín (CARTIF), Martin Floeck and Mischa Schmidt (NEC), Kai Mo (UCC)
Document Identifier:	baas_wp3_d3.6_developmentguide_1.1
Document Status:	Final
Dissemination Level:	PU - Public

Project Details

Project Acronym:	BaaS
Project Title:	Building as a Service
Project Number:	288409
Call Identifier:	FP7-ICT-2011-6
Call Theme:	ICT Systems for Energy Efficiency
Project Coordinator:	Fundacion Cartif (CARTIF)
Participating Partners:	Fundation Cartif (CARTIF, ES); NEC Europe Ltd. (NEC, UK); Honeywell, SPOL, S.R.O (HON, CZ); Fraunhofer-Gesellschaft zur Förderung der Angewandten Forschung e.V. (Fraunhofer, DE); Technical University of Crete (TUC, GR); University College Cork, National University of Ireland, Cork (UCC-IRU, IE) Dalkia Energia y Servicios (DALKIA, ES)
Instrument:	STREP
Contract Start Date:	May 1, 2012
Duration:	36 Months

Deliverable D3.6: Short Description

The BaaS system is a complex system composed of several entities interworking together in order to offer comprehensive and integrated energy management services. In this document, the technologies and methodologies for the development of the platform are specified, forming the basis for the implementation of the BaaS system. Moreover, the installation of the system should follow the guidelines specified in the deliverable so as to be able to replicate the deployment in other environments, e.g. if an Energy Service Company would like to deploy the system to additional buildings as well. Moreover, as each building is unique in terms of the BMS, the available ICT systems, or other parameters, this development guide includes guidelines for the flexible adaptation of the BaaS system to different BMS protocols, DWH vendors or external data services.

Keywords: BaaS System, Spring Dynamic Modules, DataWarehose, Bim Server, BMS, protocol, DACM, DC, deployment.

Deliverable D3.6: Revision History

Version:	Date:	Status:	Comments
0.1	22/05/2013	Draft	CARTIF: Skeleton and responsibilities
0.2	30/09/2013	Draft	CARTIF, NEC: Review of skeleton and contents
0.3	17/10/2013	Draft	CARTIF: Section 3.1.1, 3.1.2, 4.2.2
0.4	18/10/2013	Draft	CARTIF: Section 3.3, 3.2.2, 3.6
0.5	21/10/2013	Draft	NEC: Sections 2, 3.1.3, 3.4, 4.1, 4.3, 4.4, 5.1
0.5.1	22/10/2013	Draft	NEC: Addition of information in section 4.3 and review CARTIF: Review of the current content.
0.6	23/10/2013	Draft	CARTIF: Section 3.3
0.7	25/10/2013	Draft	CARTIF, NEC: Review of current sections
0.8	28/10/2013	Draft	CARTIF: Sections 1, 3.5 and 6 NEC: Review of content and inclusion of DC and DACM properties
0.9	28/10/2013	Draft	NEC: Cloud deployment concepts
0.9.1	29/10/2013	Draft	UCC: Section 4.2.1
0.9.2	29/10/2013	Under review	CARTIF: Merge the contributions
1.0	31/10/2013	Final	CARTIF: Merge the comments from the review
1.1	04/11/2013	Final	NEC, CARTIF: Amendment of some sections

information contained herein is error-free, or up to date, nor makes warranties, express, implied, or statutory, by publishing this document.

Table of Contents

1 INTRODUCTION 1

1.1 PURPOSE 1
1.2 CONTRIBUTION TO THE SCIENTIFIC OBJECTIVES 1
1.3 RELATIONSHIP WITH OTHER WPS 2
1.4 CONTRIBUTION FROM PARTNERS 2

2 SYSTEM ARCHITECTURE 4

3 TECHNOLOGIES INTEGRATED 5

3.1 OSGI FRAMEWORK 5
3.1.1 Spring Dynamic Modules 6
3.1.2 Communication based on OSGi events 7
3.1.3 OSGi framework used in BaaS Middleware 8
3.2 DATA WAREHOUSE 9
3.2.1 State of the art 9
3.2.2 Hibernate 10
3.3 BIM SERVER 11
3.3.1 Communication with BIM Server 11
3.4 BUILDING MANAGEMENT SYSTEM 12
3.4.1 BMS available protocols 12
3.4.2 Communication with the Building Management System 12
3.5 EXTERNAL SERVICES 12
3.5.1 Communication between CLL and external services 13
3.6 GRAPHICAL USER INTERFACE 13
3.6.1 Framework for the development 13
3.6.2 Behaviour of GWT 14

4 DEPLOYMENT GUIDELINE 15

4.1 SYSTEM REQUIREMENTS 15
4.2 INSTALLATION OF THE DATA SOURCES 15
4.2.1 DataWarehouse 16
4.2.2 BIM Server 19
4.3 DEPLOYMENT OF THE OSGI SERVER 22
4.3.1 Installation of the DACM 22
4.3.2 Installation of the DC 24
4.4 DEPLOYMENT OF THE APO SERVICES 25
4.5 CONSIDERATIONS REGARDING CLOUD DEPLOYMENT 25

5 DEVELOPMENT GUIDELINE 27

5.1 HOW TO ADAPT BAAS TO DIFFERENT BMS PROTOCOL 27
5.2 HOW TO ADAPT THE DATABASE 27
5.3 EXTERNAL SERVICES 28

6 CONCLUSIONS 29

REFERENCES 30

List of Figures

Figure 1: Middleware, Functional Architecture....4
Figure 2: OSGi architecture [4]....5
Figure 3: Service registry in the OSGi framework [4]....6
Figure 4: OSGi Event Mechanism....8
Figure 5: Configuration of Registering Listener....17
Figure 6: Relational Model of TUC DWH Schema....18
Figure 7: SQL for Create IfcBuilding....19
Figure 8: ETL Process....19
Figure 9: Configuration of the BIM Server....20
Figure 10: Deployment of the BIM Server....21
Figure 11: Setup BIM Server....21
Figure 12: Functionalities in the BIM Server client....22
Figure 13: Uploading the ifc file to the BIM Server....22
Figure 14: BaaS System deployment scheme....26

List of Tables

Table 1: Comparison among Oracle, PostgreSQL and MySQL 10

Abbreviations and Acronyms

API	Application Programming Interface
APO	Assess, Predict, Optimize
BaaS	Building as a Service
BMS	Building Management System
BIM	Building Information Model
CLL	Communication Logic Layer
CPU	Central Processing Unit
DACM	Data Acquisition Control Manager
DAO	Data Access Object
DC	Domain Controller
DWH	Data WareHouse
ESCO	Energy Service COmpany
ETL	Extract, Transform, Load
GUI	Graphical User Interface
GWT	Google Web Toolkit
HQL	Hibernate Query Language
HTTP	Hypertext Transfer Protocol
ICT	Information and Communications Technology
IDE	Integrated Development Environment
IoC	Inversion of Control
JAR	Java ARchive
JAX	Java API for XML
JVM	Java Virtual Machine
KPI	Key Performance Indicator
OLAP	On-Line Analytical Processing
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OSGi	Open Services Gateway initiative
RAM	Random Access Memory
SDM	Spring Dynamic Modules
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
XML	eXtended Mark-up Language

URL	Uniform Resource Locator
VPN	Virtual Private Network
WP	Work Package
WS	Web Service

Executive Summary

The development guide provided in the current deliverable specifies how the BaaS system is implemented and deployed, as well as how its functionalities can be extended. First, the technological foundations of the BaaS platform are introduced. Java/OSGi was chosen as the general environment as it inherently supports service-oriented architectures. In addition, several proven standard technologies, e.g. Hibernate, web services, BACnet, are employed in order to provide connectivity to external entities such as the BMSs, BIM servers, etc. Secondly, the specification of the deployment of the whole system is detailed. The BaaS system is composed of several entities which have to be integrated and configured in order to allow the communication between them. Therefore, a installation guideline is provided which describes how the BaaS system is set up and how the various elements of the system (DC, DACM, BIM, DWH, etc.) are connected and depend on each other. This includes the libraries needed, the directory structure and properties configuration. Last, the BaaS system must be scalable and replicable. The development guide describes possible adaptations as for instance the BMS protocol, database and external services. BaaS implements some of the available BMS protocols, but the market offers many more. Thus, the BaaS system is designed in a flexible and modular manner so that new protocols for BMSs, databases, BIM servers, etc. can easily be implemented.

In relation to other tasks and deliverables in the project, this deliverables is strongly related to D3.1, D3.2, D3.3 and D3.5 in which the detail of the BaaS middleware system is given. Nevertheless, the development and integration guidelines are needed for the implementation of the architecture presented in the previous documents. Furthermore, D6.1 and D6.2 have a relationship with this deliverable because there the communication with the available data sources in the different demo sites is detailed. Moreover, D3.6 describes the connection of the data sources for storage purposes (D2.2 and D2.3) making use of the connectors. Finally, D5.1 is related to the current one as D3.6 details the technologies used by the CLL offering the interface to the APO Layer services.

1 Introduction

1.1 Purpose

This document is the integration and development guideline for the Communication Logic Layer and the interfaces with the data sources and APO Services. The audience of this deliverable is relatively wide because it is relevant to any person installing the BaaS system at a demo site or looking for the replication of the BaaS System in the future to another environment. This includes ESCOs, another project reusing the results of BaaS, an external integrator or developer wanting to duplicate or extend the BaaS functionality or a project member/partner.

The first point concerns the technologies used in the BaaS development as basis of the platform. The BaaS system is an Service Oriented Architecture [1][2][3], therefore, an OSGi framework [4] has been chosen, integrating other technologies such as Hibernate [5] and a BIM Server [6]. Also, instead of modifying the BMS of the building to support a specific 'BaaS' protocol, BaaS implements the protocols available in the demo sites [7], considering the advantages of this approach which are explained in depth in this document.

Secondly, a guideline of how to deploy BaaS is provided. This chapter addresses the steps to be followed by the integrators and testers. Thus, how to install the data sources and the components in the CLL is esplained.

Finally, different buildings likely mean different requirements. Therefore, a guide for the adaptation and extension of the BaaS functionalities is included. Any developer looking for the replication of the BaaS platform could implement the modifications into the BaaS components making use of the current development. That is, the aim of the development guide is the specification of how to adapt BaaS to different data sources such as the BMS protocol, database or external services.

1.2 Contribution to the scientific objectives

This deliverable contributes to the Scientific and Technological Objectives highlighted in the DOW as follows:

- **Scientific Objective SO1**: Building modelling and simulation for energy performance estimation and control design.
    - Integrate measured real-time data (from operation phase) along with thermal simulation models (re-design phase) to create methods to comprehensively estimate performance and compare predictions with actually measured values.
        - The installation of the BaaS system contributes to SO1 in that it enables the APO services to access integrated and harmonised data. The guideline provided in this deliverable enables the entire consortium to set up the respective BaaS system's elements at their sites.
- **Scientific Objective SO2**: Integrated Automation and Control Services**.**
    Real-time data management (aggregation of data from building and external sources) in a harmonized way (and make them available in near real time) and Implementation of scientific outcomes as Services (models, simulations and algorithms).
        - The middleware technologies established in this document harmonizes the data collected from different sources . The infrastructure detailed in the current deliverable gathers, harmonizes and stores the information.
    - Deployment possibilities in flexible Cloud infrastructure (private, public, hybrid) enabling multitude of business models.
        - One of the features of the deliverable is the description of how to deploy the platform according to the cloud premises given in D3.5.
- **Technological Objective TO1**: Data Management

    - Harmonization of near real time data for the implementation of Scientific Objectives.
        - The installation of the system includes the management of the data and its harmonization through several components included in the architecture.
- **Technological Objective TO2**: Middleware Platform: System Integration, Interoperability and Standards.
    - Development of an Open Service Middleware Platform to implement Scientific Objectives.
    - Utilization of Open Service Middleware Platform technologies for cloud-enabled interactive e-services specialized for integration of multiple trusted and un-trusted service of EMS stakeholders (common and interoperable to existing standards).
    - Development of an ontology-based service composition framework hosted in an open cloud-based environment.
    - Integration of SOA and EDA architectures
    - Distributed loosely-coupled functional design to enable hosting in flexible Cloud infrastructure.
    - Define communication interfaces between the communication platform, external ICT systems (e.g. on the Internet) and the building systems in a secure manner (Building Gateway).
        - This document addresses the installation, deployment, adaptation and extension of the BaaS system. In this way, it allows the use of the Open Service Middleware platform, the use of technologies for cloud computing, flexibility for replicability and the interfaces of accessibility to the different services.

1.3 Relationship with other WPs

This deliverable 3.6 is relevant to all WPs which deal with entities that connect or are connected to the middleware. These are WP2 (DWH), WP4 (BIM), WP5 (APO), and WP6 (demo sites) in particular. The information provided here is important for a smooth (inter-)operation of the CLL with all other entities.

1.4 Contribution from partners

During the writing of this deliverable, the partners NEC, UCC and CARTIF have participated in different activities associated to the task. In concrete, for the current deliverable the contribution of the partners are summarised as follows:

- CARTIF
    - Task leader: Proposed structure of the document, management of the distribution of work, the partners' contributions, and the internal review.
    - Participation to the weekly conference calls, sharing the draft agenda before the call and moderating the call.
    - Contributions to the OSGi, Spring Dynamic Modules framework and state of the art of the DWHs, as well as the selection of Hibernate and BIM Server as data sources for dynamic and static information. Moreover, the technologies of External data services and GUI are described. Finally, the guides for the installation of the BIM Server and the adaptation of the database and the external sources are included.
- NEC
    - Work package leader: Review of the contributions and skeletons.
    - Task leader during August.
    - Participation to the weekly conference calls.

  - Contribution to the system architecture, as well as the selection of the Equinox Server for BaaS and the state of the art of the BMSs in the demo sites. Moreover, a detailed system requirements, the deployment of the components of the CLL and the adaptation of the BMS connector are included
- UCC
  - Participation to weekly conference calls.
  - Contribution in the installation of the Data Warehouse.

2 System architecture

The functional architecture (see Figure 1) of the BaaS system is defined in [1] based on a hierarchically layered approach. Of the three layers of the architecture are:

- APO Service Layer
- Communication Logic Layer
- Data Layer.

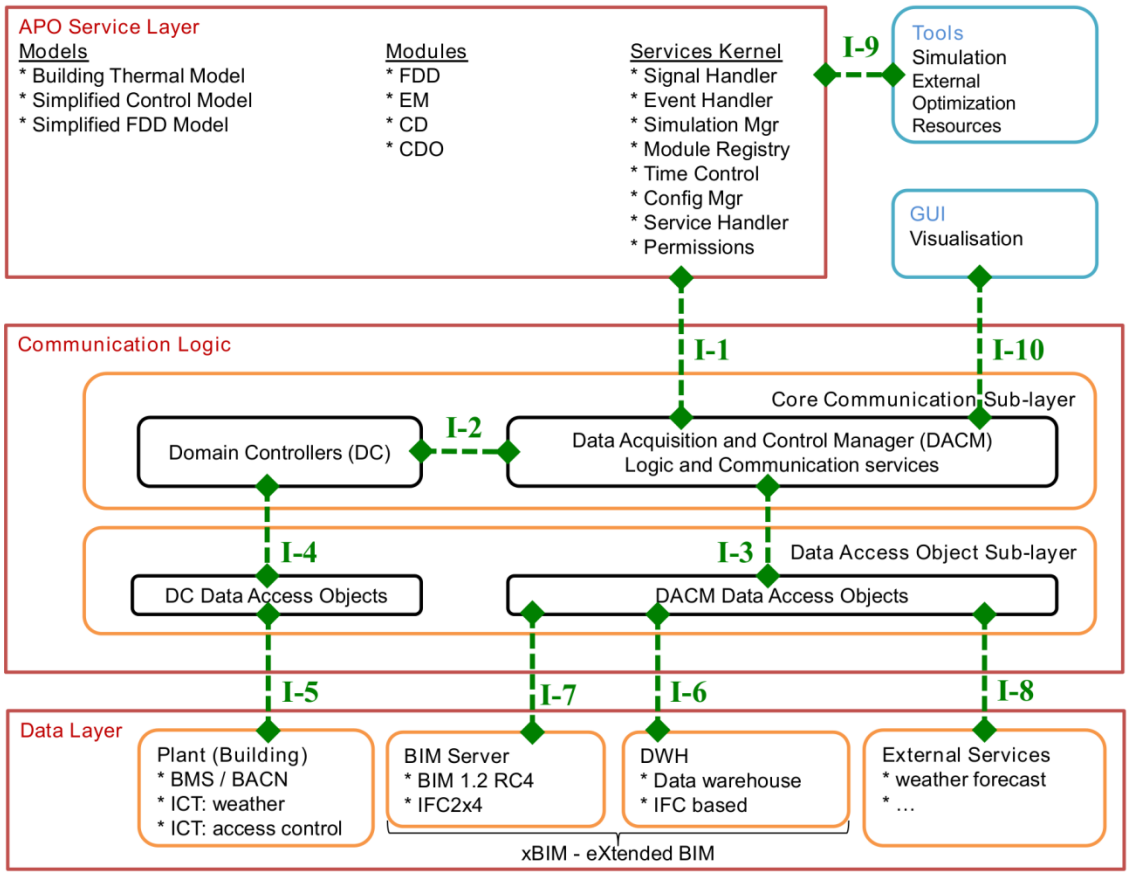

Figure 1: Middleware, Functional Architecture

The main interest of WP3 is aimed at designing and implementing the Communication Logic Layer (CLL) and the interfaces to the other layers. For this, WP3 chose a Service Oriented Architecture (SOA) approach, starting from high level use cases and resulting ultimately in the detailed system design [3].

The main components of the CLL are the functional components DACM and DC. The focus of the DC is to interface with buildings' and their systems. In contrast, the DACM aggregates multiple DCs, interfaces to other sources of information such as a weather forecast services, the BaaS Data Warehouse or the BIM server and provides data access services to the APO layer.

The detailed system design of the CLL is documented in D3.5 [3] and its interfaces as well as its internal communciation are defined in D3.3 [2].

In addition to the software _design_ documents, in this document the guidelines for its development, deployment, extension and adaptation are described. Moreover, the justification of the technologies selected for the prototype implementation are detailed in the following sections.

3 Technologies integrated

This section provides an overview of the technologies used in the development of the BaaS system. In order to help understand the technological basis of the system and reproduce a BaaS-like system.

3.1 OSGi framework

The OSGi Alliance is a worldwide consortium whose objective is the development of an open and mature specification and process in order to establish the guidelines for assembling and building modular Java applications [4]. This modular implementation reduces the software complexity of the development, being OSGi the reference model to modularize Java. OSGi Alliance is continuously creation a market for universal middleware through this technology.

OSGi is divided in several specifications so as to define the component system for Java. The set of specifications provided enables "a development model where applications are (dynamically) composed of many different (reusable) components" [4]. Making use of OSGi, the developers could hide the implementation from other components, allowing only the communication among them with the services declared in the framework. That sets up a simple model which "has far reaching effects for almost any aspect of the software development process" [4].

The architecture of OSGi framework is split into several layers such as Figure 2 [4] where the layer Bundles represents the software components developed or plugins. Services layer established the connectivity among bundles dynamically through the "publish-find-bind model" which, firstly, publishes the models, secondly, finds the references and, finally, binds dependencies. Also, there is a layer for the Life Cycle in order to control the installation, starting-up, updating and stopping of bundles. Modules layer defines how a bundle imports and exports code and the Execution Environment specifies methods and classes in the platform. Finally, there is a parallel layer called Security for the security aspects in the framework.

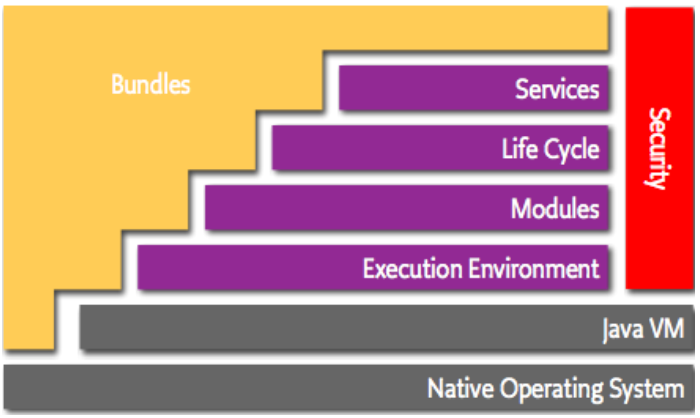

Figure 2: OSGi architecture [4]

The communication and interworking among components is done with the service references declared in every single module as dependency. In that way, OSGi offers factories for the loading of dynamic classes with the implementation of a service registry. Thus, a module can create an object and register it under one or more interfaces. Other modules can seek in this service registry the list of objects that have been registered. The Figure 3 [4] illustrates the process for registering, getting and listening to services. These services also include a set of properties so as to distinguish services under the same interface.

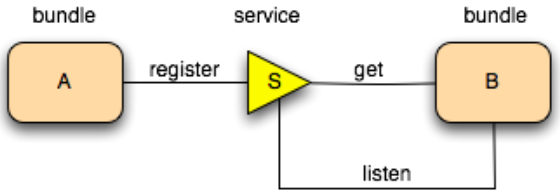

Figure 3: Service registry in the OSGi framework [4]

The BaaS Middleware design follows a modular approach that naturally fits OSGi philosophy. Therefore, **OSGi has been chosen as the framework for developing and deploying the software within the BaaS project**. In addition to the modularity, OSGi offers other advantages which the BaaS project benefits from:

- OSGi is very simple where the core API is only a package of less than 30 classes/interfaces. This core API is enough for managing the bundles. However, there are additional libraries providing optional functionality.
- Component-based platform reduces time-to-market and development costs because it enables integration of pre-built and pre-tested modules.
- Reduction of the complexity of the development by providing a modular architecture. The individual modules are Java bundles, communicating among them through well-defined services, hiding their internals which results in more flexibility of upgrading or changing module internals.
- Integration of multiple devices in a networked environment, tackling maintenance and remote service management. Also, the OSGi framework establishes how components are installed and managed.
- Mature component system running in a surprising number of environments, e.g. in many commercial Java Application Server products.
- The OSGi bundles are reusable, being easy the re-use of third-party components in the developments, such as commercial libraries. For BaaS system this property is interesting because it can be used the libraries for the BMS connection, Graphical User Interface and so on.
- OSGi is dynamic, allowing installing, starting, stopping, updating and uninstalling modules during runtime (on the fly) without bringing down the whole system.
- OSGi provides transparency because the management API gives access to the internal state of a bundle as well as how it is connected to other bundles, allowing the debugging in a live system via the OSGi console.
- The memory footprint is small with a minimum of Java Virtual Machine requirements.
- Lazy: OSGi technology has many mechanisms in place to load bundles only when they are really needed.
- OSGi loads the classes from bundles, whereas, in traditional Java, the JARs are visible and the classes are listed. OSGi pre-wires modules and knows exactly which bundle is implementing the class. This speeds up the start-up.
- OSGi has a fine-grained security model.

3.1.1 Spring Dynamic Modules

The Spring Dynamic Modules (SDM) is an Execution Environment for the OSGi framework which allows the use of dependency injection through the Inversion of Control feature (IoC) and service abstraction [8]. It is based on the Spring Framework and eases the use of Spring features that can be deployed in an OSGi framework, taking advantage of the features and services offered by OSGi. Thus, SDM allows instantiating, configuring, assembling, and decorating components within and across modules. Moreover, SDM controls the life cycle of the bundles and components deployed in the system instead of the developed or integrator. As well, it has excellent support for modularity and versioning [8]. However, in BaaS, the goal is not to develop Spring applications, but use the OSGi framework provided by SDM for the deployment

and the usage of IoC, above of all, for the Graphical User Interface. Furthermore SDM manages the life cycle of the bundle without the need of taking care of that during the development and deployment. Moreover, SDM instantiates the plugins when needed, decreasing the installation time.

Spring Dynamic Modules supports JDK level 1.4 and above and OSGi R4 and above. Bundles deployed for use with Spring Dynamic Modules should specify "`Bundle-ManifestVersion: 2`" in their manifest. It has been tested against popular OSGi servers such as Equinox, Felix, and Knopflerfish.

The IoC is one of the key features in Spring because it avoids the insertion of Java code in the modules for seeking the references in a service. Thus, the bundle code does not need to search for the interface in the service registry, but the Spring framework is in charge of the service registry access, controlled by XML files in the bundle JAR (Inversion of Control). For registering services, the bundle must declare the properties of the interface; meanwhile the listener has to specify the reference to the service. Then, the Spring framework inserts the service references automatically in the reference object in the Java code. An example of how to use services is below where the tag `osgi:service` declares the service with an "Id", the "reference" and the "interface". For consuming a service, the `bean` and `osgi:reference` are the useful tags. The former tag specifies the "class" making use of the service, the "property" in the class which is the instance of the service and the "reference" to the service. This "id" is also used as a reference by the latter tag for listening to the service. Thus, this second tag refers to the bean and the interface implementing the service.

```
<xml version="1.0" encoding="UTF-8"?>;
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:osgi="http://www.springframework.org/schema/osgi"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/osgi
       http://www.springframework.org/schema/osgi/spring-osgi.xsd">

   <osgi:service id="simpleService" ref="simpleService"
      interface="org.xyz.MyService" />

   <bean id="simpleConsumer" class="org.xyz.MyServiceImpl">
        <property name="attribute" ref="simpleService"></property>
   </bean>
   <osgi:reference id="simpleConsumer" interface="org.xyz.MyService"/>
</beans>
```

3.1.2 Communication based on OSGi events

One efficient pattern of communication among OSGi modules is based on the OSGi event mechanism. Events are broadcast in the OSGi framework via the EventAdmin service to all modules that registered themselves at the EventAdmin for receiving events. In other words, the EventAdmin service allows modules to publish and listen to events. Then, the combination of the events and the SDM helps the handle of the events because the bundle does not need to implement any code, but only making use of the IoC feature for managing the events.

As it has been specified in D3.3 [2], the essence of using this approach is the exchange of events as messages carrying properties with BaaS data defined inside the event.

In this context there are two ways of sending events using EventAdmin: asynchronous (postEvent(…)) or synchronous (sendEvent(…)) [4]. When a synchronous event is sent, the EventAdmin service finds all EventHandlers subscribed to the Event's topic, and notifies each one in turn; only continuing the Event sending module if all Handlers have executed.

Asynchronous event publishing is used when it is not important for the EventPublisher at which point in time the EventHandlers will receive and process a specific event.

To listen to events, a class must implement the interface EventHandler which registers the events. Moreover, it is necessary to state the topics the class is interested in, using the org.osgi.service.event.EventConstants.EVENT_TOPIC constant. The EventHandler object is registered with the Framework service registry and is notified with an Event object when an event is sent or posted. The class implementing this EventHandler has to define the appropriate actions that have to be executed when a particular kind of event (a particular topic) has occurred.

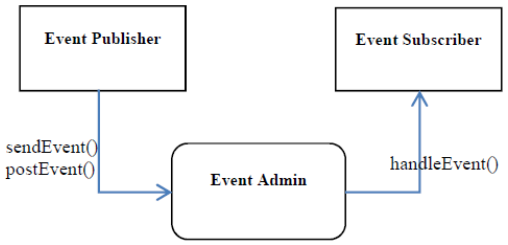

Figure 4: OSGi Event Mechanism

3.1.3 OSGi framework used in BaaS Middleware

The bundles have to be deployed on an OSGi framework, the bundle runtime environment. This is not a container like Java Application Servers, it is a collaborative environment. Bundles run in the same VM and can actually share code. The framework uses the explicit imports and exports to wire up the bundles so they do not have to concern themselves with class loading. Another contrast with the application servers is that the management of the framework is standardized. A simple API allows bundles to install, start, stop, and update other bundles, as well as enumerating the bundles and their service usage. This API has been used by many management agents to control OSGi frameworks [4].

Hence, for running DACM and DC components, a state of the art OSGi framework is needed to run the OSGi bundles. At the time of writing several commercial as well as open source implementations exist. In this deliverable we focus on the prominent Open Source Frameworks:

- **Apache Felix [9]**, implements OSGi Release 4 specifications under Apache License, driven by the Apache community. Last stable release from March 2013.
- **Concierge OSGi [10]**, implements OSGi Release 3 specifications intended for resource-constrained devices like mobile and embedded systems. Last stable release from April 2009.
- **Equinox [11]**, implements OSGi Release 4 specifications, driven by the Eclipse community. Latest stable release from September 2013.
- **Knopflerfish [12]**, implements OSGi Release 4 specifications available under BSD style license. The development is driven by Makewave. Latest stable release from July 2013.

As BaaS is interested in using state of the art OSGi technology, BaaS intends to select the most advanced but stable framework release. While OSGi specifications are at Release 5 since June 2012, framework implementations of Apache Felix, Equinox, Knopflerfish and Eclipse Gemini are at Release 4, Concierge OSGi at Release 3.

Therefore, and due to the fact that resource-constrained devices are not the prime target of deployment, Concierge OSGi is considered inappropriate for BaaS.

Of the remaining up-to-date frameworks, technically each is a viable option for the developing and deploying the BaaS system, with Equinox potentially being the quickest to be updated to new OSGi Releases – according [11] Equinox is even considered the reference implementation for the OSGi Release 4 framework specification. Additionally, Hibernate [5] (for DWH access, see 3.2.2) and Google Web Toolkit [13] (GWT, for GUI, see 3.6) natively support Equinox.

From an implementer's point of view, also the integration of Eclipse Equinox with the widely used Eclipse IDE poses an argument for using Equinox as it eases the testing process without the need of exporting the bundle into another OSGi framework. Furthermore, BaaS relies also on Equinox specifications for which Eclipse Gemini Blueprint (formerly Spring Dynamic Modules, see [14]) provides an OSGi based implementation that easily integrates with Eclipse. All these reasons motivate **our choice of using Eclipse Equinox as the base OSGi framework for the BaaS Middleware** DC and DACM components. In short, the main advantages are [11]:

- Integration with Eclipse, helping in the testing process.
- Easy integration of SDM, Hibernate and GWT.
- Easy configuration.
- Equinox first installs the bundles, looks for the dependencies and, then, starts the bundles in order to avoid dependency errors in the deployment.

Based on this choice of Equinox as OSGi framework container, the concrete deployment framework chosen in the BaaS project is the Equinox Server version Kepler (3.9.1) that makes use of the Spring Dynamic Modules version 1.2.1. The BaaS project considers a migration to Eclipse Gemini Blueprint Services as a possible evolution of the BaaS System. The libraries to be used in the deployment can be found in the distribution of the framework.

3.2 Data Warehouse

A data warehouse (DWH) is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources [15].

In addition to a relational database, a data warehouse environment includes an extraction, transportation, transformation, and loading (ETL) solution, an online analytical processing (OLAP) engine, client analysis tools, and other applications that manage the process of gathering data and delivering it to business users [15].

The key characteristics of a data warehouse are as follows [16]:

- Some data is de-normalized for simplification and to improve performance
- Large amounts of historical data are used
- Queries often retrieve large amounts of data
- Both planned and ad hoc queries are common
- The data load is controlled
- Fast query performance with high data throughput

In the BaaS project context the DWH is needed for the storage of historical data from the various data sources such as BMS, ICT systems and external sources. In this way, the DWH maintains the historical log of dynamic data from the different demo sites.

3.2.1 State of the art

In the market, several vendors are available for DataWarehousing such as Oracle, PostreSQL, SQL Server, MySQL and others. Oracle and SQL server are propiertary databases, whereas PostgreSQL and MySQL (supported by Oracle) are free-license ones.

Comparing PostgreSQL, MySQL and Oracle, the features are sumarissed in the Table 1 [16].

Feature	Oracle	MySQL	PostgreSQL
Data types compatibility	Subset of SQL'92 types plus specific types	Broad subset of SQL'92 types	Broad set of native data types
Constraints level	Very good (primary and foreign key and check constraint)	Average (No foreign key)	Very good (primary and foreign key and check constraint restricted)
Views	Yes	No	Yes, but using rules

Transactions	Very good (including rollback)	Not supported	Very good (without rollback)
Multi-users	Yes, no limit	Yes, limited	Yes, limited
Back-up	Yes	Yes, not online	Yes, not online
Scalability	Parallel multi-thread	Multi-thread	Not threaded
Analytical processing	Yes	No	No
Data size limit	Limited only by max. number of columns and max. size of columns of specific data types.	65534	16k maximum

Table 1: Comparison among Oracle, PostgreSQL and MySQL

Taking into consideration the features, the best suitable DWH is Oracle, in spite of the license, because it offers higher performance, more data size, check constraint in order to avoid wrong information and analytical processing for calculation of KPIs.

3.2.2 Hibernate

Hibernate [5] is a framework for the communication with databases in order to help the development of persistence. In this way, the goal is to store Java Objects beyond the scope of the JVM. However, the databases are relational which means that they do not work with objects, but relationships. That is the paradigm mismatch where an object cannot be directly mapped into a relational database. [5] presents the following five challenges addressed by Hibernate:

1. Granularity. The object model is more granular than the relational model because it has more classes than the number of corresponding tables in the database.
2. Subtypes (inheritance). Relational databases do not define any kind of inheritance which is common in object programming languages.
3. Identity. Relational databases define exactly one notion of 'sameness': the primary key, whereas objects define both object identity (a==b) and object equality (a.equals(b)).
4. Associations. Object relationships are represented in a unidirectional way, meanwhile the relationships in databases are ‘foreign keys’ which are bidirectional.
5. Data navigation. In objects the navigation is from one association to another walking through the object network which maximizes the SQL queries, being not desirable.

For these reasons, Hibernate is in BaaS used as database connector technology which bridges the gap between relational databases and objects. Thus, Hibernate allows the development of persistent classes, following object-oriented idioms including inheritance, polymorphism, association, composition, and the Java collections framework, which are mapped into tables in the database. Moreover, it does not require any interface or classes and offers high performance supporting lazy initialisation. Other important features are the stability, quality, reliability and scalability, being usable in any environment. Finally, Hibernate enables the usage of Hibernate Query Language (HQL) instead of native SQL, avoiding the knowledge of this language [5].

The current release, which is used in the BaaS project, is the version 4.4 that includes the following libraries.

- antlr-2.7.7.jar
- avro-1.7.5.jar
- commons-compress-1.5.jar
- dom4j-1.6.1.jar
- hibernate-core-4.2.6.Final.jar
- hibernate-jpa-2.0-api-1.0.1.Final.jar
- hibernate-search-engine-4.4.0.Final.jar
- hibernate-search-orm-4.4.0.Final.jar
- jackson-core-asl-1.9.2.jar
- jackson-mapper-asl-1.9.2.jar

- javassist-3.15.0-GA.jar
- jboss-logging-3.1.3.GA.jar
- jms-1.1.jar
- jsr250-api-1.0.jar
- jta-1.1.jar
- lucene-core-3.6.2.jar
- paranamer-2.3.jar
- slf4j-api-1.6.1.jar

3.3 BIM Server

The Building Information Model Server (BIM Server) is basically a data repository for information of a construction (or other building related) project. The core of the software is based on the open standard IFC and therefore knows how to handle IFC data [6]. The BIM Server uses the Model-driven architecture approach which means that IFC data are interpreted by a core-object and stored in an underlying database. The main advantage of this approach is the possibility to query, merge and filter the BIM-model and generate IFC files on the fly.

The BaaS project decided using the BIM Server in order to store the static data of the project: the demo sites' models are stored for providing information mainly to the simulation and APO Services, as well as the Graphical User Interface. Furthermore, the BaaS project is using IFC4 as data model, but the latest version of the BIM Server is compliant with IFC2x3. For bridging this gap, as outcome of the project, the BIM Server has been adapted to IFC4 [17].

In summary, the characteristics are as follows [6][17]:

- Check-in/check-out/download IFC: The BIM Server is able to version the projects based on IFC, allowing the maintenance of different versions of the models.
- Multi-user: Several users could interact with the BIM Server at the same time.
- Authentication: It applies with some of the security premises to be obeyed in the BaaS project.
- BimQL: An open library for querying the server so as to retrieve information of the buildings. Moreover, it brings the capability for filtering the queries for specific results of the requests.
- Merging: Ability to merge changes in a model using the object level calls which enables the possibility of checking the consistency between BIM and other data sources.
- Interfaces: It is allowed various open and standard interfaces for the communication such as SOAP, JSON and Protocol Buffers.

3.3.1 Communication with BIM Server

For the communication with the BIM Server, it is required to implement the BIM server client. Thus, three interfaces are defined: SOAP, JSON and Protocol Buffers. For performance reasons (see [17]), BaaS selected the JSON variant. There are two possibilities for getting the libraries to be used in the development phase.

1. TNO offers the download of the libraries for the client directly from the BIM Server Web Page [6].
2. At the first time deploying the BIM Server, a folder named "lib" is created containing the client libraries as well as the libraries of the server.

Based on the BIM Server used for development, an example of the list of client libraries is represented below. Note that depending on the release candidate and version the name could be slightly different.

- bimserver-1.2.RC5-2013-03-08-utils.jar
- bimserver-1.2.RC5-2013-03-08-shared.jar
- bimserver-1.2.RC5-2013-03-08-ifc.jar
- bimserver-1.2.RC5-2013-03-08-emf.jar

- bimserver-1.2.RC5-2013-03-08-client-lib.jar
- bimserver-1.2.RC5-2013-03-08-xsltserializer.jar

The libraries in the two cases before are the same and they enclose all the required functionalities of the BIM Server, such as the connection through any of the interfaces, check-in/out of IFC files, management of users and rendering queries.

A way of implementing the client functionality is to follow the guidelines offered in deliverable D2.3 [17]. Another alternative is to use the JavaDocs available at the BIM Server project webpage [6].

3.4 Building Management System

3.4.1 BMS available protocols

[7] lists the following technologies for building automation being in use in the demonstrator sites:

- In Demosite 1, Center for Sustainable Building. Kassel (Germany): a Saia Burgess system offers HTTP based access via CGI to the building systems
- Demosite 2, Technical University of Crete: provides various protocols such as OPC DA, BACnet/IP and Web based access.
- Demosite 3: Cartif offices building. Valladolid (Spain) is based on a LonWorks network and provides also a SOAP/XML based access.
- Demosite 4: Husa Chamartin Hotel. Madrid (Spain) and Demosite 5: Sierra Elvira School. Granada (Spain) both offer BACnet/IP.

For a description of state of the art building protocols we refer to [2].

3.4.2 Communication with the Building Management System

Depending on the different Demo sites, the DC-DAO sub layer needs to implement the appropriate protocol in the OSGi bundle responsible for BMS connectivity, see [2].

- Based on experiences of Campus 21 [18], we recommend relying on the bacnet4j library [19] for realizing BACnet/IP functionality for connecting to Demosites 4 and 5.
- For OPC foundation [20] members, a Java stack is available for download. Non-members will need to investigate the availability of open source alternatives. [21] provides an overview of Java libraries for OPC, more studies and experiments are needed at the current stage however.
- In case of Demosite 1, the open source Apache HTTPComponents Client library [22] is considered as adequate for the needs of BaaS to interact with the HTTP CGI script. [23] gives a brief overview of the URL structure. The individual product sheets need more investigation in order to implement the client interface towards this demosite.
- For the HTTP and SOAP based access at Demosite 3, [24] provides in chapter 22 an extensive description how access via the Java API for XML Web Services using JAX-WS 2.0 or 2.1 can be achieved. Notably, the manufacturer provides the WSDL file defining the messages and interactions for clients.

3.5 External Services

As it is explained in the D6.2 [7], in BaaS project there are several weather forecast services taken into account such as AEMET [25], Weather Underground [26] and The Weather Channel [27]. Within the BaaS project any other external service has not taken into consideration because the weather prediction is the only information (external to BaaS) needed for the APO layer.

Comparing among them, the first one is a Spanish service which provides European information but related to the main cities in Europe. Therefore, the useful information is regarding the forecast for Spanish cities, which is useless for the BaaS project. On the other hand, the other two are international services with weather forecast data for a large amount of regions in the world. However, Weather Underground offers an API for developers [28] that eases the management and access to the information of the forecast.

Weather Underground offers several plans for accessing the weather forecast data, where BaaS has chosen the Anvil plan because it is the more completed one giving hourly forecast for several days in advance. The characteristics of the service and the information included are summarised below.

- Anvil plan
  - Geolookup
  - Autocomplete
  - Current conditions
  - 3-day forecast summary
  - Astronomy
  - Almanac for today
  - 10-day forecast summary
  - Hourly 1-day forecast
  - Satellite thumbnail
  - Dynamic Radar image
  - Severe alerts
  - Tides and Currents
  - Tides and Currents Raw
  - Hourly 10-day forecast
  - Yesterday's weather summary
  - Travel Planner
  - Webcams thumbnails
  - Dynamic animated Radar image
  - Dynamic animated Satellite image
  - Current Tropical Storms

3.5.1 Communication between CLL and external services

For the communication point of view between the connector and the external service, Weather Underground is accessible through the IP addresses specified in D3.3 [2], as for example, http://38.102.136.138/api/05f74070aebc96d7/hourly/q/ES/Valladolid.xml in the Valladolid demo site. For that reason, the viability of using HTTP as communication protocol is taken as advantage because Java provides libraries for managing XML data (JAX libraries). Also, the information is available in JSON interface, but BaaS is going to re-use and improve the bundle developed in the sister project Campus21 [18].

The response is composed of several tags which the connector filters in order to store the data required in the BaaS project. Thus, in function of the building the modification is only established in the URL and not in the connector. Thus, for another city, the country code (ES, DE and GR in BaaS) and the name of the city are the changes needed.

3.6 Graphical user interface

Within the BaaS project has been defined a Graphical User Interface (GUI) in order to allow users to access the functionalities of the system. This requirement comes from the end-user point of view because of the access to the historical data, configuration parameters and optimization results. Thus, a GUI is intended for the visualization and interaction with the BaaS System.

3.6.1 Framework for the development

Several possibilities have been studied for the development of the GUI within the project such as AWT, Java Swing, Echo3 Web Framework and Google Web Toolkit (GWT). Finally, the last one is the selected technology due to the features offered. First of all, GWT [13] is a development toolkit for building and optimizing complex browser-based applications and is open source, completely free, and used by thousands of developers around the world, therefore,

the documentation is extensive. GWT enables productive and high-performance development of web-applications. It is based on JavaScript and Ajax languages, but avoiding the complexity of them. The developer implements the GUI in a common Java code through the Java APIs and the compiler is in charge of translating this code into Ajax and JavaScript. The great advantage is that it can be run across all the browsers, but also mobile phones.

GWT contains two powerful tools for creating optimized web applications. The GWT compiler performs comprehensive optimizations across your codebase - in-lining methods, removing dead code, optimizing strings, and more.

3.6.2 Behaviour of GWT

GWT projects are divided into two parts, the client and the server. Firstly, GWT compiles the Java source code into optimized, stand-alone JavaScript files that automatically run on all major browsers, as well as mobile browsers for Android and the iPhone. This is the client side where the code is downloaded at the first time to the browser in order to speed up the loading of the Web pages. On the other side, the server is running remotely in some machine. The server offers the main functionality such as the business logic of the GUI. Thus, the client contacts the server via Remote Procedure Calls (RPC) for requesting any operation. The server could itself perform the action or delegate the operations. In the case of the BaaS project, the server forwards the activities to the CLL making use of the event communication defined in [2]. In this way, GWT only allows the development of the screens in the client side without adding any business code; otherwise, an exception is launched in compilation time.

The version of GWT used is the latest one, 2.5.1, whose libraries are as follows.

- gwt-dev.jar
- gwt-servlet.jar
- gwt-user.jar
- validation-api-1.0.0.GA.jar

4 Deployment guideline

4.1 System requirements

- **Operating System**

Both DC and DACM Middleware components require the up to date OSGi Equinox framework. The framework is available for various standard Operating Systems and thus site owners and system administrators have big flexibility in choosing the Operating System. BaaS recommends using a recent release of Ubuntu Linux for both DACM and DC.

- **RAM**

The Middleware treats data transiently, i.e. it mainly forwards data between BMS, external ICT systems, internal ICT systems and the APO services using OSGi events. Therefore memory need is limited. Taking into account the fact that OSGi frameworks are run in a Java VM we base the recommendation for RAM size on the most recent change in default maximum heap size: Since Java 1.6.0_18 [29] the "default maximum heap size is half of the physical memory up to a physical memory size of 192 megabytes and otherwise one fourth of the physical memory up to a physical memory size of 1 gigabyte."

Hence, moderate amounts of RAM (1-2 GB) are more than sufficient for the DC and DACM components even for default maximum heap size settings.

- **CPU**

As the middleware does not run complex algorithms, using a modern low-end multi-core CPU is sufficient to run the DC and DACM components in small deployments. If many building DCs with many BMS devices each are to be aggregated and multiple APO services are to be served simultaneously, CPU demands will increase.

Thus, we recommend a modern Quad Core CPU for the DACM.

- **Hard Disk**

In productive use of the Middleware, extensive logging will not be needed as data is stored in the DWH. However, during development and an initial test phase, log files may be considerable in size – though not excessively big.

We recommend Hard disk capacity in the range of 100s of GB. It is rather important, that the configuration is resilient against failures of the hard disk, i.e. RAID configuration appears prudent.

- **Connectivity**

DC and DACM need internet connectivity, ideally multi-Mbps speed in up- and downlink. At the bare minimum a DSL link is needed, but this may result in delays if larger pieces of data are transferred between DC and DACM. Employing compression on the interfaces between DC and DACM can mitigate this effect somewhat.

For protecting the infrastructure and the communication, we propose to adopt the Campus21 deployment approach where the DCs establish a VPN connection with the DACM component. The Campus21 project made good practical experiences with the open source OpenVPN software both on Windows and Ubuntu Linux and thus OpenVPN is recommended for the BaaS project, too.

4.2 Installation of the data sources

The data sources in the BaaS system are one of the main parts because this layer provides the incoming information for rendering the APO services and optimization tasks. For that reason, several data sources such as DataWarehouse and BIM Server are designed in order to store the data from the BMS, APO calculation results, and static information from the buildings.

4.2.1 DataWarehouse

The first entity is the DWH which is developed focused on supporting objects concerning the building controls, HVAC, plumbing fire protection, and electrical domains utilizing IFC extensions. The entity is compliant with IFC4.

Currently, an Oracle database is implemented under UCC Zuse3 domain for managing all the data from demo sites. Oracle is a high performance relational database management system available for both Windows and Linux operating systems. Its primary query language is PL/SQL. To install an Oracle server, following steps are provided for future references.

1. Firstly, we install windows 2008.
2. Secondly, run all updates for operating system, reboot machine might be required for several times.
3. Get Oracle installation files from Oracle website or installation disc (win64_11gR2_database_1of2 and win64_11gR2_database_2of2, run setup.exe) and run it.
4. Set username and password for all database account. It is possible to set one password for all accounts.
5. Finally, register listeners and configure database server. Giving example of register listener as in Figure 5.

The existing database schema is implemented by exploiting the semantic richness of standard meta-model IFC, which will focus on developing a smart platform supporting advanced building performance analysis and control. Relational model in Figure 6 demonstrates the availability of BIM data for the categorization and classification (Dimensional Data) of monitoring data (Fact Data) within TUC demo site.

```
#PS C:\Windows\system32> sqlplus.exe sys AS sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Tue Mar 20 14:12:45 2012
Copyright (c) 1982, 2010, Oracle.  All rights reserved.
Enter password:
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
#SQL> alter system register;
System altered.
6b)
#PS C:\app\stephan\product\11.2.0\dbhome_1\BIN> ./emca -config dbcontrol db

STARTED EMCA at Mar 20, 2012 2:13:44 PM
EM Configuration Assistant, Version 11.2.0.0.2 Production
Copyright (c) 2003, 2005, Oracle.  All rights reserved.

Enter the following information:
#Database SID: orcl
#Listener port number: 1521
#Listener ORACLE_HOME [ C:\app\stephan\product\11.2.0\dbhome_1 ]:
#Password for SYS user:
#Password for DBSNMP user:
#Password for SYSMAN user:
#Password for SYSMAN user: Email address for notifications (optional):
#Outgoing Mail (SMTP) server for notifications (optional):
-----------------------------------------------------------------

You have specified the following settings

Database ORACLE_HOME ..... C:\app\stephan\product\11.2.0\dbhome_1

Local hostname ..... WIN-G4O6MKUJQSL.localdomain
Listener ORACLE_HOME ..... C:\app\stephan\product\11.2.0\dbhome_1
Listener port number ..... 1521
Database SID ..... orcl
Email address for notifications .....
Outgoing Mail (SMTP) server for notifications .....
```

Figure 5: Configuration of Registering Listener

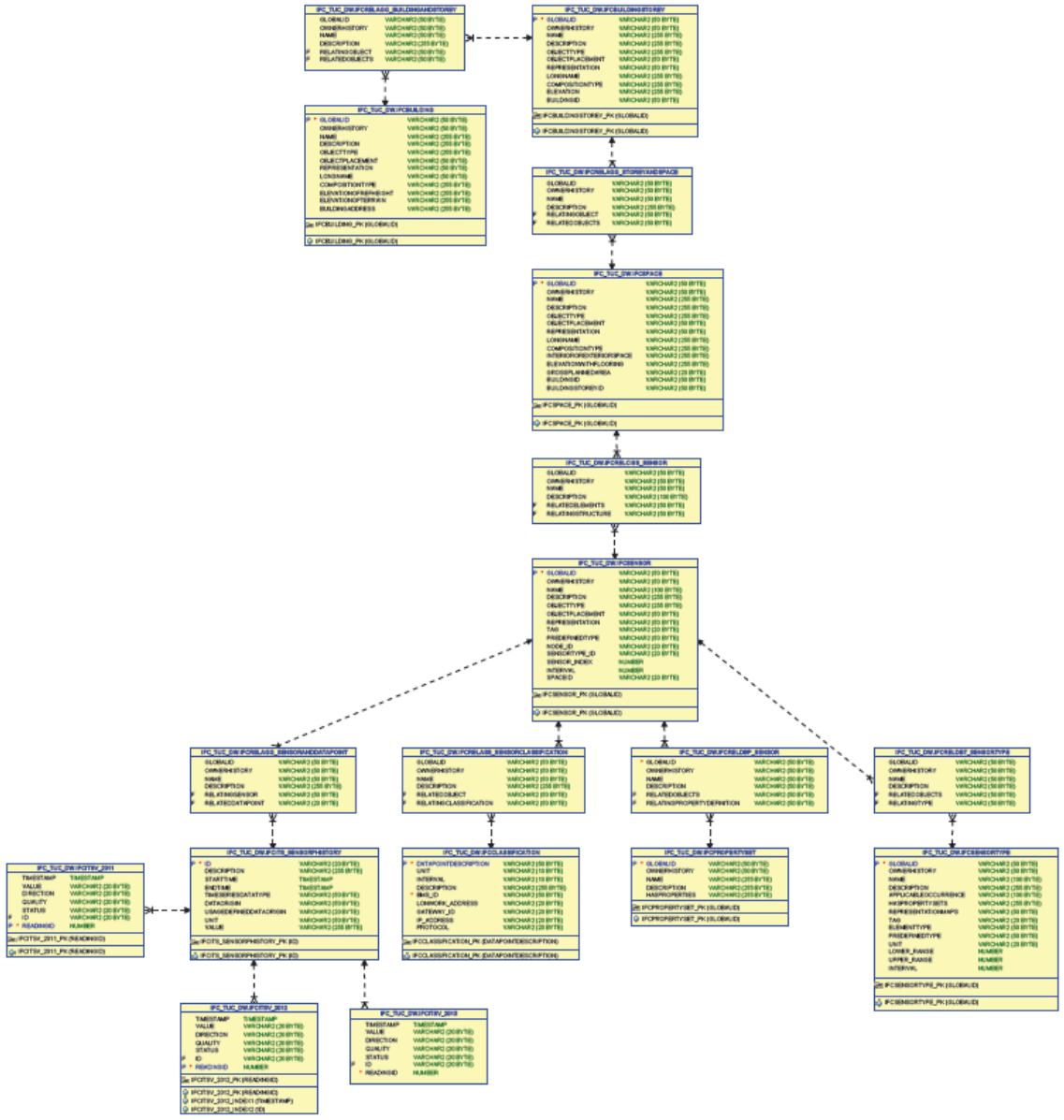

Figure 6: Relational Model of TUC DWH Schema

Major steps to re-engineering of an existing DW solution that compliant to IFC meta-model has been provided in BaaS deliverable WP2 Task 2. Giving example of SQL script to create IfcBuilding table in Figure 7.

```
CREATE TABLE ""IFCBUILDING"
 (  "GLOBALID" VARCHAR2(50 BYTE) NOT NULL ENABLE,
"OWNERHISTORY" VARCHAR2(50 BYTE),
"NAME" VARCHAR2(255 BYTE),
"DESCRIPTION" VARCHAR2(255 BYTE),
"OBJECTTYPE" VARCHAR2(255 BYTE),
"OBJECTPLACEMENT" VARCHAR2(50 BYTE),
"REPRESENTATION" VARCHAR2(50 BYTE),
"LONGNAME" VARCHAR2(50 BYTE),
"COMPOSITIONTYPE" VARCHAR2(255 BYTE),
"ELEVATIONOFREFHEIGHT" VARCHAR2(255 BYTE),
"ELEVATIONOFTERRAIN" VARCHAR2(255 BYTE),
"BUILDINGADDRESS" VARCHAR2(255 BYTE),
 CONSTRAINT "IFCBUILDING_PK" PRIMARY KEY ("GLOBALID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS" ;
```

Figure 7: SQL for Create IfcBuilding

The first population of DWH for the installation of BaaS system is executed manually. A specific Java program is created which first extracts data from BMS CSV file archive, eliminates inconsistencies such as duplicate rows, and then transforms the CSV file structure to IFC meta-model structure, finally loads the CSV file to the tables in the DWH schema. Compile BIM data to IFC DWH schema is simpler. Another Java program is used to extract relevant data from an IFC file exported from BIM model and loads into DWH tables. The ETL (Extraction, Transformation and Load) process is simulated in Figure 8.

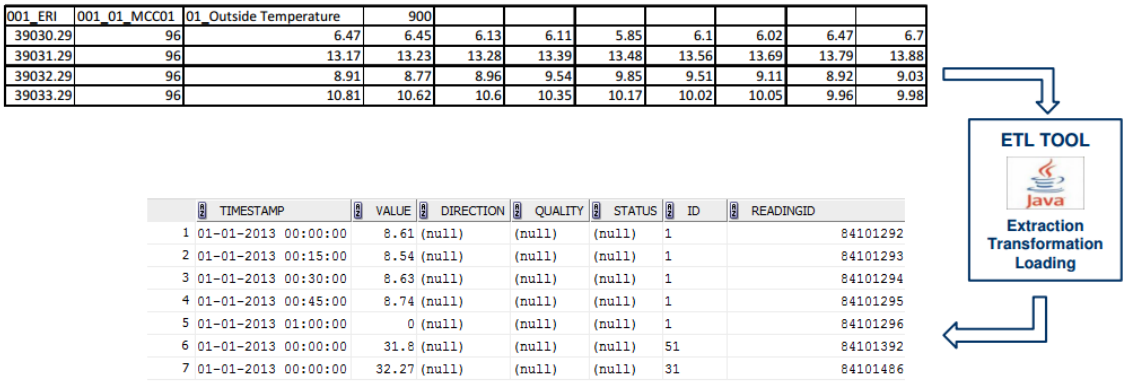

Figure 8: ETL Process

4.2.2 BIM Server

One of the entities in the data layer is the BIM Server which contains the static data of the building such as structure, materials, sensors installed and so on. This entity is based on the TNO BIM Server 1.2 RC4 which is compliant with IFC2x3. However, it has been adapted in the BaaS project to IFC4. Therefore, the need for uploading BIM information is a building model made up in IFC4.

The BIM Server is distributed in a JAR runnable file together with administration software for the maintenance of the server. First of all, the installation of the server requires a **Java Virtual Machine** in the system, as well as association in the Windows configuration of jar files with the virtual machine in order to run the Java application as stand-alone. Once all these parameters are ready, a single double-click in the file and the application is able to be launched, such as Figure 9. In the snapshot is shown the **properties to be set up** which are describe as following [6]:

- JVM: It is the Java Virtual Machine running environment in the system. Any user can change the *default* JVM for any other installed in the Operating System clicking in "Browse". Usually, it is left as default.
- Home directory: This property specifies the folder where the projects, files and configurations are saved in the system. For modifying the folder, any user has to click in the button "Browse".
- Address: That is the IP address where the BIM Server is accessed which is usually *localhost*. If the server where the service is accessible is different, here it must be written down the full IP address.
- Port: The communication port where the BIM Server is accessible. This port must be opened in the router, firewall or gateway for external purposes. If the opened port is modified, this property must be aligned.
- Max. Heap Size: This property configures the amount of heap memory assigned to the BIMserver in the JVM daemon; more heap means larger models can be stored/retrieved. Usually BIM Server needs at least 2GB of memory available in the machine hosting the instance.
- Max. Perm. Size: This parameter is dedicated to the memory of the plugins and 256MB should be enough.
- Stack Size: BIM Server runs in threads and this is the amount of memory dedicated to each thread. A common instance of the BIM Server makes use of 512KB but if more is required, a StackOverflowError exception is shown in the screen.

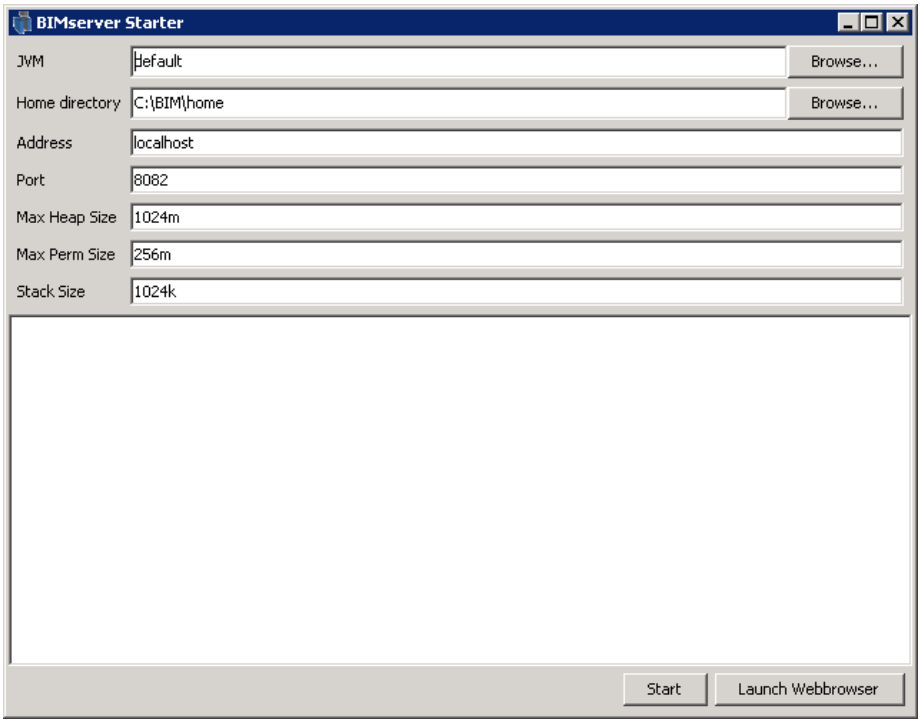

Figure 9: Configuration of the BIM Server

With all the properties correctly configured, the user must click on "Start" and the server begins the deployment. When it finishes a log message indicating "Server started" is displayed such as Figure 10. By clicking in "Launch Webbrowser" can be checked if the BIM Server is accessible. It is recommended to do it because the first time the configuration of the administrator user has to be completed (Figure 11). Firstly, the site address is shown and the mail properties below. Thus, the mail SMTP server and mail address are the properties associated. Furthermore, the

administrator account is described with the parameters name, username and password. The last two properties are the most important because they are useful at the beginning for the configuration of the BIM Server.

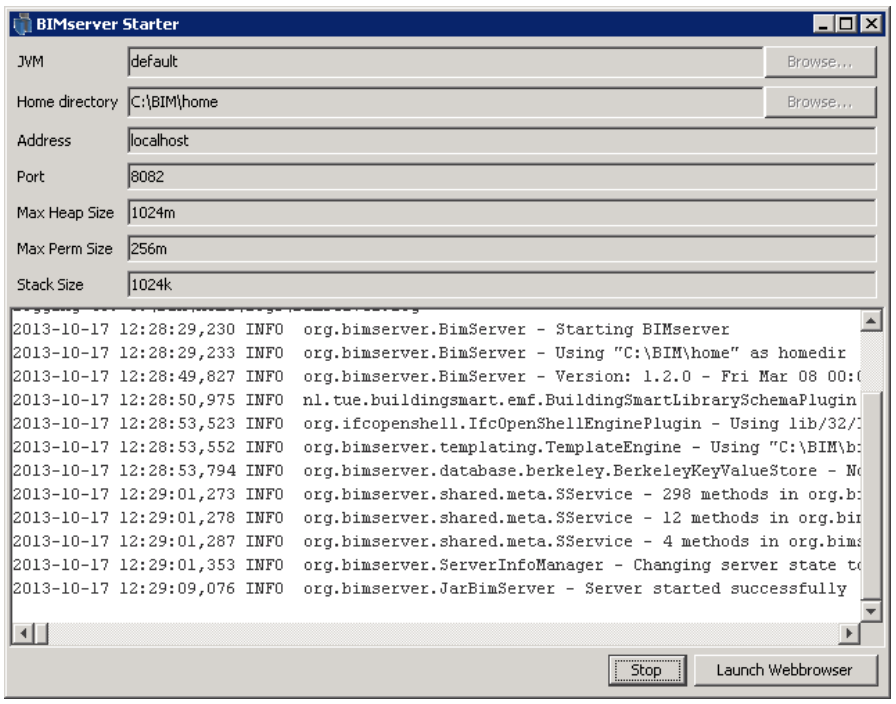

Figure 10: Deployment of the BIM Server

Figure 11: Setup BIM Server

On the other hand, no Graphical User Interface is distributed with the BIM Server, therefore is not accessible through a Web browser. For that reason, a message based user interface is developed and distributed as a jar running and executable file (Figure 12). That interface is

important from the administrator point of view because in the BaaS platform has been discarded the possibility of interacting through the middleware with the BIM Server apart from updated of malfunctioning sensors. Thus, the functionalities offered in the client side are:

1. Creating a new project: This function must be run at the beginning in order to create a new project with a specific name. The name of the project will be used afterwards for associating the project with users and IFC files, as well as running queries in the specific project/model.
2. Selection of a project: It selects a specific project into the possible in order to interact with the information related to it. This function has to be completed after the creation of the new project so that the selected one would be the usable.
3. Add a new user to the system: The goal is to include new users in the system in order to assign them to the projects. For that purpose, a form with some information such as name, username, mail and password has to be filled in. This functionality is run in the next step if there are no users in the system or it is required any new user for the project with the exception of the BIM administrator.
4. Add a user to a project: Within users in the system, it could be chosen one of them for the relationship with the project in order to allow him/her the exchange of information.
5. Upload IFC file: This functionality is the most important because the project created is empty and it needs the information of the model. For that reason, the IFC file is needed as well the absolute path where the file is (Figure 13).
6. Download project: It could also be run the download of the model in an IFC file, but it is an added value function.
7. Java Query Engine: In order to test the model is correctly uploaded, a query example could be performed.

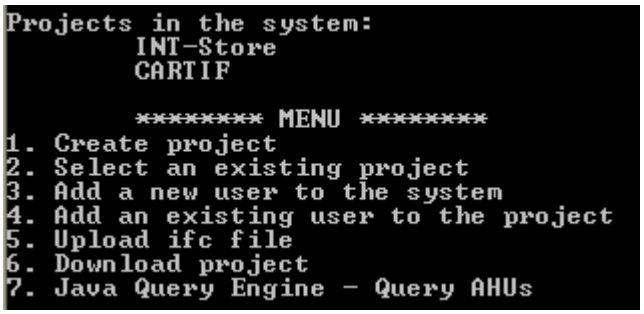

Figure 12: Functionalities in the BIM Server client

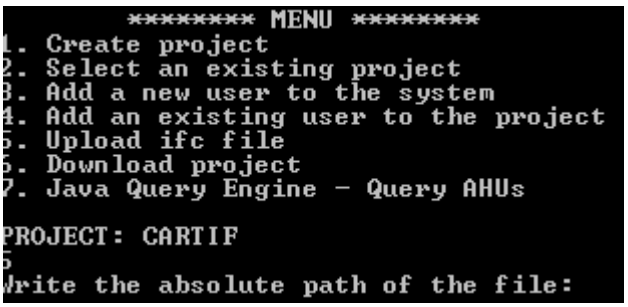

Figure 13: Uploading the ifc file to the BIM Server

4.3 Deployment of the OSGi server

4.3.1 Installation of the DACM

BaaS will rely on the Equinox Launcher Bundle to automatically install and start all DACM bundles. The corresponding Equinox jar bundles are OS dependent.

We denote the DACM installation folder as `[baas-dacm]`. The exact path format is OS dependent and can be chosen by the implementer freely, e.g. `/opt/baas-dacm` or `c:/baas-dacm`

- Download and extract the Equinox OSGi framework to a folder `[baas-dacm]`
- Download and place the following bundles (version numbers may change) in the `[baas-dacm]/plugins/` directory: `org.eclipse.equinox.launcher_X.X.X.jar`, `org.eclipse.osgi_X.X.X.jar`, `org.eclipse.update.configurator_X.X.X.jar` and `org.eclipse.equinox.common_X.X.X.jar`
- Download and place the bundles associated to the SDM framework into the plugins directory.
- Download and place the bundles associated to the Apache CXF framework into the plugins directory.
- Download and place the bundles associated to the Hibernate framework into the plugins directory.
- Place the `run.bat` and `run.sh` scripts for starting the DACM to the `[baas-dacm]` folder
- Copy DACM software bundles to `[baas-dacm]/plugins`
- Copy the DACM property file `dacm.properties` to `[baas-dacm]/` and edit this standard Java based properties file as needed.
    - At the time of writing, the exact properties contained in `dacm.properties` was not decided.
- Edit the `[baas-dacm]/configuration/config.ini` to read:

```
osgi.bundles=org.eclipse.equinox.common@2:start,\
      org.eclipse.update.configurator@3:start
eclipse.ignoreApp=true
osgi.noShutdown=true
org.osgi.framework.bootdelegation=javax.servlet, javax.naming,
      javax.naming.spi, javax.naming.event, javax.management,
      javax.management.loading, javax.management.modelmbean, javax.net,
      javax.net.ssl, javax.crypto, javax.crypto.interfaces,
      javax.crypto.spec, javax.security.auth, javax.security.auth.spi,
      javax.security.auth.callback, javax.security.auth.login,
      javax.security.cert, javax.xml.parsers, javax.xml.xpath,
      javax.xml.transform.sax, javax.xml.transform.dom,
      javax.xml.namespace, javax.xml.transform,
      javax.xml.transform.stream, javax.xml.validation, org.xml.sax,
      org.xml.sax.helpers, org.xml.sax.ext,
      com.sun.org.apache.xalan.internal,
      com.sun.org.apache.xalan.internal.res,
      com.sun.org.apache.xml.internal.utils,
      com.sun.org.apache.xpath.internal,
      com.sun.org.apache.xpath.internal.jaxp,
      com.sun.org.apache.xpath.internal.objects,
      com.sun.org.apache.xml.internal, org.w3c.dom,
      org.w3c.dom.traversal, org.w3c.dom.ls, javax.sql, sun.misc,
      javax.swing, javax.swing.event, javax.swing.border,
      javax.sql.rowset, javax.sql.rowset.spi, javax.imageio,
      javax.swing.text,
      javax.swing.tree,javax.xml.stream,com.ctc.wstx.stax,javax.smartca
      rdio,javax.sound.sampled, javax.persistence
```

At the time of writing, the list of DACM bundles was not finished and thus is not shown below. As an example, the following folder structure is created. The below example is based on a specific versioning and the assumption that Windows is used as OS.

```
[baas-dacm]/
  run.bat
  run.sh
  dacm.properties
[baas-dacm]/plugins/
  org.eclipse.equinox.common_3.3.0.jar
  […DACM bundles…]
  […SDM bundles…]
  […Apache CXF bundles…]
  […Hibernate bundles…]
  org.eclipse.equinox.launcher_1.0.0.jar
  org.eclipse.osgi_3.3.0.jar
  org.eclipse.update.configurator_3.2.100.jar
[baas-
dacm]/plugins/org.eclipse.equinox.launcher.win32.win32.x86_1.0.0/
   eclipse_1017a.dll
   […other files…]
[baas-dacm]/configuration/
    config.ini
```

4.3.2 *Installation of the DC*

BaaS will rely on the Equinox Launcher Bundle to automatically install and start all DC bundles. The corresponding Equinox jar bundles are OS dependent.

We denote the DC installation folder as `[baas-dc]`. The exact path format is OS dependent and can be chosen by the implementer freely, e.g. `/opt/baas-dc` or `c:/baas-dc`

- Download and extract the Equinox OSGi framework to a folder `[baas-dc]` Download and place the following bundles (version numbers may change) in the `[baas-dc]/plugins/` directory: `org.eclipse.equinox.launcher_X.X.X.jar`, `org.eclipse.osgi_X.X.X.jar`, `org.eclipse.update.configurator_X.X.X.jar` and `org.eclipse.equinox.common_X.X.X.jar`
- Download and place the bundles associated to the SDM framework into the plugins directory.
- Download and place the bundles associated to the Apache CXF framework into the plugins directory.
- Place the `run.bat` and `run.sh` scripts for starting the DC to the `[baas-dc]/` folder
- Copy DC software bundles to `[baas-dc]/plugins`
- Copy the DC property file `dc.properties` to `[baas-dc]/` and edit this standard Java based properties file as needed.
    - At the time of writing, the exact properties contained in `dc.properties` was not decided. For example, the internal identifier of the DC instance is configurable via the line: `baas.communicator.id = TUC-DC-0x01`
- Edit the `[baas-dc]/configuration/config.ini` to read:

```
osgi.bundles=org.eclipse.equinox.common@2:start,\
      org.eclipse.update.configurator@3:start
eclipse.ignoreApp=true
osgi.noShutdown=true
```

```
org.osgi.framework.bootdelegation=javax.servlet, javax.naming,
      javax.naming.spi, javax.naming.event, javax.management,
      javax.management.loading, javax.management.modelmbean, javax.net,
      javax.net.ssl, javax.crypto, javax.crypto.interfaces,
      javax.crypto.spec, javax.security.auth, javax.security.auth.spi,
      javax.security.auth.callback, javax.security.auth.login,
      javax.security.cert, javax.xml.parsers, javax.xml.xpath,
      javax.xml.transform.sax, javax.xml.transform.dom,
      javax.xml.namespace, javax.xml.transform,
      javax.xml.transform.stream, javax.xml.validation, org.xml.sax,
      org.xml.sax.helpers, org.xml.sax.ext,
      com.sun.org.apache.xalan.internal,
      com.sun.org.apache.xalan.internal.res,
      com.sun.org.apache.xml.internal.utils,
      com.sun.org.apache.xpath.internal,
      com.sun.org.apache.xpath.internal.jaxp,
      com.sun.org.apache.xpath.internal.objects,
      com.sun.org.apache.xml.internal, org.w3c.dom,
      org.w3c.dom.traversal, org.w3c.dom.ls, javax.sql, sun.misc,
      javax.swing, javax.swing.event, javax.swing.border,
      javax.sql.rowset, javax.sql.rowset.spi, javax.imageio,
      javax.swing.text,
      javax.swing.tree,javax.xml.stream,com.ctc.wstx.stax,javax.smartca
      rdio,javax.sound.sampled, javax.persistence
```

At the time of writing, the list of DC bundles was not finished and thus is not shown below. As an example, the following folder structure is created. The below example is based on a specific versioning and the assumption that Windows is used as OS.

```
[baas-dc]/
  run.bat
  run.sh
  dc.properties
[baas-dc]/plugins/
  org.eclipse.equinox.common_3.3.0.jar
  […DC bundles…]
  […SDM bundles…]
  […Apache CXF bundles…]
  org.eclipse.equinox.launcher_1.0.0.jar
  org.eclipse.osgi_3.3.0.jar
  org.eclipse.update.configurator_3.2.100.jar
[baas-dc]/plugins/org.eclipse.equinox.launcher.win32.win32.x86_1.0.0/
   eclipse_1017a.dll
   […other files…]
[baas-dc]/configuration/
    config.ini
```

4.4 Deployment of the APO services

The APO services can be deployed both stand-alone or on the server hosting the DACM, provided they follow the interface specification in [2]. For more information about how to deploy the APO Services, it is recommended to look up the documentation in the WP5.

4.5 Considerations regarding Cloud Deployment

As explained in [3], the BaaS System is compatible with current cloud computing approaches: it supports using private, public or hybrid cloud models as well as different business models.

Figure 14 indicates the interconnected BaaS system of which one of more entities might be hosted in a cloud setting – as also explained below. For the initial deployment, Cloud

technology is not needed, communication link security is established by securing interfaces I-1 to I-11 with dedicated VPN links.

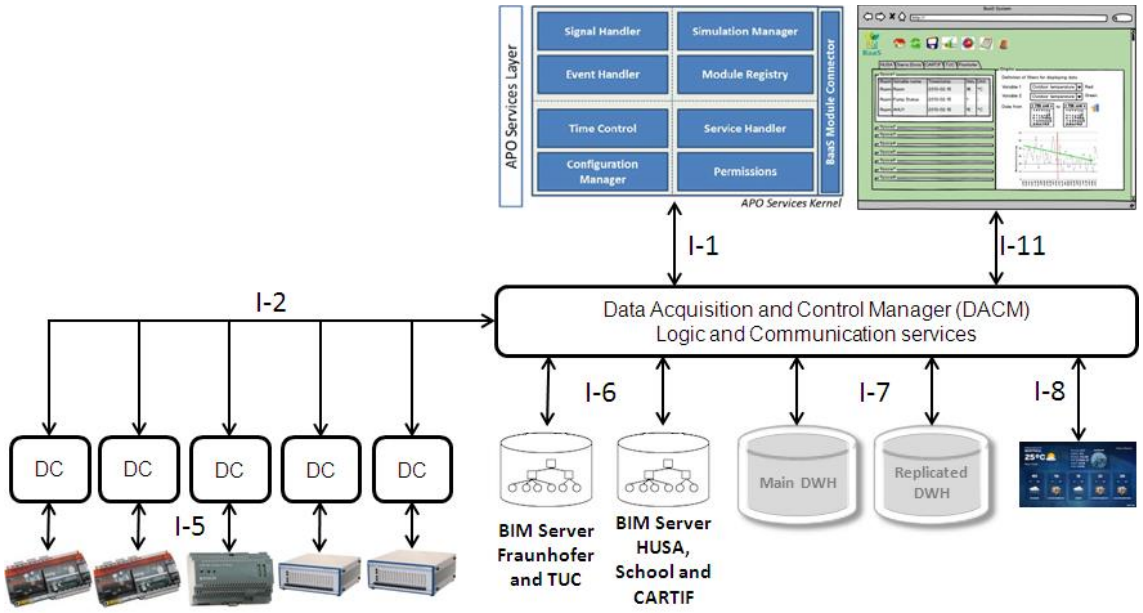

Figure 14: BaaS System deployment scheme

For a Cloud based system deployment however, BaaS could make use of:

- **IaaS** to host large scale versions of the system entities such as the DACM or the DWH – for example to serve and aggregate a large number of Buildings.
- **PaaS** to implement and deploy the different entities of the BaaS system, e.g. a DACM – assuming the PaaS offer provides the appropriate deployment environment as indicated above.
- **SaaS** could come into play to where BaaS makes use of external services and data sources.
- **NaaS** for interconnecting the different entities of the BaaS system in a secure manner with a high performance.

At the same time, BaaS could also offer APO services in a SaaS manner to users. Further, the BaaS system as a whole could be offered to third party developers as a PaaS offering to create advanced value added services, e.g. on top of the CLL. That means, BaaS could be itself a cloud for external users. For example, an ESCO, which looks for the replication of the BaaS system and it does not have the resources for installing a cluster of computers, could deploy the DC that is the only component associated to the building and make use of the "BaaS cloud" for running the algorithms or operations.

5 Development guideline

BaaS System is a software platform which is focused on the buildings used as demo site within the project. For that reason, some of the connectors are specific for the project such as the BMS connector adapted to the BMSs available, the DWH deployed in Oracle and Weather Underground as external services. Nevertheless, one the main goal in BaaS is the replicability of the results in other buildings, as for example, an ESCO with the idea of "selling" the service. Yet, this is difficult because not all the buildings present the same characteristics from the BMS protocol point of view. Also, the integrator could prefer a free database such as PostgreSQL or another external service for the convenience. In that way, this section treats to give a guideline for the integration of different data sources in the CLL.

5.1 How to adapt BaaS to different BMS protocol

We consider the technologies used in the demo sites as sufficiently diverse to address most buildings' needs. However, of course, additional protocol support may be necessary and potentially also variations in e.g. SOAP based access methods exist. In these cases we recommend implementing a new BMS connector component in the DC that translates the specific BMS protocol information into the OSGi event data structure defined for the BMS connector component. This new component then needs to be deployed in the DC component on the respective site.

In the vast majority of cases the aforementioned implementation should be sufficient to incorporate a new BMS protocol. However, in rare cases, this might require also updating the event definition, in case unforeseen data elements are used in this protocol. In this case, unfortunately, also changes to other components, e.g. the DWH may be necessary in order to store these new information elements.

5.2 How to adapt the database

As aforementioned, in BaaS project the DWH is Oracle which offers several great features. However, it is needed the payment of a license for the usage of the database that is sometimes not desirable. Thus, an adaptation process is required for different database vendors.

First of all, the creation of the tables and the initial insertion of data have to be adapted because the SQL queries are not completely standard in all the cases. However, in this way there is not a "universal" guideline for the adaptation because it depends on the database. It is highly recommended to look up the user manual in every single case in order to modify the queries accordingly.

On the other side, for the queries and the communication with the database it is used the Hibernate framework which eases this change. This framework maps the Java classes in relational tables through XML files and implements the queries using HQL language. The XML files have not to be changed because the Java objects are the same as well as the structure of the database. With regard to the HQL queries, these are the same owing to database independence. Furthermore, the only modification is the configuration file `hibernate.cfg.xml`.

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>

        <!-- Database connection settings -->
        <property
name="connection.driver_class">org.hsqldb.jdbcDriver</property>
        <property
name="connection.url">jdbc:hsqldb:hsql://localhost</property>
        <property name="connection.username">sa</property>
```

```
        <property name="connection.password">pass</property>

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">10</property>

        <!-- SQL dialect -->
        <property
name="dialect">org.hibernate.dialect.HSQLDialect</property>

      <!-- Enable Hibernate's automatic session context management -->
        <property
name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache  -->
        <property
name="cache.provider_class">org.hibernate.cache.NoCacheProvider</prope
rty>
        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>

        <mapping resource="org/hibernate/Event.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

In bold, it is highlighted the properties which have to be modified in the configuration file. First of all, the driver has to be referred to the database vendor (`oracle.jdbc.OracleDriver` for Oracle). Moreover, the URL of the database is different depending on the manufacturer, therefore, it has to be adapted. The user and password could have to be changed in case the database requires different credentials for the communication. Finally, the dialect specifies how Hibernate translates the HQL queries developed in Java into the specific SQL language related to the database.

5.3 External services

BaaS project considers Weather Underground as the only external service in the CLL. Therefore, this connector renders the communication with this service without caring of any other external service. The communication from the CLL to the Weather Underground service is performed via HTTP and in XML format. However, the services could be diverse and so much complicated to be adapted. Taking into consideration two possible adaptations:

1. Modification of the URL. A properties file specifies the URLs used for the communication with Weather Underground. These parameters map the building code [2][17] with the URL of the forecast associated to the city. Thus, the adaptation to another building is realised adding a new property mapping the building code and the URL for the forecast request.
2. If additional services has to be added is highly recommended the development of new class(es). In the BaaS connector, the class `ExtCommunicator` runs the connectivity with the service. This class has to be replicated or modified to read the data from the specific service. Furthermore, the events [2] could be different and the internal communication in the connector could require some adaptation.

6 Conclusions

During the writing of the present deliverable, three points were taken into account: technologies, deployment and extension guide. Bearing in mind the audience of the document which is a public one, not only for the BaaS partners, but also for any external user that wants to replicate the BaaS System in another building. Thus, the first section established the technologies selected in the implementation phase. At the beginning of the developments, the technologies and frameworks should be chosen so as to give the basis of the platform. In this way, BaaS decided the use of OSGi jointly Spring Dynamic Modules so that the features could enhance the behaviour of the CLL. Also, the integration of other technologies such as Hibernate or BIM Server help the adaptation of the system according to the interoperability and scalability requirements.

Secondly, the deployment guide helped any integrator who looks for the installation of the system. The steps for accomplishing the proper deployment were given so that the installation of the component could be as easier as possible. It is important the inclusion of this point owing to maintenance staff.

Finally, BaaS system could not cover all the possible solutions. Therefore, a development guide is included, so that any developer could adapt the system to a different building requirements or extend the functionalities for achieving other needs. Thus, the main components which are involved into the probably adaptations were described.

With all this content collected in the document, as well as the detailed design of the system, the next step is the development of the whole implementation. Within the following task, T3.4, the development is carrying out, which should follow the guidelines established both in this document and the other deliverables associated to the T3.3.

The critical points remarked in the life of the task and deliverable is the selection of the technologies. Owing to the expertise of different partners, it is sometimes difficult to reach an agreement of the development. The final decision must be made in function of the ability and capacity of the partners involved because, in another case, the implementation could lack of quality.

As future lines, the migration from Spring Dynamic Modules to Gemini Blueprint project is an “open door” in order to aling the BaaS system with the most up-to-date technology. Moreover, the adaptation and extension of functionalities section is centred in some of the data sources used in the BaaS project. However, more adatable systems could be discussed which is set up as an open point for future developments or integrations. The strongness, weakness, opportunities and threatness detected in the implementation will be described in future deliverables because in this document is difficult to move ahead those extensions which could be realised.

References

Authors are written using initial and surname. Books (including proceedings) are written in italics using the original title case. Titles of papers, chapters, etc are written using normal text with only the first word capitalized (with the exception of proper names, which may be capitalized). Each block of bibliographic information (i.e. authors, paper title, book title, date, location) is terminated with a period.

[1] BaaS Project Team, "D3.2: Functinal Architecture Specification", BaaS Project, 30th of April 2013.

[2] BaaS Project Team, "D3.3: Interfaces definition", BaaS Project, 31st of October 2013.

[3] BaaS Project Team, "D3.5: Detailed system design", BaaS Project, 31st of October 2013.

[4] OSGi Alliance, OSGi, http://www.osgi.org/Main/HomePage, OSGi Web Page, last visited on 17th of October 2013.

[5] JBoss Community, http://www.hibernate.org/, Hibernate Framework Web site, accessed 18th of October 2013.

[6] Open Source BIM Server, http://bimserver.org/, BIM Server web page, accessed 18th of October 2013.

[7] BaaS Project Team, "D6.2: Operative pilots after adapting", BaaS Project, August 2013.

[8] Spring Alliance, http://docs.spring.io/osgi/docs/1.0/reference/html/, Spring Dynamic Modules Reference Guide, Spring Framework, Spring Source web page, last visited on 18th of October 2013.

[9] Apache Felix. http://felix.apache.org/, accessed October 2013.

[10] Maven Concierge OSGi, http://concierge.sourceforge.net/, accessed October 2013.

[11] Equinox (OSGi), http://www.eclipse.org/equinox/, Equinox Eclipse Project, accessed October 2013.

[12] Knopflerfish OSGi, http://www.knopflerfish.org/, accessed October 2013.

[13] GWT Project, http://www.gwtproject.org/, GWT framework, GWT Web page, accessed 21st of October 2013.

[14] Spring Dynamic Modules becomes Eclipse Gemini Blueprint, http://www.eclipse.org/gemini/blueprint/documentation/reference/1.0.2.RELEASE/html/eclipse-migration.html, accessed October 2013.

[15] Oracle Docs, http://docs.oracle.com/cd/B10501_01/server.920/a96520/concept.htm, Oracle Data Warehousing concepts, accessed October 2013.

[16] BaaS Project Team, "D2.2: DataWarehouse", BaaS Project, 31st of October 2013.

[17] BaaS Project Team, "D2.3: BIM Repository", BaaS Project, 31st of October 2013.

[18] Campus 21, Deliverable 4.1: Specification of Integrated BMS Data Models and Protocols selection, December 2011.

[19] BACnet I/P for Java project ("bacnet4j"), http://bacnet4j.sourceforge.net/, accessed October 2013.

[20] OPC Foundation, http://www.opcfoundation.org, accessed October 2013.

[21] OPC, http://www.opcconnect.com/java.php, accessed October 2013.

[22] Apache HttpComponents Client, https://hc.apache.org/, accessed October 2013.

[23] SAIA, http://www.saia-pcd.com/en/technology/it-with-saia/cgi-interface/, accessed October 2013.

[24] i.LON® SmartServer 2.0, Programmer's Reference, http://home.elka.pw.edu.pl/~mkolasin/078-0347-01D_i.LON_SmartServer_2_Programmers_Reference.pdf, accessed October 2013.

[25] AEMET, "Agencia Estatal de Meteorología", Ministerio de Agricultura, Alimentación y medio ambiente, Gobierno de España, http://www.aemet.es/en/portada, accessed October 2013.

[26] Weather Underground, "Weather Forecast & Reports",Weahter Undergroud Service, http://www.wunderground.com/, accessed October 2013.

[27] The Weather Channel, http://www.weather.com/, National and Local Weather Forecast, accessed October 2013.

[28] Weather Underground, "Weather Underground API", Weather Underground Service, http://www.wunderground.com/weather/api/, accessed October 2013.

[29] Oracle Update Release Notes, Changes in 1.6.0_18 (6u18), http://www.oracle.com/technetwork/java/javase/6u18-142093.html, accessed October 2013.