Grant Agreement No.: 318600

SODALES

SOftware-Defined Access using Low-Energy Subsystems

Funding Scheme**: *Small or medium-scale focused research project STREP - CP-FP-INFSO***
Activity: **ICT-8-1.1 - Future Networks**

D3.4 Control and Management Plane Software

Due date of the Deliverable: Month 24
Actual submission date: 21/11/2014
Start date of project: November 1st 2012 Duration: 36 months
Project Manager: Carlos Bock | i2CAT
Version: 1.0

Author List: Jordi Ferrer Riera (i2CAT), Carlos Bock (i2CAT), David Levi (ETH), Victor Marques (PTIN), Gil Mahlev (ETH), Alex Weis (ETH), Pedro Mendes (PTIN)

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract

This deliverable contains the description of the prototype developed for the SODALES hardware devices. The management plane consists of a set of software components capable of providing an unified configuration platform towards the heterogeneous devices considered (i.e. the Active Remote Node and the Customer Premises Equipment), which development is being lead by different partners within the consortium. The prototype becomes the centralised management point for the SODALES convergent access network solution.

The prototype is built on the basis of two major cornerstones:

- Providing an open access network model over the physical infrastructure considered
- Providing an unified platform for management and operation of the devices

The deliverable is composed of both the report and the developed prototype.

Executive Summary

This deliverable contains the report associated to the prototype released for the SODALES software management component, which is built in order to cover two major requirements on the developed SODALES architecture and hardware, i.e. the open access features to provide several service providers with the capability of providing services without owning the physical infrastructure itself, and the objective of building a common software platform for the different hardware components developed and/or considered in the project, mainly being the ARN and the CPE.

Section one introduces the report, and the prototype, linking them to the previous deliverables including the requirements and design utilized for the development.

Section two contains a complete description of the environment used in order to develop the software tool. It focuses on the structure of the development team, how different tools have been used during the process and what functionality or feature each tool provided.

Section three contains the technical description of the prototype. It focuses on the different components developed, what have been the major modules implemented for each device, and how open access has been implemented through the Open Network-as-a-Service framework.

Section four contains the validation of the prototype performed. The validation performed are described in terms of requirements covered

Section five comprises the different steps required to download and install the developed software. In order to make it work it is required to download and install a couple of additional libraries. All the instructions are provided in the section.

Finally, section six contains concludes the document, with the future work to be done in the software component during the field trial and the pilot deployment, and the lessons learnt during the development stage.

Table of Contents

List of Figures

List of Tables

1 Introduction

Deliverable D3.3 [SODALES-D3.3] contained the initial software design for all the modules required to implement the different functionalities of the SODALES management plane. The different functional blocks populating both components of the system (i.e. the integrated element for the infrastructure provider and the virtualized element for the service provider) were presented in the report, including the different requirements covered by each functional block.

In order to cope with the functional design covered during the first months of execution of the work package, the development activities were started within the second year. The development has been focused on covering the required aspects to provide the two major objectives of the SODALES software layer:

- To provide an open access network model over the physical infrastructure considered, enabling the infrastructure provider to create independent slices of the infrastructure.
- To provide an unified platform for management and operation of the different devices, both virtual and physical, for the infrastructure providers and/or the service providers, depending on the type of resource addressed.

The functional design was performed over the Open Network-as-a-service [OpenNaaS] framework. OpenNaaS is an open-source framework, which provides tools for managing the different resources present in any network infrastructure. The software platform was created in order to offer a neutral tool to the different stakeholders comprising an open access network. It allows them to contribute and benefit from a common NaaS software-oriented stack for both applications and services. It is based on a lightweight, abstracted, operational model, which is decoupled from actual vendors' specific details, and is flexible enough to accommodate different designs and orientations. In fact, the OpenNaaS framework provides tools to implement the logic of an SDN-like control and management plane on top of the lightweight abstracted model. The elements loaded in OpenNaaS contain a model, which stores the information about the resource, and a set of capabilities that allows the operation and manipulation of the software-represented resources. We have followed this approach in order to implement the SODALES software layer.

The figure below depicts the general OpenNaaS architecture and the resource abstraction concept inherent in the architecture and extended by the SODALES activities. The architecture image was already included in Deliverable D3.3 [SODALES-D3.3].

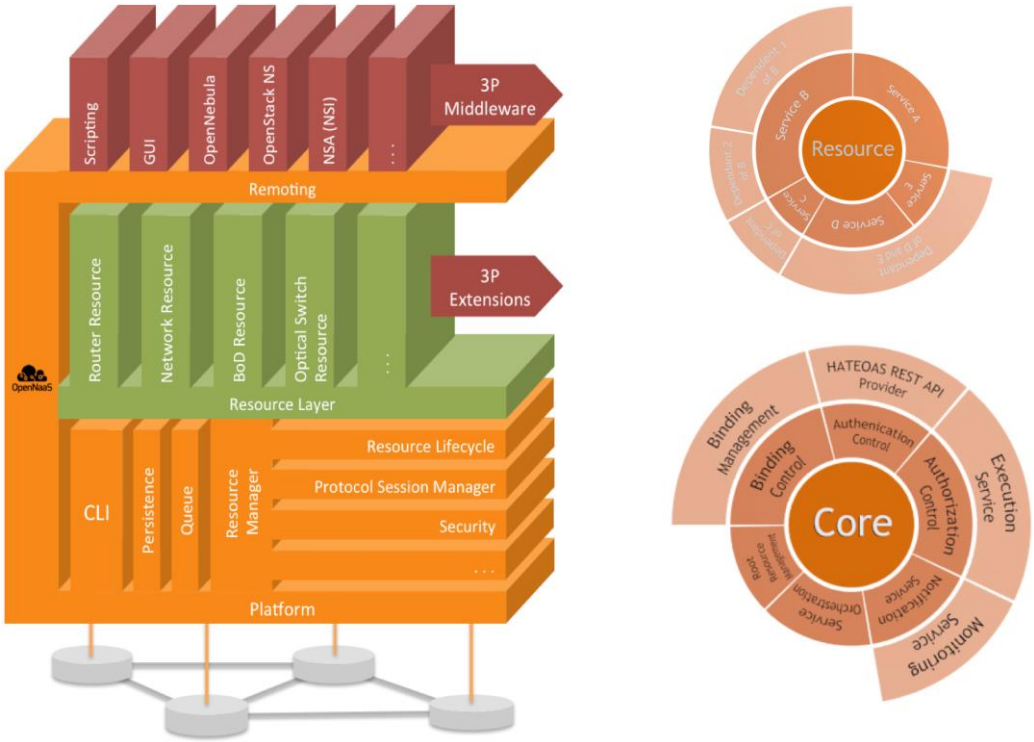

Figure 1: OpenNaaS architecture, and resource abstraction concept

The deliverable contains the description of the Active Remote Node and the Customer Premises Equipment clients and data models implemented, associated to the abstracted model and the set of capabilities attached to these devices. Furthermore, the deliverable contains the different capabilities and features implemented in order to provide the open access network model.

The rest of the document is structured as follows. Section two contains a global and generic description of the development environment utilized for the SODALES project, including the software management tools used. Section three contains a description of the most important and representative parts of the SODALES prototype. For the sake of readability, we have included only separated parts of the interfaces and source code implemented in the prototype. There are then other attachments in the different Appendixes. Section four contains the validation and tests performed to ensure the developments work as expected. Section five describes how the prototype is released and how it can be deployed, and then section six concludes the document.

2 Software Development Environment

Task 3.4 of the SODALES Description of Work states that SCRUM [Scrum] development methodologies will be applied in order to optimize development costs and simplify the project management. This section contains the description of the software development environment used in order to enable SCRUM development methodologies.

The section contains information about the software repository used, and the different development management tools in place. It must be remarked that the development of the SODALES project has been integrated into the development planning and periods of the OpenNaaS framework, in order to ensure that a correct integration and re-use of the available tools were maximised.

The following image depicts the development organisation we have followed in order to implement the management plane for the SODALES project.

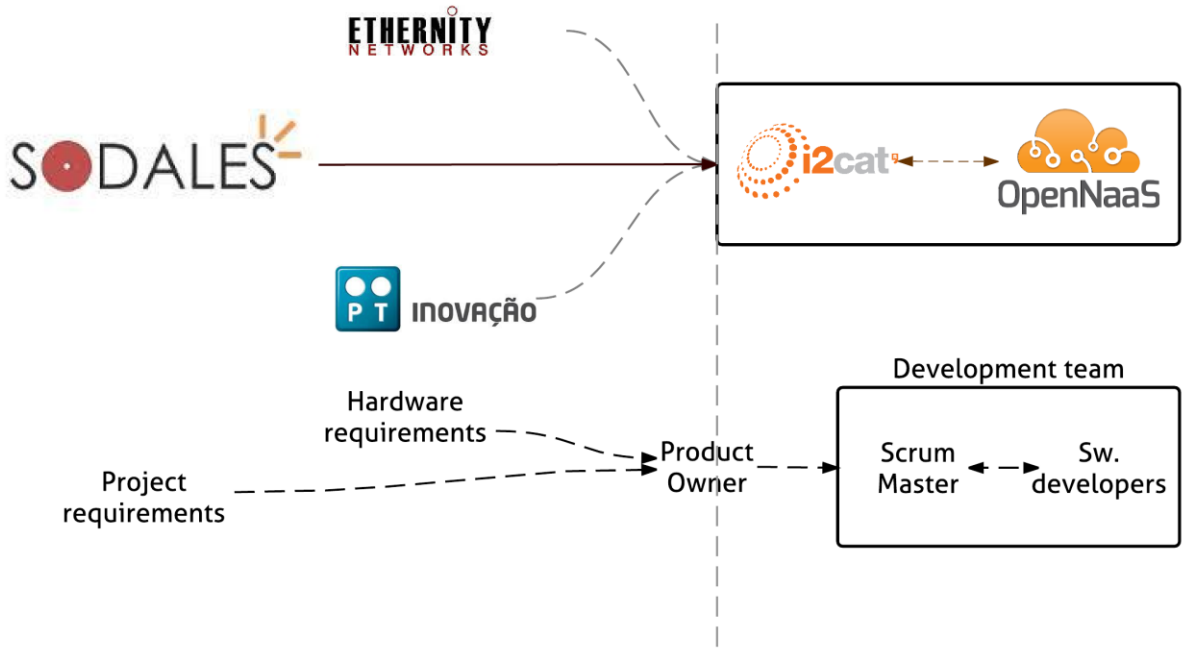

Figure 2: SODALES software development structure

First, there were two major types of requirements to be covered, the ones coming from the project, and the specific ones coming from the hardware devices. Those requirements comprehended the customer part of the development. The requirements were gathered by the product owner and then exchanged with the development team. We tried to fully duplicate a commercial oriented software development environment. However, there were some flaws that required to be adapted to the specific SODALES environment: (i) the developers teams from each institution were fully distributed, so any change in the hardware requirements due to new developments or updates in the equipment (which was being implemented) supposed time delays in the translation of this requirements to specific features in the management plane; and (ii) the product owner was within the development team, when it must be external to the development team; however, it was not possible to work with a team completely external to the SODALES project.

2.1 Ticketing and planning system: the JIRA platform

In order to manage all the development process we have used the JIRA [Jira] system, from Atlassian. JIRA is the tracker for teams planning and building software products. It provides bug tracking, issue tracking and project management functions. For the development phase, in SODALES we have mainly used project management and issue tracking functionalities; while it is planned that for Y3, for the deployment of the pilot, the bug reporting feature will be the one mostly used by the project.

As a curiosity, the product name JIRA is not an acronym, but the truncation of the word *Gojira*, the Japanese name for Godzilla [Godzilla]. According to Atlassian numbers, over 25.000 customers within more than one hundred countries use JIRA for issue tracking and project management worldwide.

The following image shows an example on how JIRA has been used for the SODALES project. It is deployed at i2CAT premises in the following address:

```
http://jira.i2cat.net
```

In order to log in it is necessary an user and password. In the corresponding section software release and deployment we provide a generic user account in order to access the different systems if required for downloading and deploying the OpenNaaS platform with the extensions.

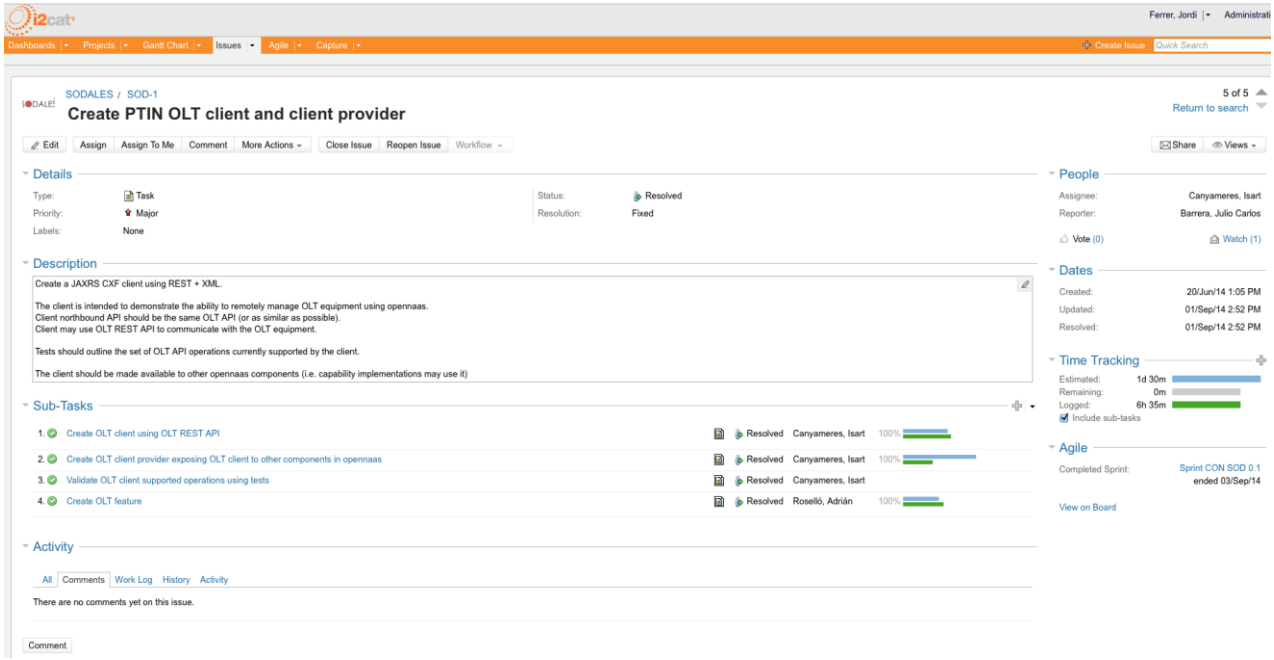

Figure 3: JIRA for the SODALES project

The image contains the description of an user story (Create PTIN OLT client and client provider), and how this story is divided into four different tasks. It can be seen also to which development sprint the story was assigned, and the members of the team responsible for the development. There were also time tracking features, which were only used as an internal component for the i2CAT team, not made extensible for the rest of the involved partners.

2.2 Software repository

We have used Git [Git] as the revision control system for the software repository. The repository used is installed at i2CAT facilities under a private deployment. It is located at the following address

```
http://stash.i2cat.net
```

Git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. In order to access the repository for the SODALES software developments access credentials are required. Details are provided on section five, software release.

2.3 The Bamboo Platform

The third tool used is associated to the Jira suite. It is the bamboo platform. It is an automated, and integrated testing platform, responsible for the continuous validation of the developments performed. The Open network-as-a-service development team has used it in order to validate SODALES developments.

```
http://bamboo.i2cat.net
```

Access credentials are also required to access the platform.

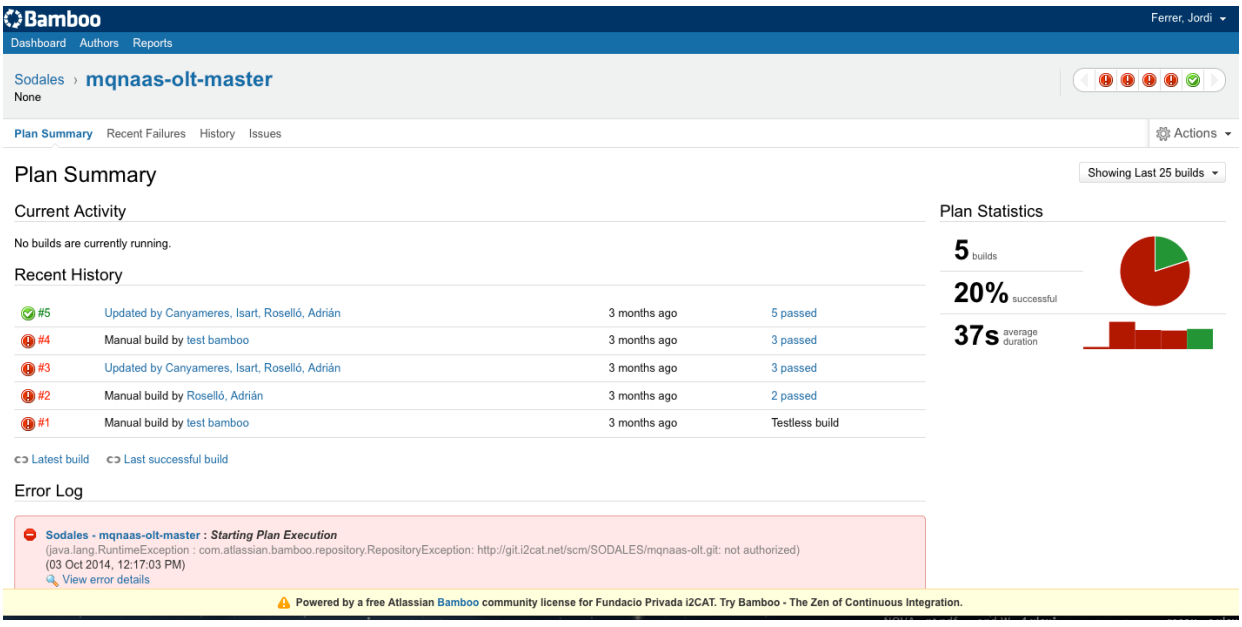

Figure 4: Bamboo for the SODALES project

3 Software Implementation

This section describes the software implementation for the management plane of the SODALES Project. The software development has been focused on two major cornerstones, as extracted from the initial requirements:

- Building a common software management platform for the hardware devices involved in SODALES (ARN, CPE). The software platform needs to be modular and extensible, since different CPEs can be then easily integrated with the minimum required effort
- Building an open access enabler for converged access networks, focusing on the SODALES ecosystem for demonstrations and validations

As stated in D3.3 [SODALES-D3.3], the SODALES software plane has been built over the Open network as a service framework, an open-source framework developed within the EU Mantychore project [MANTYCHORE-FP7]. Apart from this, the software has been built over the two proprietary APIs used for configuration of the physical devices, i.e. the ARN and the CPE.

The solution implemented has followed a centralized SDN-based approach. Thus, a central entity (i.e. OpenNaaS release) holds all the knowledge of the access network, physical connections, and physical devices. This is the classic SDN base, where by means of the separation of the control and data plane there is one centralized controller performing all the networking decisions. This was firstly achieved with the OpenFlow protocol.

However, for SODALES, we have followed a more generic approach. First, because OpenFlow is only for L2 switching equipment, and thus it is not supported in the multi-service, multi-tenant ARN. Secondly, because the modularity defined in our centralized management plane enables to include different devices, independent from the technology, by means of implementing specific web services clients

The following image depicts how the integration of our control plane with the hardware devices has been achieved. In the southbound interface of the OpenNaaS platform, simplified REST Web Services [REST] are being used in order to interact with the different devices. Thus, in order to integrate any other device from any other vendor, which is a very likely event in access environments with different vendors, it is only required to implement the corresponding web service client and integrate the data model of the resource with the corresponding generic data models in the platform.

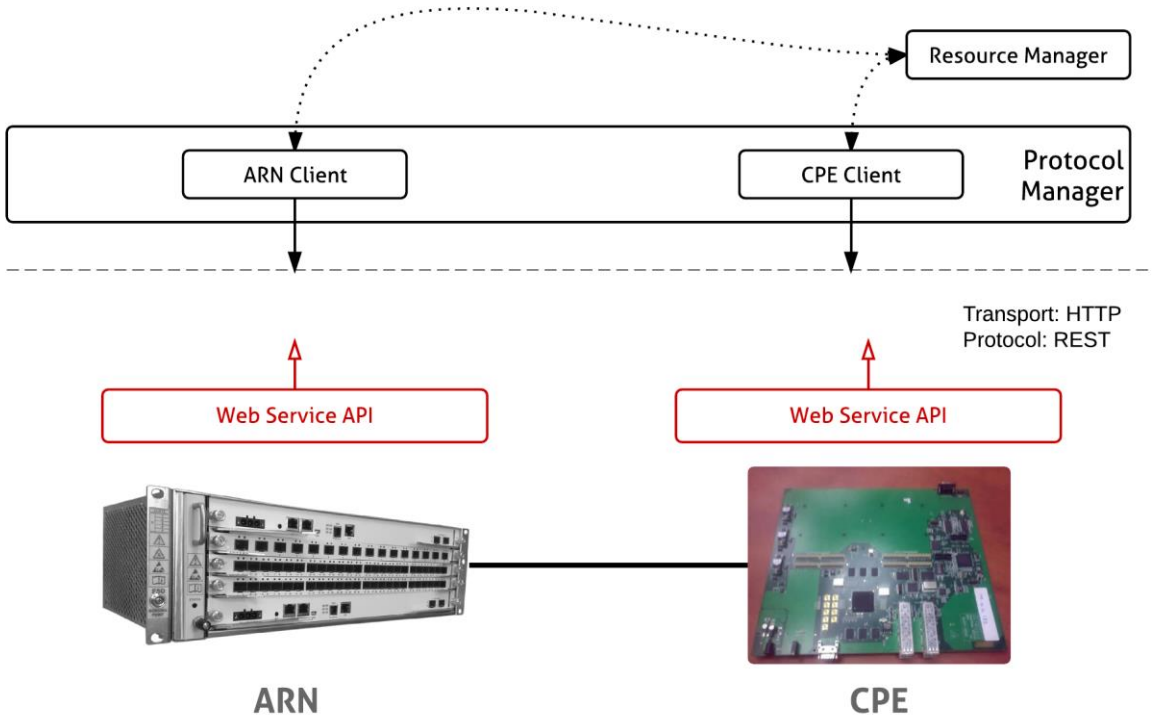

Figure 5: SODALES Southbound protocols

Furthermore, through this generic solution, the open access (or virtualization) model selected within the management plane is built over the generic data model created and it is not bound to any specifics of any network protocol. There is however one drawback of this solution, that is the network devices must support or provide web services APIs for management and configuration of the devices themselves, or either adapt the existing ones in order to provide a complete interface for configuration of the device.

The rest of the section is structured as follows. First, we provide the description of the client for the ARN and the data model associated used within the management plane. Secondly, we provide the description for the CPE, which has consisted of creating the API to be provided by the equipment and then the data model within the OpenNaaS system. Finally, the section contains the description on how the open access is enabled by means of OpenNaaS and the associated generic reservation capability, which enabled the service providers to reserve for a given slice for a determined period of time.

3.1 Active Remote Node

The ARN is the key device of the SODALES architecture. It has been developed as an active node in order to provide convergence in the access between wireless and wired access. The architecture, the associated characteristics, and the different features have been well described in different deliverables [SODALES-D2.1, D2.2, D2.4] Please refer to those deliverables for further details on the hardware specifications and low-level characteristics of the device.

In order to develop the components, which provides support for the ARN in the OpenNaaS framework we have used as a basis the API provided by the equipment vendor.

The data model created using as a basis the management interfaces available for the ARN chassis is based on four major components:

- System Discovery (e.g. equipment identification, retrieve information)
- Performance Monitoring (e.g. equipment interfaces, running services)
- Alarm and Event Monitoring (e.g. status information)
- Systems and Service Provisioning (e.g. system configuration, service provisioning)

The data model has been built using XML Schema Data (XSD) files, which can then directly be translated into Java files. The following figure contains the structure and the different components for the data model. The files are licensed under PTI copyright, since they represent the available management interface. Appendix D contains the JAXB bindings that enable the translation of the data model at the ARN client in the SODALES software component from XSD files to Java files, which are then ready to be manipulated.

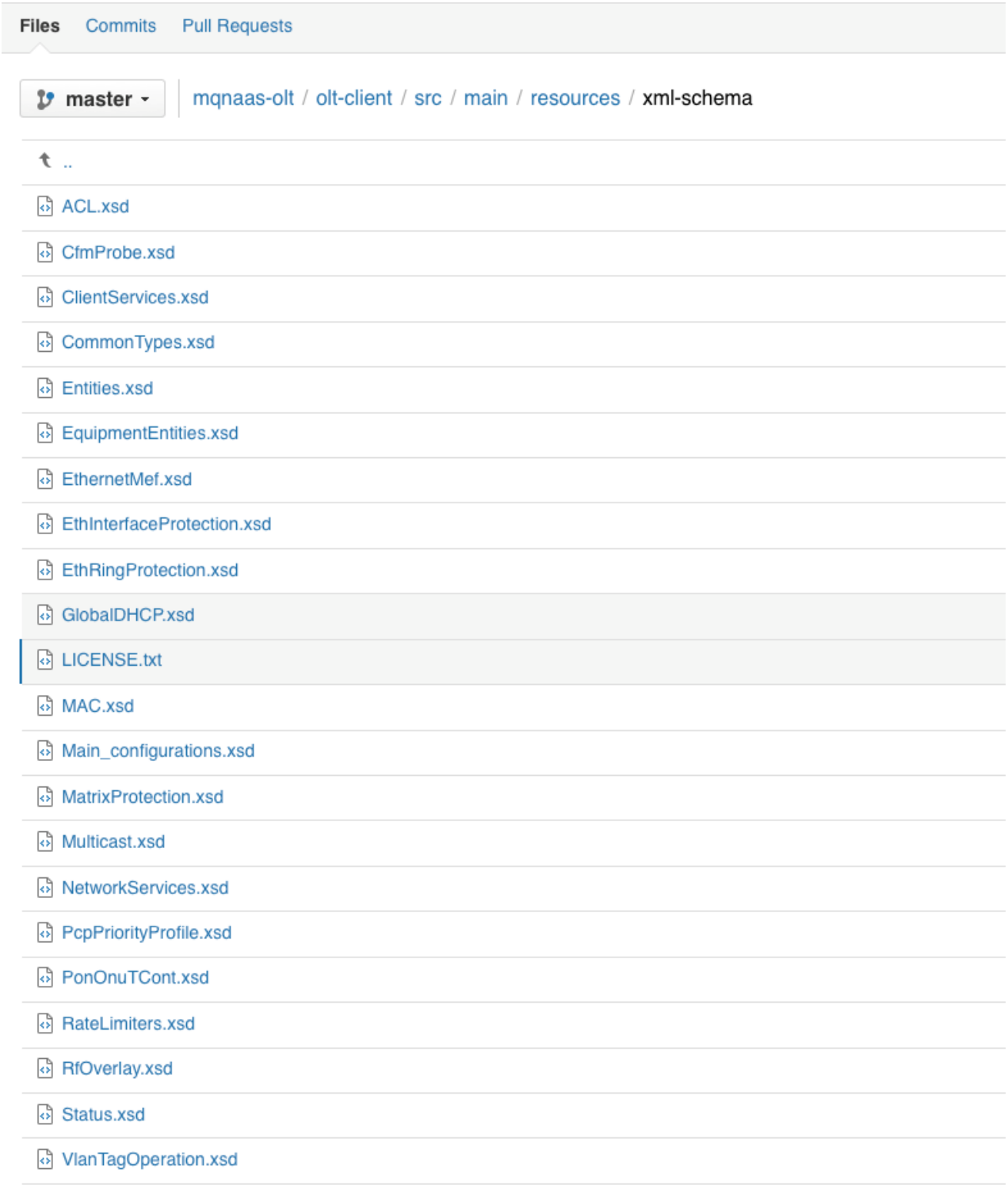

Figure 6: SODALES ARN Data Model

All these classes are directly integrated into the OpenNaaS release by means of the Java classes. Follow an example of the Java class for the Equipment. We have not included in the report all the classes for the sake of readability.

```
package net.i2cat.dana.ptin_olt.client.api.model;

import java.math.BigInteger;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlSchemaType;
import javax.xml.bind.annotation.XmlType;

/**
 *
 *         Explanaition of each atribute:
 *            id:
 *            type:
 *            admin:
 *            status:
 *            accessType:      0 the equipment is managed directly.
 *                             1 the equipment is managed by means of a intermediate
equipment (e.g ONU is configured by the OLT).
 *            accessSlot:      The OLT slot ID of the respective card where the ONU is
connected
 *            accessInterface: The OLT PON interface ID where the ONU is connected
 *            accessIndex:     The ONU ID
 *            numSlots:        The maximum number of Slots on the OLT.
 *            command:         Integer that identifies a command. Used for config
operations only. Does not appear in the XML response for show.
 *
 *
 * <p>Java class for Equipment complex type.
 *
 * <p>The following schema fragment specifies the expected content contained within
this class.
 *
 * <pre>
 * &lt;complexType name="Equipment">
 *   &lt;complexContent>
 *     &lt;restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;all>
 *         &lt;element name="system" type="{}Tsystem" minOccurs="0"/>
 *         &lt;element name="cardList" type="{}CardList" minOccurs="0"/>
 *         &lt;element name="fecCountersUpstream" type="{}TfecCountersUp"
minOccurs="0"/>
 *         &lt;element name="fecCountersDownstream" type="{}TfecCountersDown"
minOccurs="0"/>
 *         &lt;element name="multicast" type="{}Tmulticast" minOccurs="0"/>
```

```
 *         &lt;element name="firmware" type="{}Firmware" minOccurs="0"/>
 *         &lt;element name="networkServiceList" type="{}TnetworkServiceList"
minOccurs="0"/>
 *         &lt;element name="lagList" type="{}TinterfaceList" minOccurs="0"/>
 *         &lt;element name="synchronism" type="{}TsynchronismEntity" minOccurs="0"/>
 *         &lt;element name="clientServiceList" type="{}TclientServiceList"
minOccurs="0"/>
 *         &lt;element name="swUpgrade" type="{}TswUpgrade" minOccurs="0"/>
 *         &lt;element name="ontExtensions" type="{}TontExtensions" minOccurs="0"/>
 *         &lt;element name="igmpCounters" type="{}Tigmp_v3_Counters" minOccurs="0"/>
 *       &lt;/all>
 *       &lt;attribute name="id" type="{}Tid" />
 *       &lt;attribute name="type" type="{}TequipmentType" />
 *       &lt;attribute name="admin" type="{}Tadmin" />
 *       &lt;attribute name="status" type="{}TentityStatus" />
 *       &lt;attribute name="accessType" type="{}TaccessType" />
 *       &lt;attribute name="accessSlot"
type="{http://www.w3.org/2001/XMLSchema}unsignedInt" />
 *       &lt;attribute name="accessInterface" type="{}Tid" />
 *       &lt;attribute name="accessIndex"
type="{http://www.w3.org/2001/XMLSchema}unsignedInt" />
 *       &lt;attribute name="numSlots"
type="{http://www.w3.org/2001/XMLSchema}unsignedInt" />
 *       &lt;attribute name="command" type="{}TequipmentCommand" />
 *       &lt;attribute name="cardId" type="{}Tid" />
 *       &lt;attribute name="interfaceId" type="{}Tid" />
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 *
 */
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "Equipment", propOrder = {

})
public class Equipment {

    protected Tsystem system;
    protected CardList cardList;
    protected TfecCountersUp fecCountersUpstream;
    protected TfecCountersDown fecCountersDownstream;
    protected Tmulticast multicast;
    protected Firmware firmware;
```

```
    protected TnetworkServiceList networkServiceList;
    protected TinterfaceList lagList;
    protected TsynchronismEntity synchronism;
    protected TclientServiceList clientServiceList;
    protected TswUpgrade swUpgrade;
    protected TontExtensions ontExtensions;
    protected TigmpV3Counters igmpCounters;
    @XmlAttribute(name = "id")
    protected Long id;
    @XmlAttribute(name = "type")
    protected BigInteger type;
    @XmlAttribute(name = "admin")
    protected Integer admin;
    @XmlAttribute(name = "status")
    protected BigInteger status;
    @XmlAttribute(name = "accessType")
    protected BigInteger accessType;
    @XmlAttribute(name = "accessSlot")
    @XmlSchemaType(name = "unsignedInt")
    protected Long accessSlot;
    @XmlAttribute(name = "accessInterface")
    protected Long accessInterface;
    @XmlAttribute(name = "accessIndex")
    @XmlSchemaType(name = "unsignedInt")
    protected Long accessIndex;
    @XmlAttribute(name = "numSlots")
    @XmlSchemaType(name = "unsignedInt")
    protected Long numSlots;
    @XmlAttribute(name = "command")
    protected Integer command;
    @XmlAttribute(name = "cardId")
    protected Long cardId;
    @XmlAttribute(name = "interfaceId")
    protected Long interfaceId;

    /**
     * Gets the value of the system property.
     *
     * @return
     *     possible object is
     *     {@link Tsystem }
     *
     */
    public Tsystem getSystem() {
```

```
        return system;
    }

    /**
     * Sets the value of the system property.
     *
     * @param value
     *     allowed object is
     *     {@link Tsystem }
     *
     */
    public void setSystem(Tsystem value) {
        this.system = value;
    }

    /**
     * Gets the value of the cardList property.
     *
     * @return
     *     possible object is
     *     {@link CardList }
     *
     */
    public CardList getCardList() {
        return cardList;
    }

    /**
     * Sets the value of the cardList property.
     *
     * @param value
     *     allowed object is
     *     {@link CardList }
     *
     */
    public void setCardList(CardList value) {
        this.cardList = value;
    }

    /**
     * Gets the value of the fecCountersUpstream property.
     *
     * @return
     *     possible object is
```

```
     *     {@link TfecCountersUp }
     *
     */
    public TfecCountersUp getFecCountersUpstream() {
        return fecCountersUpstream;
    }

    /**
     * Sets the value of the fecCountersUpstream property.
     *
     * @param value
     *     allowed object is
     *     {@link TfecCountersUp }
     *
     */
    public void setFecCountersUpstream(TfecCountersUp value) {
        this.fecCountersUpstream = value;
    }

    /**
     * Gets the value of the fecCountersDownstream property.
     *
     * @return
     *     possible object is
     *     {@link TfecCountersDown }
     *
     */
    public TfecCountersDown getFecCountersDownstream() {
        return fecCountersDownstream;
    }

    /**
     * Sets the value of the fecCountersDownstream property.
     *
     * @param value
     *     allowed object is
     *     {@link TfecCountersDown }
     *
     */
    public void setFecCountersDownstream(TfecCountersDown value) {
        this.fecCountersDownstream = value;
    }
// Rest of the getters & setters have not ben included in the text.
}
```

The XML interface at the southbound allows support of simultaneous operations within the same request. Each of these operations must be identified with a unique token number, the type of operation and entity type. In turn, the corresponding response will deliver the same number of operations as a result of the requested. For example, we can consider a request that comprises two operations retrieving equipment through the ARN.

The following details the XML request and an example of a given response

Issuing the REQUEST:

```
<?xml version="1.0" encoding="utf-8"?>
<request>
 <operation token="333" type="show" entity="equipment/system">
  <equipment />
 </operation> <operation token="222" type="show" entity="equipment/system">
   <equipment id="554945"/>
 </operation>
</request>
```

Obtained RESPONSE:

```
<?xml version='1.0' encoding='utf-8'?>
<response timestamp='947991579' utcDate='2000/01/16 02:59:39' version='' >
 <operation token='333' type='show'>
  <equipmentList>
    <equipment id='0' type='10032' admin='1' status='3' accessType='0'
accessInterface='0' accessIndex='0' numSlots='45'>
       <system name='OLT360' description='OLT360' contact='' ipAddress='10.112.42.80'
rack='1' subRack='1' shelf='1' serialNum='' date='2000/01/16' time='02:59:39'
hwVersion='' fwVersion='OLT360 v2.1.0-r32' admin='1' />
     </equipment>
     <equipment id='67108865' type='10023' admin='1' status='6' accessType='1'
accessSlot='4' accessInterface='67371008' accessIndex='1' numSlots='1'>
       <system name='ONT 4.1.1' description='ONT 4.1.1' contact='' location=''
ipAddress='10.112.42.80' rack='1' subRack='1' shelf='1' serialNum='AAAAAAAAAAAAAAAA'
date='2000/01/16' time='02:59:39' hwVersion='' fwVersion='---' admin='1' />
       <ont profileId='1' swUpgradeMode='1' planedVersion='OFF'/>
      </equipment>
      <equipment id='67108919' type='10023' admin='2' status='6' accessType='1'
accessSlot='4'
accessInterface='67371008' accessIndex='55' numSlots='1'>
       <system name='ONT 4.1.55' description='ONT 4.1.55' contact='' location=''
ipAddress='10.112.42.80' rack='1' subRack='1' shelf='1' serialNum='1122334455667788'
accessSlot='0'
location=''
date='2000/01/16' time='02:59:39' hwVersion='' fwVersion='---' admin='1' />
       <ont profileId='1' swUpgradeMode='1' planedVersion='' />
```

```
      </equipment>
  </equipmentList> <result error='00000000' errorStr='OK'/>
 </operation>
 <operation token='222' type='show'>
   <result error='0A030003' errorStr='ERROR'/>
 </operation>
</response>
```

In oder to implement such a generic mechanism, we have included in the OpenNaaS a generic API in order to send requests and receives responses to the physical ARN.

```
package net.i2cat.dana.ptin_olt.client.api;

import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import net.i2cat.dana.ptin_olt.client.api.model.Request;
import net.i2cat.dana.ptin_olt.client.api.model.Response;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@Path("/")
public interface PTIN_OLT_RPCClientAPI {

      /**
       * Sends given @code{Request} to be processed, and returns appropriated
Response.
       *
       * @param request
       * @return Response for processed given request
       */
      @POST
      @Consumes(MediaType.APPLICATION_XML)
      @Produces(MediaType.APPLICATION_XML)
      public Response sendRequest(Request request);

}
```

3.2 Customer Premises Equipment

The CPE considered in SODALES is the device developed and provided by Ethernity Networks, based on the enet-38xx motherboard. The enet-38xx access flow processor is a family of high performance configurable flow processor and traffic manager solutions optimized for the Metro access market. Enet-38xx is specially focused on carrier Ethernet Access. It comes with many interface and device options. It is a multi-service fabric flow processor (FFP) integrating packet processing, protocol interworking, traffic management, Ethernet, and Layer 2/3/4 switch.

The original API of the CPE provides a rich amount of operations to be performed over the different internal components of the device. For confidentiality details the schematic blocks of the device are not included in the deliverable. However, the API was only available by means of direct connection to the equipment through the Telnet protocol. Thus, in order to be controlled and managed through the SODALES software it has been required to implement the web service API in the device.

All the code listed in this section can be completely found in the corresponding folder of the aforementioned software repository.

The web service provides direct access to all the required components within the device. The communication is performed via REST web services. The following are two examples of XML files exchanged representing the model of the CPE. To avoid repetition the rest of created XML files are contained in Appendix D.

serviceVlanDetail.xml

```
<meaGetServiceVlanId xmlns="http://www.ethernitynet.com/enet/ServiceParams">
  <serviceDataBase>
    <unit>0</unit>
    <serviceId>6</serviceId>
    <policerId>2</policerId>
    <pmId>3</pmId>
    <eIngressType>1</eIngressType>
    <outer_vlanId>10</outer_vlanId>
    <inner_vlanId>0</inner_vlanId>
    <src_port>105</src_port>
    <serviceOutputPort>
      <cluster>
        <clusterId>104</clusterId>
      </cluster>
    </serviceOutputPort>
    <vlanEdit_flowtype>2</vlanEdit_flowtype>
    <vlanEdit_outer_command>3</vlanEdit_outer_command>
    <vlanEdit_outer_vlan>20</vlanEdit_outer_vlan>
    <vlanEdit_inner_command>0</vlanEdit_inner_command>
    <vlanEdit_inner_vlan>0</vlanEdit_inner_vlan>
  </serviceDataBase>
</meaGetServiceVlanId>
```

meaPmCounter.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<meaPmCounter xmlns="http://www.ethernitynet.com/enet/PmCounter">
  <PmCounter>
    <pmId>10</pmId>
    <FwdGreen-pkts>4862959</FwdGreen-pkts>
    <FwdGreen-bytes>4862958000</FwdGreen-bytes>
    <FwdYellow-pkts>0</FwdYellow-pkts>
    <FwdYellow-bytes>0</FwdYellow-bytes>
    <DisGreen-pkts>0</DisGreen-pkts>
    <DisGreen-bytes>0</DisGreen-bytes>
    <DisYellow-pkts>0</DisYellow-pkts>
    <DisYellowRed-bytes>0</DisYellowRed-bytes>
    <DisRed-pkts>0</DisRed-pkts>
    <DisOther>0</DisOther>
    <DisMtu>0</DisMtu>
    <FwdGreenRate>29920000</FwdGreenRate>
    <FwdYellowRate>0</FwdYellowRate>
  </PmCounter>
</meaPmCounter>
```

The CPE client implemented in the southbound is capable of exchanging XML messages with the physical device, and thus provides a means of configuration, monitoring, and management for the device. The CPE itself has been enhanced to support Web Services, which is a clear innovation towards including the CPE hardware in open access enabled environments, so it facilitates the management operations through the new supported technologies.

The internal client API used for the communication follows:

```
package net.i2cat.enet38xx.client;

import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;

import net.i2cat.enet38xx.client.model.MeaEgressPortInfo;
import net.i2cat.enet38xx.client.model.MeaGetServiceVlanId;
import net.i2cat.enet38xx.client.model.MeaIngressPortInfo;
import net.i2cat.enet38xx.client.model.MeaPmCounter;
import net.i2cat.enet38xx.client.model.MeaPolicerAllProfiles;
import net.i2cat.enet38xx.client.model.MeaPolicerProfile;
```

```
import net.i2cat.enet38xx.client.model.MeaPortMapping;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@Path("/")
public interface Enet38xxAPI {

	//port management

	/**
     * Retrieves all ports in given unit.
     *
     * GET http://ADDR:8080/meaPortMapping.xml?unit=0
     *
     * @param unitId
     * @return Collection of port ids in given unit, elements in the collection contain both
the unitid and the portid
     */
	@Path("meaPortMapping.xml")
	@GET
    MeaPortMapping getAllPorts(@QueryParam("unit") int unitId);

	/**
	 *
	 * GET http://ADDR:8080/meaIngressPortInfo.xml?unit=unitId&port_id=portId
	 *
	 * @param unitId
	 * @param portId
	 * @return
	 */
	@Path("ingress_port_setting.xml")
	@GET
    MeaIngressPortInfo getIngressPortInfo(@QueryParam("unit") int unitId,
@QueryParam("port_id") int portId);

	/**
	 *
	 * GET http://ADDR:8080/meaEgressPortInfo.xml?unit=unitId&port_id=portId
	 *
	 * @param unitId
	 * @param portId
	 * @return
	 */
	@Path("egress_port_setting.xml")
	@GET
```

```
    MeaEgressPortInfo getEgressPortInfo(@QueryParam("unit") int unitId, @QueryParam("port_id")
int portId);

       /**
        *
        * POST
http://ADDR:8080/ingress_port_setting.html?unit=0&port_id=104&rx_enable=1&crc_check=1&My_mac=00
:11:22:33:44:55:66
        *
        * @param unit
        * @param port_id
        * @param rx_enable
        * @param crc_check
        * @param My_mac
        */
       @Path("ingress_port_setting.html")
       @POST
    void configureIngressPort(@QueryParam("unit") int unit, @QueryParam("port_id") int port_id,
@QueryParam("rx_enable") Integer rx_enable, @QueryParam("crc_check") Integer crc_check,
@QueryParam("My_mac") String My_mac);

       /**
        *
        * POST
http://ADDR:8080/egress_port_setting.html?unit=0&port_id=104&tx_enable=1&crc_check=1
        *
        * @param unit
        * @param port_id
        * @param tx_enable
        * @param crc_check
        */
       @Path("egress_port_setting.html")
       @POST
    void configureEgressPort(@QueryParam("unit") int unit, @QueryParam("port_id") int port_id,
@QueryParam("tx_enable") Integer tx_enable, @QueryParam("crc_check") Integer crc_check);

    // policer profiles management

    /**
     * Retrieves the id of all policer profiles configured in given unit
     *
     * @param unitId
     * @return Collection of policer profile ids in given unit.
     */
    MeaPolicerAllProfiles getPolicerProfiles(@QueryParam("unit") int unitId);

    MeaPolicerProfile getPolicerProfile(@QueryParam("unit") int unitId,
@QueryParam("profileId") int profileId);
```

```
    /**
     *
     * POST
http://ADDR:8080/policerProfile.html?unit=0&profileId=3&CIR=100000000&EIR=0&&CBS=64000&EBS=0&gn
_type=2
     *
     * @param unitId
     * @param profileId
     * @param type
     * @param cir
     * @param cbs
     * @param eir
     * @param ebs
     * @param gn_sign
     * @param gn_type
     * @param comp
     * @param color
     * @return
     */
    @Path("policerProfile.html")
    @POST
    MeaPolicerProfile createPolicerProfile(@QueryParam("unit") int unitId,
@QueryParam("profileId") int profileId,
                @QueryParam("Type") int type, @QueryParam("CIR") long cir, @QueryParam("CBS")
long cbs, @QueryParam("EIR") long eir, @QueryParam("EBS") long ebs,
                @QueryParam("gn_sign") int gn_sign, @QueryParam("gn_type") int gn_type,
@QueryParam("comp") int comp, @QueryParam("color") int color);


    void removePolicerProfile(@QueryParam("unit") int unitId, @QueryParam("profileId") int
profileId);


    // service vlan management


    /**
     * Retrieves the id of all services vlan configured in given unit
     *
     * @param unitId
     * @return Collection of services vlan ids in given unit.
     */
    List<String> getServicesVlan(@QueryParam("unit") int unitId);


    MeaGetServiceVlanId getServiceVlan(@QueryParam("unit") int unitId, @QueryParam("serviceId")
int serviceId);


    /**
     *
     * POST
http://ADDR:8080/createServiceVlan.html?unit=0&serviceId=6&srcPort=105&policerId=2&pmId=3&eIngr
essType=1&outer_vlanId=10&clusterId=104&vlanEdit_flowtype=2&vlanEdit_outer_command=3&vlanEdit_o
uter_vlan=20
```

```
     *
     * @param unitId
     * @param serviceId
     * @param srcPort
     * @param policerId
     * @param pmId
     * @param eIngressType
     * @param outer_vlanId
     * @param clusterId
     * @param vlanEdit_flowtype
     * @param vlanEdit_outer_command
     * @param vlanEdit_outer_vlan
     * @param vlanEdit_inner_command
     * @param vlanEdit_inner_vlan
     * @return
     */
    @Path("createServiceVlan.html")
    @POST
    MeaGetServiceVlanId createServiceVlan(@QueryParam("unit") int unitId,
                @QueryParam("serviceId") int serviceId, @QueryParam("srcPort") int srcPort,
@QueryParam("policerId") int policerId, @QueryParam("pmId") int pmId,
                @QueryParam("eIngressType") int eIngressType, @QueryParam("outer_vlanId") int
outer_vlanId, @QueryParam("clusterId") int clusterId,
                @QueryParam("vlanEdit_flowtype") int vlanEdit_flowtype,
@QueryParam("vlanEdit_outer_command") int vlanEdit_outer_command,
                @QueryParam("vlanEdit_outer_vlan") int vlanEdit_outer_vlan,
@QueryParam("vlanEdit_inner_command") int vlanEdit_inner_command,
@QueryParam("vlanEdit_inner_vlan") int vlanEdit_inner_vlan);

    void removeServiceVlan(@QueryParam("unit") int unitId, @QueryParam("serviceId") int
serviceId);

    // performance monitoring management

    /**
     * Retrieves the id of all performance monitoring counters configured in given unit
     *
     * @param unitId
     * @return Collection of performance monitoring counter ids in given unit.
     */
    List<String> getPerformanceMonitoringCounters(@QueryParam("unit") int unitId);

    /**
     *
     * GET http://ADDR:8080/meaPmCounter.xml?unit=0&pmId=3
     *
     * @param unitId
     * @param pmId
     * @return
```

```
     */
    @Path("meaPmCounter.xml")
    @GET
    MeaPmCounter getPerformanceMonitoringCounter(@QueryParam("unit") int unitId,
@QueryParam("pmId") int pmId);

    void clearPerformanceMonitoringCounter(@QueryParam("unit") int unitId, @QueryParam("pmId")
int pmId);
}
```

The CPE client is then responsible for creating the data model within the SODALES software implemented. The model is directly created from the XML files, and transformed to software objects that can be later on programmed and manipulated.

meaGetServiceVlanId.java

```
package net.i2cat.enet38xx.client.model;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@XmlRootElement(name="meaGetServiceVlanId",
namespace="http://www.ethernitynet.com/enet/ServiceParams")
@XmlAccessorType(XmlAccessType.FIELD)
public class MeaGetServiceVlanId {

       private ServiceDataBase serviceDataBase;

       public ServiceDataBase getServiceDataBase() {
              return serviceDataBase;
       }

       public void setServiceDataBase(ServiceDataBase serviceDataBase) {
              this.serviceDataBase = serviceDataBase;
       }

       @XmlType(namespace="http://www.ethernitynet.com/enet/ServiceParams", propOrder={"unit",
"serviceId", "policerId", "pmId", "eIngressType",
                     "outer_vlanId", "inner_vlanId", "src_port", "serviceOutputPort",
"vlanEdit_flowtype", "vlanEdit_outer_command", "vlanEdit_outer_vlan",
                     "vlanEdit_inner_command", "vlanEdit_inner_vlan"})
```

```
	@XmlAccessorType(XmlAccessType.FIELD)
	class ServiceDataBase {

		private int unit;
		private int serviceId;
		private int policerId;
		private int pmId;
		private int eIngressType;
		private int outer_vlanId;
		private int inner_vlanId;
		private int src_port;
		private ServiceOutputPort serviceOutputPort;
		private int vlanEdit_flowtype;
		private int vlanEdit_outer_command;
		private int vlanEdit_outer_vlan;
		private int vlanEdit_inner_command;
		private int vlanEdit_inner_vlan;

		public int getUnit() {
			return unit;
		}

		public void setUnit(int unit) {
			this.unit = unit;
		}

		public int getServiceId() {
			return serviceId;
		}

		public void setServiceId(int serviceId) {
			this.serviceId = serviceId;
		}

		/**
		 *
		 * @return the identifier of the @link{MeaPolicerProfile} this Service is linked
to.
		 */
		public int getPolicerId() {
			return policerId;
		}

		public void setPolicerId(int policerId) {
			this.policerId = policerId;
		}

		/**
```

```
		 *
		 * @return the identifier of the @link{MeaPmCounter} monitoring traffic in this
service
		 */
		public int getPmId() {
			return pmId;
		}

		public void setPmId(int pmId) {
			this.pmId = pmId;
		}

		/**
		 *
		 * @return how many vlans are allowed
		 */
		public int geteIngressType() {
			return eIngressType;
		}

		public void seteIngressType(int eIngressType) {
			this.eIngressType = eIngressType;
		}

		public int getOuter_vlanId() {
			return outer_vlanId;
		}

		public void setOuter_vlanId(int outer_vlanId) {
			this.outer_vlanId = outer_vlanId;
		}

		public int getInner_vlanId() {
			return inner_vlanId;
		}

		public void setInner_vlanId(int inner_vlanId) {
			this.inner_vlanId = inner_vlanId;
		}

		public int getSrc_port() {
			return src_port;
		}

		public void setSrc_port(int src_port) {
			this.src_port = src_port;
		}
```

```
public ServiceOutputPort getServiceOutputPort() {
	return serviceOutputPort;
}

public void setServiceOutputPort(ServiceOutputPort serviceOutputPort) {
	this.serviceOutputPort = serviceOutputPort;
}

public int getVlanEdit_flowtype() {
	return vlanEdit_flowtype;
}

public void setVlanEdit_flowtype(int vlanEdit_flowtype) {
	this.vlanEdit_flowtype = vlanEdit_flowtype;
}

public int getVlanEdit_outer_command() {
	return vlanEdit_outer_command;
}

public void setVlanEdit_outer_command(int vlanEdit_outer_command) {
	this.vlanEdit_outer_command = vlanEdit_outer_command;
}

public int getVlanEdit_outer_vlan() {
	return vlanEdit_outer_vlan;
}

public void setVlanEdit_outer_vlan(int vlanEdit_outer_vlan) {
	this.vlanEdit_outer_vlan = vlanEdit_outer_vlan;
}

public int getVlanEdit_inner_command() {
	return vlanEdit_inner_command;
}

public void setVlanEdit_inner_command(int vlanEdit_inner_command) {
	this.vlanEdit_inner_command = vlanEdit_inner_command;
}

public int getVlanEdit_inner_vlan() {
	return vlanEdit_inner_vlan;
}

public void setVlanEdit_inner_vlan(int vlanEdit_inner_vlan) {
	this.vlanEdit_inner_vlan = vlanEdit_inner_vlan;
}
```

```
		@XmlType(namespace="http://www.ethernitynet.com/enet/ServiceParams")
		@XmlAccessorType(XmlAccessType.FIELD)
		class ServiceOutputPort {
			private Cluster cluster;

			public Cluster getCluster() {
				return cluster;
			}

			public void setCluster(Cluster cluster) {
				this.cluster = cluster;
			}

			@XmlType(namespace="http://www.ethernitynet.com/enet/ServiceParams")
			@XmlAccessorType(XmlAccessType.FIELD)
			class Cluster {
				private int clusterId;

				public int getClusterId() {
					return clusterId;
				}

				public void setClusterId(int clusterId) {
					this.clusterId = clusterId;
				}
			}
		}
	}
}
```

meaPmCounter.java

```
package net.i2cat.enet38xx.client.model;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@XmlRootElement(namespace="http://www.ethernitynet.com/enet/PmCounter")
@XmlAccessorType(XmlAccessType.FIELD)
```

```
public class MeaPmCounter {

	private PmCounter pmCounter;

	public PmCounter getPmCounter() {
		return pmCounter;
	}

	public void setPmCounter(PmCounter pmCounter) {
		this.pmCounter = pmCounter;
	}

	/**
	 *
	 * @author Isart Canyameres Gimenez (i2cat)
	 *
	 */
	@XmlAccessorType(XmlAccessType.FIELD)
	@XmlType(namespace="http://www.ethernitynet.com/enet/PmCounter", propOrder={"pmId",
"fwdGreen_pkts", "fwdGreen_bytes", "fwdGreen_bytes", "fwdYellow_pkts", "fwdYellow_bytes",
"disGreen_pkts", "disGreen_bytes", "disYellow_pkts", "disYellowRed_bytes", "disRed_pkts",
"disOther", "disMtu", "fwdGreenRate", "fwdYellowRate"})
	class PmCounter {

		private int pmId;
		private long fwdGreen_pkts;
		private long fwdGreen_bytes;
		private long fwdYellow_pkts;
		private long fwdYellow_bytes;
		private long disGreen_pkts;
		private long disGreen_bytes;
		private long disYellow_pkts;
		private long disYellowRed_bytes;
//		private long disYellow_bytes;
		private long disRed_pkts;
//		private long disRed_bytes;
		private long disOther;
		private long disMtu;
		private long fwdGreenRate;
		private long fwdYellowRate;

		public int getPmId() {
			return pmId;
		}

		public void setPmId(int pmId) {
			this.pmId = pmId;
		}
```

```
public long getFwdGreen_pkts() {
	return fwdGreen_pkts;
}

public void setFwdGreen_pkts(long fwdGreen_pkts) {
	this.fwdGreen_pkts = fwdGreen_pkts;
}

public long getFwdGreen_bytes() {
	return fwdGreen_bytes;
}

public void setFwdGreen_bytes(long fwdGreen_bytes) {
	this.fwdGreen_bytes = fwdGreen_bytes;
}

public long getFwdYellow_pkts() {
	return fwdYellow_pkts;
}

public void setFwdYellow_pkts(long fwdYellow_pkts) {
	this.fwdYellow_pkts = fwdYellow_pkts;
}

public long getFwdYellow_bytes() {
	return fwdYellow_bytes;
}

public void setFwdYellow_bytes(long fwdYellow_bytes) {
	this.fwdYellow_bytes = fwdYellow_bytes;
}

public long getDisGreen_pkts() {
	return disGreen_pkts;
}

public void setDisGreen_pkts(long disGreen_pkts) {
	this.disGreen_pkts = disGreen_pkts;
}

public long getDisGreen_bytes() {
	return disGreen_bytes;
}

public void setDisGreen_bytes(long disGreen_bytes) {
	this.disGreen_bytes = disGreen_bytes;
```

```
	}

	public long getDisYellow_pkts() {
		return disYellow_pkts;
	}

	public void setDisYellow_pkts(long disYellow_pkts) {
		this.disYellow_pkts = disYellow_pkts;
	}

	public long getDisYellowRed_bytes() {
		return disYellowRed_bytes;
	}

	public void setDisYellowRed_bytes(long disYellowRed_bytes) {
		this.disYellowRed_bytes = disYellowRed_bytes;
	}

	public long getDisRed_pkts() {
		return disRed_pkts;
	}

	public void setDisRed_pkts(long disRed_pkts) {
		this.disRed_pkts = disRed_pkts;
	}

	public long getDisOther() {
		return disOther;
	}

	public void setDisOther(long disOther) {
		this.disOther = disOther;
	}

	public long getDisMtu() {
		return disMtu;
	}

	public void setDisMtu(long disMtu) {
		this.disMtu = disMtu;
	}

	public long getFwdGreenRate() {
		return fwdGreenRate;
	}

	public void setFwdGreenRate(long fwdGreenRate) {
```

```
                        this.fwdGreenRate = fwdGreenRate;
                }

                public long getFwdYellowRate() {
                        return fwdYellowRate;
                }

                public void setFwdYellowRate(long fwdYellowRate) {
                        this.fwdYellowRate = fwdYellowRate;
                }
        }
}
```

With the Web Service in place, and the data model abstracted within the SODALES software layer, the operation of the CPE is enabled. For example, in order to set the ingress and egress ports on the device, it is required to perform the following operation:

```
Ingress port : ingress_port_setting.html? port_id=105& rx_enable=1&& crc_check=1
Egress port: egress_port_setting.html? port_id=105& tx_enable=1& crc_check=1
```

This will enable port 105 to be in receive and transmit mode correspondingly. In the ingress port there are some optional fields for CFM/OAM, which include the possibility to configure the MAC address per interface.

Once the data model is incorporated, we have implemented a functional northbound API to be used both internally for the reservation capability (later explained) and in order to enable operation over the created resources. The northbound interface created is equal for both the physical CPE and the virtual ones created on top of it to enable the open access feature. It is divided into four major components:

- Port management
- Profiles management
- Service VLAN management
- Performance monitoring

The interface specification follows

```
package net.i2cat.enet38xx.client.nortbound;

import java.util.List;

import net.i2cat.enet38xx.client.model.MeaEgressPortInfo;
import net.i2cat.enet38xx.client.model.MeaGetServiceVlanId;
import net.i2cat.enet38xx.client.model.MeaIngressPortInfo;
import net.i2cat.enet38xx.client.model.MeaPmCounter;
```

```
import net.i2cat.enet38xx.client.model.MeaPolicerAllProfiles;
import net.i2cat.enet38xx.client.model.MeaPolicerProfile;
import net.i2cat.enet38xx.client.model.MeaPortMapping;

/**
 *
 * @author i2CAT
 *
 */
public interface Enet38xxClientAPI {

      //port management

      /**
     * Retrieves all ports in given unit.
     *
     * @param unitId
     * @return Collection of port ids in given unit, elements in the collection
contain both the unitid and the portid
     */
    MeaPortMapping getAllPorts(String unitId);
    MeaIngressPortInfo getIngressPortInfo(String unitId, String portId);
    MeaEgressPortInfo getEgressPortInfoString(String unitId, String portId);
    void configureIngressPort(MeaIngressPortInfo ingressPortInfo);
    void configureEgressPort(MeaEgressPortInfo egressPortInfo);

    // policer profiles management

    /**
     * Retrieves the id of all policer profiles configured in given unit
     *
     * @param unitId
     * @return Collection of policer profile ids in given unit.
     */
    MeaPolicerAllProfiles getPolicerProfiles(String unitId);
    MeaPolicerProfile getPolicerProfile(String unitId, String profileId);
    MeaPolicerProfile createPolicerProfile(String unitId, String profileId,
MeaPolicerProfile profile);
    void removePolicerProfile(String unitId, String profileId);

    // service vlan management

    /**
     * Retrieves the id of all services vlan configured in given unit
```

```
     *
     * @param unitId
     * @return Collection of services vlan ids in given unit.
     */
    List<String> getServicesVlan(String unitId);
    MeaGetServiceVlanId getServiceVlan(String unitId, String serviceId);
    MeaGetServiceVlanId createServiceVlan(String unitId, MeaGetServiceVlanId
serviceVlanDetail);
    void removeServiceVlan(String unitId, String serviceId);

    // performance monitoring management

    /**
     * Retrieves the id of all performance monitoring counters configured in given
unit
     *
     * @param unitId
     * @return Collection of performance monitoring counter ids in given unit.
     */
    List<String> getPerformanceMonitoringCounters(String unitId);
    MeaPmCounter getPerformanceMonitoringCounter(String unitId, String pmId);
    void clearPerformanceMonitoringCounter(String unitId, String pmId);

}
```

3.3 Open Access

Open Access is one of the key features of the SODALES project. Basically, open access implies that the management plane is capable of creating isolated slices of the SODALES physical infrastructure. These slices are then offered towards the actual service provider. The actual network service is thus decoupled from the actual infrastructure where it is provisioned, following the Open Access Network model.

While virtualization is a well-studied, and well-known project in networking research environments [Chowdury, Botero], the solution is not yet widely deployed in production environments. In fact, there is no yet network virtualization in any access network. Several solutions are emerging at the moment jointly with the SDN principles, but most of them are focused on the OpenFlow protocol. For example, the de-facto standard SDN controller, Opendaylight [ODL] provides a software bundle, the so-called Virtual Tenant Network, which is responsible to create slices to be granted to each tenant. However, the VTN component is focused on OpenFlow, and at the moment it only works with this protocol.

There is another component in the Opendaylight controller devoted to the creation of overlays for data centre networks, the IBM-developed OpenDove component [OPENDOVE], which mainly works with GRE tunnels at the physical level.

In order to implement the open access component, the SODALES management plane has re-used (and re-defined) the reservation capability of the OpenNaaS system, which provides a service for reserving in advance slices of the infrastructure, following the Infrastructure as a Service paradigm, full compatible with the open access model. Basically, as an infrastructure provider, the administrator wants to be able of reserving any type of physical resource and lease the virtual representations to any service provider.

Figure below depicts how the open access model is implemented through the management plane.

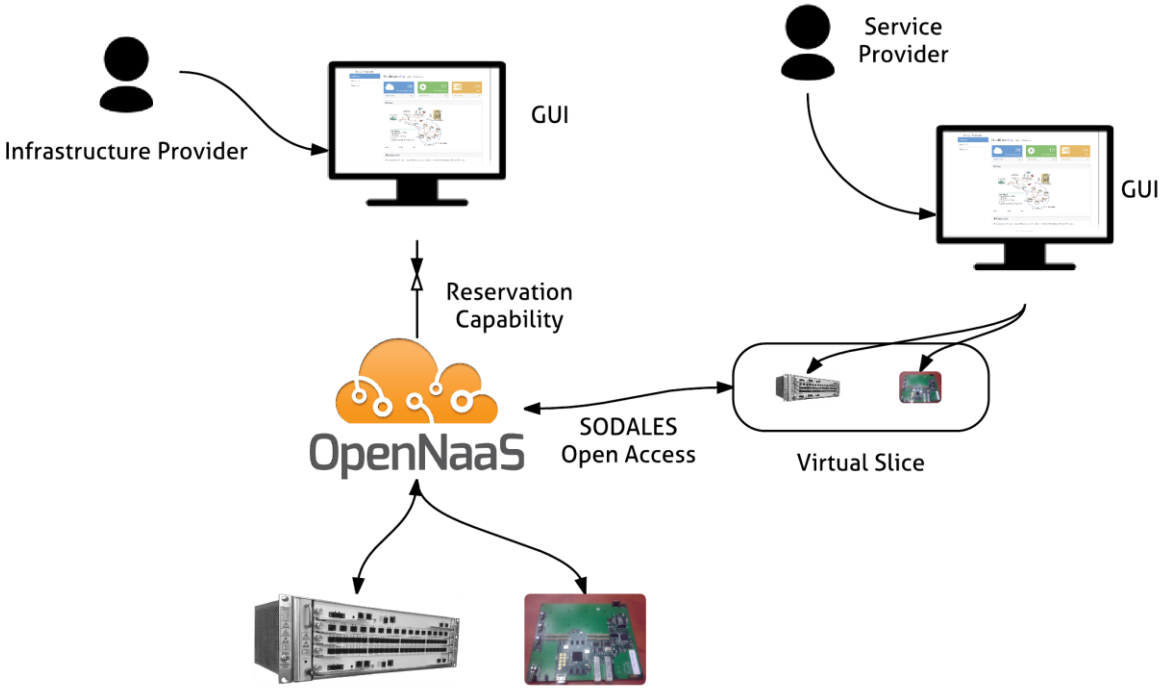

Figure 7: SODALES Open Access Overview, through the reservation Capability

The ARN and the CPE has been already registered into the system, which is now aware of all the details of each device, and the data models have been correctly instantiated. Then the infrastructure provider, by means of the SODALES graphical user interface, accesses the system through the reservation capability. He/She selects the physical components that wants to allocate to a given virtual slices (that will be later leased to a given service provider).

Once the selection process has finished, the reservation is created. At start time, the virtual slice is instantiated within OpenNaaS, creating all the corresponding APIs for the virtual resources. Then, in case the service provider wants to start providing services (i.e. configuring the devices), it is only required to access the graphical user interface and start performing the indicated actions. It is worth to mention at this stage that the graphical user interface

Figure below depicts the internal components of the SODALES software devoted to the management of the open access feature, through the reservation capability (by means of the aforementioned workflow). The internal communications between the OpenNaaS components are performed through Open Services Gateway Alliance (OSGi) services, i.e. services communicating

within the same Java Virtual Machine (JVM), while the external communications between the reservation capability and the graphical user interface are performed through REST web services.

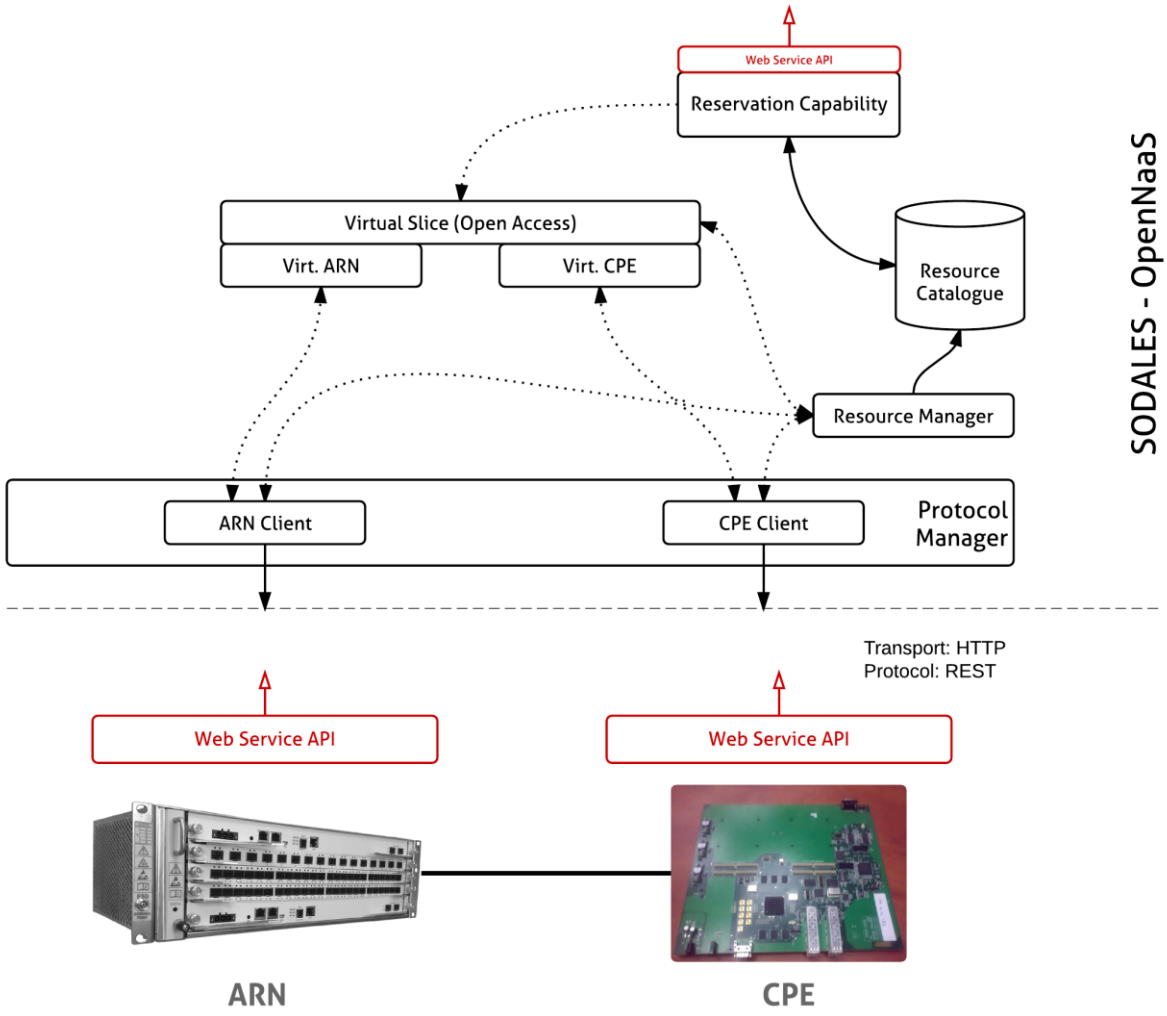

Figure 8: SODALES reservation capability internals

The reservation capability contains basic operations around the reservation concept:

- create reservation
- cancel reservation
- perform or get reservation

The complete specification of the reservation capability follows

```
package net.i2cat.dana.mqnaas.capability.reservation;
```

```
import java.util.Set;

import
net.i2cat.dana.mqnaas.capability.reservation.exception.ResourceReservationException;
import net.i2cat.dana.mqnaas.capability.reservation.model.Device;
import net.i2cat.dana.mqnaas.capability.reservation.model.Period;
import net.i2cat.dana.mqnaas.capability.reservation.model.Reservation;

import org.mqnaas.core.api.ICapability;

/**
 * <p>
 * Capability for reserving devices.
 * </p>
 * <p>
 * This interface provides the methods to be implemented for reserving {@link Device
devices} during a specific {@link Period} of time.
 * </p>
 *
 * @author i2CAT
 *
 */
public interface IReservationCapability extends ICapability {

	public Reservation createReservation(Set<Device> devices, Period period) throws
ResourceReservationException;
	public void cancelReservation(Reservation reservation) throws
ResourceReservationException;
	public Set<Reservation> getReservations();
	public void performReservation(Reservation reservation) throws
ResourceReservationException;
	public void releaseReservation(Reservation reservation) throws
ResourceReservationException;
}
```

It is important also to remark that the generic reservation capability definition has been created for SODALES, but it is being currently used in another projects (e.g. the European FP7 CONTENT project [CONTENT-FP7]). The specific implementation is customised for each project, however, the generic interface can be re-used along any system aiming at providing virtual slices to different service providers or virtual operators. This is one of the key advantages on using an existing open-source framework where different requirements can be covered by different implementations using the same generic interfaces.

The reservation capability provides two different states for a given reservation once that it has been created: planned, and reserved.

```
public enum ReservationState {

	PLANNED,
	RESERVED
}
```

This two-step instantiation process enables the management of advance reservation capabilities. The first stated planned, means that the reservation has been successfully created and planned, but the virtual resources are not yet instantiated (and so they cannot yet be used) within the Open network as a service framework.

A reservation (thus a virtual slice) is defined by the state of the reservation, the identifier, the set of devices contained in the reservation (either virtual or physical), and the period for which the reservation should be valid. The complete class, which provides the specification of a Reservation is provided in Appendix A, jointly with the Reservation Request, which is utilized in order to request for a given virtual slice. The reservation request for SODALES comes from the Web-enabled user interface, where the infrastructure provider selects the set of devices (or parts of the devices) that will compose the slice for a given service provider.

```
public class Reservation implements Identifiable {
	ReservationState  reservationState;
	Set<Device>             devices;
	Set<IRootResource>      resources;
	Period                        period;
	String                        id;
```

The implementation of the reservation concept is materialised through the generic slicing capability implemented within the management framework, which enables the creation of virtual resources from a given physical resource. The slicing capability is built over the concept of the slice management units (SLUs). An SLU is the minimum portion of a given resource in which the resource can be partitioned. For example, a L2 link can be partitioned in different VLANs, from 0 to 4096. Thus, each VLAN (i.e. VLAN 1, VLAN 2, etc.) represents and slice management unit.

The slicing capability and the slicing units are the generic components of the OpenNaaS framework that are used to provide open access features to the SODALES. Figure below details the concept of slicing unit from the SODALES perspective, and how the ARN (for example) can be partitioned using the abstract data model and the SLU concept.

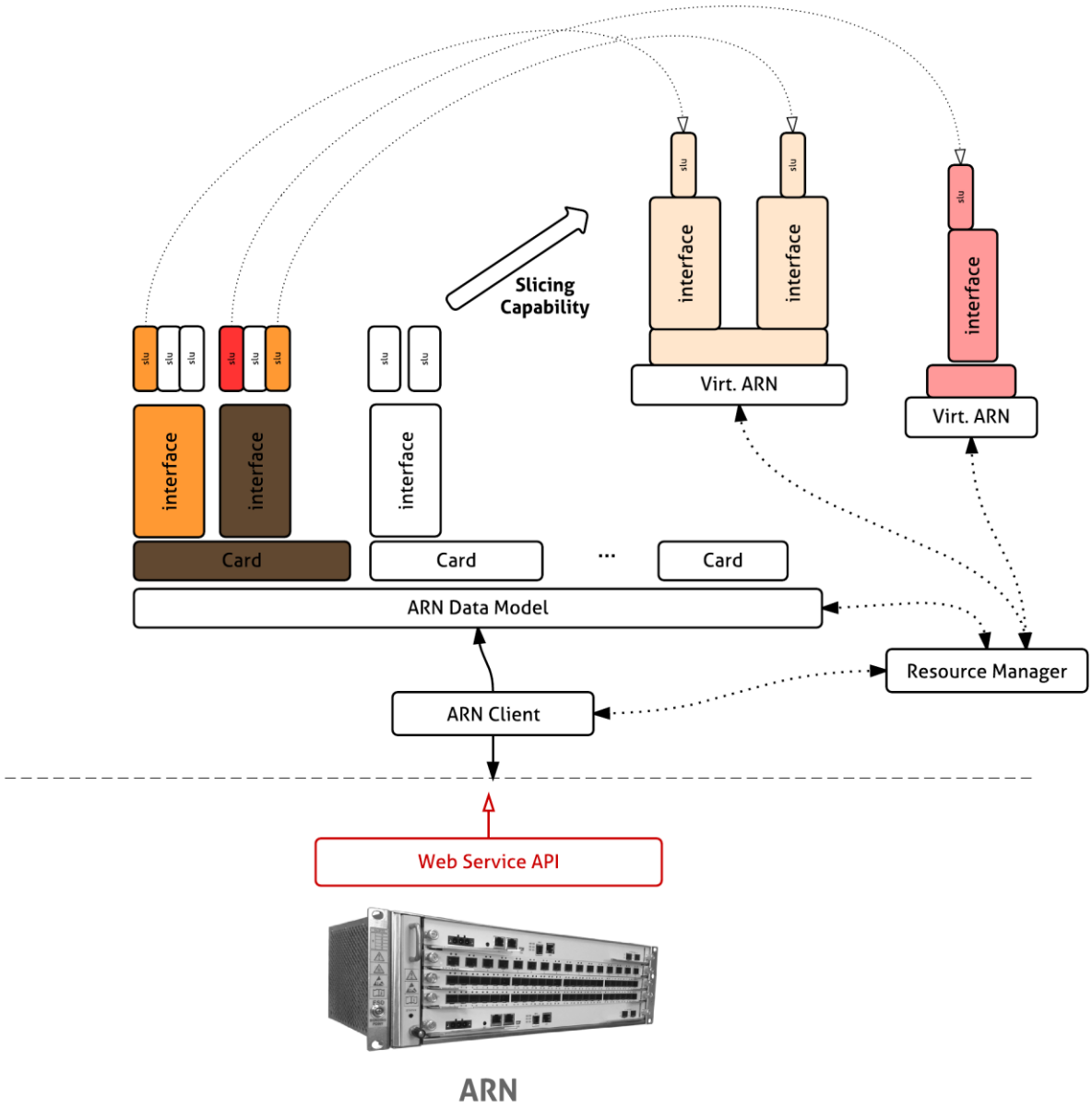

Figure 9: SODALES reservation capability internals

The logic is implemented through ranges of slicing units. A complete, fragmented definition of a slice therefore always is represented by an array of ranges for each slicing unit (i.e. Map<SlicingUnit, Range[]>), where each entry in the Map represents the information of one slicing dimension.

The information on slice units for each specific resource in the Open NaaS system is provided within the resource descriptor (concept extended from SODALES deliverable D3.3 [SODALES-D3.3]). There is only one type of slice defined, the multi-dimensional slice. A slice in the framework

is thus defined as a list of *SliceUnits*, which is added to the specification of the root resource descriptor as an attribute.

A *SliceUnit* represents one dimension of the multi-dimensional space offered by the slice. For example, in the ARN we can consider three different dimensions for the slicing units, Cards, Interfaces, and VLANs. For the moment in the management plane implementation there are only discrete slice units.

```
public class SliceUnit {

	public enum SliceType {
		DISCRETE
	}

	private String name;

	private SliceType type;

	public SliceUnit(String name) {
		this.name = name;
		this.type = SliceType.DISCRETE;
	}

	public String getName() {
		return name;
	}

	public SliceType getType() {
		return type;
	}
}
```

The slice units of one object (resource) can be accessed via the specification

```
private List<SliceUnit> sliceUnits;

...

List<SliceUnit> getSliceUnits() { return sliceUnits; }
```

The open access enabler bundle thus implements the definition of the slicing mechanisms that permits to create new slices for the different service providers by means of using the definition of slice unit. There are several Java classes implemented in order to provide the mechanisms and materialise the concept. The *Slice* class manages the original state to be able to ensure that no

slicing units added were part of the original state or were already used within another slice. A software copy of the original state is used to manage the current state of the slice, *cuts* and *adds* methods are performed over this state in order to manage the different slicing units moved from the original slice (physical substrate) towards the different virtual slices to be used by the different service providers.

```
public class Slice {

	private SliceUnit[] units;
	private Object data;
	public Slice(SliceUnit[] units, int[] sizes) {
		this.units = units;
		data = Array.newInstance(boolean.class, sizes);
	}

	private void set(int[] lbs, int[] ubs, boolean v) {
		int l = lbs.length;
		int[] c = Arrays.copyOfRange(lbs, 0, l);
		do {
			set(c, v);

			int i = 0;
			c[i]++;
			while ( i < l - 1 &&  c[i] > ubs[i] ) {
				c[i] = lbs[i];
				i++;
				c[i]++;
			}
		} while (c[l-1] <= ubs[l-1]);
	}

	public SliceUnit[] getUnits() {
		return units;
	}

	/**
	 * Is the given other slice contained in this slice, e.g. can be cut it
out
	 * of this slice.
	 */
	boolean contains(Slice other) {
		return false;
	}

	/**
	 * Method cutting the given slice from this slice, e.g. adapting the
```

```
	 * internal ranges to not any longer contain the ranges of the given
slice.
	 */
	Slice cut(Slice slice) {
		return slice;
	}

	/**
	 * Method adding the ranges of the given slice to this slice.
	 */
	void add(Slice slice) {
	}

	public boolean get(int[] coords) {
		switch (units.length) {
		case 1:
			boolean d1[] = (boolean[]) data;
			return d1[coords[0]];
		case 2:
			boolean d2[][] = (boolean[][]) data;
			return d2[coords[0]][coords[1]];
		case 3:
			boolean d3[][][] = (boolean[][][]) data;
			return d3[coords[0]][coords[1]][coords[2]];
		default:
			throw new RuntimeException(
					"Only up to three dimensions implemented");
		}
	}

	void set(SliceCube... cubes) {

		int[] lowerBounds = new int[units.length];
		int[] upperBounds = new int[units.length];
		for (SliceCube cube : cubes) {
			Range[] ranges = cube.getRanges();
			for (int i = 0; i < units.length; i++ ) {
				lowerBounds[i] = ranges[i].getLowerBound();
				upperBounds[i] = ranges[i].getUpperBound();
			}

			set(lowerBounds, upperBounds, true);
		}
	}
```

```
	public void set(int[] coords, boolean value) {
		switch (units.length) {
		case 1:
			boolean d1[] = (boolean[]) data;
			d1[coords[0]] = value;
			break;
		case 2:
			boolean d2[][] = (boolean[][]) data;
			d2[coords[0]][coords[1]] = value;
			break;
		case 3:
			boolean d3[][][] = (boolean[][][]) data;
			d3[coords[0]][coords[1]][coords[2]] = value;
			break;
		default:
			throw new RuntimeException(
					"Only up to three dimensions implemented");
		}
	}

	/**
	 * Is the given other slice contained in this slice, e.g. can be cut it
out
	 * of this slice.
	 */
	boolean contains(Slice other) {
		...
	}

	/**
	 * Method cutting the given slice from this slice, e.g. adapting the
	 * internal ranges to not any longer contain the ranges of the given
slice.
	 */
	boolean cut(Slice slice) {
		...
	}

	/**
	 * Method adding the ranges of the given slice to this slice.
	 */
	void add(Slice slice) {
		...
	}
```

```
}
```

The *Range* and *SliceCube* classes' specification is provided within Appendix B for further details. Complete implementations can be found in the software repository.

4 Software Validation

This section contains the validation performed over the developed test-bed. This validation comprehends the initial validation process and the preliminary results for the software itself. However, the complete validation of the SODALES software layer will be performed during the last year of execution of the project, by means of the pilot deployment and field-trial validation that will happen within WP5. It is expected that some updates to the software management layer will be performed during the third year of the project, in order to adjust the software to real deployments.

4.1 Functional requirements

The following table describes how the initial requirements described in D3.1 [SODALES-D3.1], and functionally analysed in D3.3 [SODALES-D3.3] are covered by the different software components.

id	Title	Status	Justification
CMP-REQ-001	Resource abstraction mechanisms and associated data models	Covered	The ARN and the CPE clients (and the drivers included within the software) cover the abstraction and data models requirements
CMP-REQ-002	On-demand slice-provisioning capabilities	Covered	Through the reservation capability interface, and the specific implementation for the SODALES access network the requirement is completely covered.
CMP-REQ-003	Resource discovery	Partially covered	Resource discovery has been only manually enabled. Thus, in order to add a physical resource the graphical user interface enables the option for the infrastructure provider, who can include the physical resource into the SODALES system by means of providing access credentials and the management URL of the device.
CMP-REQ-004	Security management	Partially covered	SODALES re-utilizes the authentication and authorization concepts of the OpenNaaS system. During the validation process in the field-trial it will be validated whether these are enough or there is the need to extend them with some specific SODALES features.
CMP-REQ-005	Operation of the heterogeneous physical resources	Covered	The operation of the heterogeneous physical resources involved in the SODALES ecosystem (ARN and CPE) is enabled by means of both the REST APIs attached at the virtual resources for operation purposes and utilized from the web-based user interface and the console access granted also through the user interface.
CMP-REQ-006	Resource Monitoring	Partially covered	Resource monitoring is enabled at the SODALES software layer. However it is not completely enabled at the web-enabled user interface, where not all the information can be monitored directly in the graphical

			components, only be menas of the local administration consoles of the devices.
CMP-REQ-007	Remote access to offered functionalities	Covered	Equally to CMP-REQ-005, SODALES provides remote access to the functionalities offered by the devices through the corresponding REST operation web services. In case the functionalities are not supported by the operation interface, the SODALES management user interface enables remote access to the terminal console of the devices.
CMP-REQ-011	Resource Failure Management	Partially covered	Resource failure management has only been covered in terms of identification of alarms and faliures. However, there is no internal application that automatically deals with failure management.

Table 1: Requirement and Software analysis

4.2 Tests

The software tests performed at this stage of the development have been focused in validating individual features of the SODALES software layer. For each one of the capabilities and features developed there is a set of JUnit tests implemented. Each one of these JUnit tests are focused on testing specific characteristics of the implemented modules.

The testing process has been controlled by means of the bamboo platform. All the tests performed can be found in the software repository under *test* folder, following the standard Maven structure for Java projects.

As an example, we include in this section some of the tests performed by the ARN and the CPE clients. The following code is the tests utilized to retrieve information from the real ARN, located at PTIN facilities, and accessed through a virtual private network.

```
package net.i2cat.dana.ptin_olt.client.api;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringWriter;
import java.net.URISyntaxException;
import java.nio.charset.Charset;

import javax.ws.rs.WebApplicationException;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
```

```
import net.i2cat.dana.ptin_olt.client.api.model.Request;
import net.i2cat.dana.ptin_olt.client.api.model.Response;

import org.apache.commons.io.IOUtils;
import org.apache.cxf.jaxrs.client.JAXRSClientFactory;
import org.junit.Assert;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
public class ClientTestWithRealServer {

	private static final String 			TESTSERVER_URL 		= 
"http://10.112.41.67/cgi-bin/xml-parser.cgi";

	private static final String 			REQUESTS_DIRECTORY 	= 
"/requests";

	private static PTIN_OLT_RPCClientAPI 	client;

	@BeforeClass
	public static void initClient() {
		client = JAXRSClientFactory.create(TESTSERVER_URL, 
PTIN_OLT_RPCClientAPI.class);
	}

	/**
	 * Sends each request sample to TESTSERVER_URL and retrieves the 
response.
	 *
	 * This test connects to a server at TESTSERVER_URL, which is in a 
private network. The test is not run, because we can not guarantee that any - *
	 * system running tests will have connectivity to TESTSERVER_URL. Please, 
comment Ignore annotation to run this test in your system, once you have
	 * - * required connectivity.
	 *
	 * @throws JAXBException
	 * @throws IOException
	 * @throws URISyntaxException
	 */
	@Test
```

```
	@Ignore
	public void sendRequestsTest()
			throws JAXBException, IOException, URISyntaxException {

		java.net.URL folderUrl =
this.getClass().getResource(REQUESTS_DIRECTORY);
		File folder = new File(folderUrl.toURI());
		StringBuffer report = new StringBuffer();
		report.append("Proccessed samples: " +
System.getProperty("line.separator"));
		try {
			for (File fileEntry : folder.listFiles()) {
				if (!fileEntry.isDirectory()) {
					if (fileEntry.getName().endsWith(".xml")) {
						InputStream in = new
FileInputStream(fileEntry);
						String xmlRequest = IOUtils.toString(in);
						Request request = loadRequestFromXML(new
ByteArrayInputStream(xmlRequest.getBytes(Charset.forName("UTF-8"))));

						Response response = null;
						try {
							response =
client.sendRequest(request);
						} catch (WebApplicationException ex) {
							// following response will only be
available
							// when the exception has been
produced after receiving the response (i.e. while parsing inside the client
proxy)
							javax.ws.rs.core.Response r =
ex.getResponse();
							System.out.println(r);
							// FIXME probably better to throw a
protocol exception or something similar
							throw ex;
						}
						System.out.println(toXml(response));
						Assert.assertNotNull(response);
						report.append(fileEntry.getName() +
System.getProperty("line.separator"));
					}
				}
			}
		} finally {
			System.out.println(report.toString());
		}
```

```
	}

	private Request loadRequestFromXML(InputStream in) throws JAXBException {
		return fromXml(in, Request.class);
	}

	/**
	 * Deserialize the XML InputStream into an instance of provided class
	 *
	 * @param xml
	 * @param objectClass
	 * @return
	 * @throws JAXBException
	 */
	public static <T> T fromXml(InputStream xml, Class<T> objectClass) throws
JAXBException {
		JAXBContext context = JAXBContext.newInstance(objectClass);
		Object obj = context
				.createUnmarshaller().unmarshal(xml);
		return (T) obj;
	}

	public static String toXml(Object obj) throws JAXBException {
		StringWriter sw = new StringWriter();
		JAXBContext context = JAXBContext.newInstance(obj.getClass());
		Marshaller m = context.createMarshaller();

		m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
		m.marshal(obj, sw);
		return sw.toString();
	}
}
```

The last Appendix (i.e. Appendix F) contains a test was designed to send requests and responses by means of using an emulator of the device, so the XML requests and responses were stored locally in order to avoid sending commands to the real device. This has enabled the development team to proceed with the developments in some occasions.

The following test is an example on how the data model of the Enet-38xx is serialized in order to create the corresponding software-objects once the information from the REST API in the equipment has been retrieved.

```
package net.i2cat.enet38xx.client.model;

import static org.junit.Assert.*;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import javax.xml.bind.JAXBException;

import org.apache.commons.io.IOUtils;
import org.custommonkey.xmlunit.XMLAssert;
import org.junit.Test;
import org.xml.sax.SAXException;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
public class ModelSerializationTest {

	private static final String meaIngressPortInfoSample =
	"/modelserialization/meaIngressPortInfo.xml";
	private static final String meaEgressPortInfoSample =
	"/modelserialization/meaEgressPortInfo.xml"; // missing
	private static final String meaPmCounterSample =
	"/modelserialization/meaPmCounter.xml";
	private static final String meaPolicerAllProfilesSample =
	"/modelserialization/meaPolicerAllProfiles.xml";
	private static final String meaPolicerProfileSample =
	"/modelserialization/meaPolicerProfile.xml";
	private static final String meaPortMappingSample =
	"/modelserialization/meaPortMapping.xml";
	private static final String meaGetServiceVlanIdSample =
	"/modelserialization/serviceVlanDetail.xml";

	@Test
	public void meaIngressPortInfoSerializationTest() throws JAXBException,
SAXException, IOException {

		// sample data, according with the content of sample xml
		int unit = 0;
		int portId = 111;
		int rx_enable = 0;
		int crc_check = 1;

		MeaIngressPortInfo src = new MeaIngressPortInfo();

		MeaIngressPortInfo.IngressData data = src.new IngressData();
```

```
            src.setIngress_Data(data);

            data.setUnit(unit);
            data.setPort_id(portId);
            data.setRx_enable(rx_enable);
            data.setCrc_check(crc_check);

            String xml = SerializationUtils.toXml(src);

            String originalXml =
IOUtils.toString(this.getClass().getResourceAsStream(meaIngressPortInfoSample))
;
            XMLAssert.assertXMLEqual(originalXml, xml);

            XMLAssert.assertXMLEqual(originalXml,
SerializationUtils.toXml(SerializationUtils.fromXml(originalXml,
MeaIngressPortInfo.class)));
      }

      @Test
      public void meaEgressPortInfoSerializationTest() throws JAXBException,
IOException, SAXException {

            MeaEgressPortInfo src = new MeaEgressPortInfo();

            fail("Not yet implemented");

            String xml = SerializationUtils.toXml(src);

            String originalXml =
IOUtils.toString(this.getClass().getResourceAsStream(meaEgressPortInfoSample));
            XMLAssert.assertXMLEqual(originalXml, xml);

            XMLAssert.assertXMLEqual(originalXml,
SerializationUtils.toXml(SerializationUtils.fromXml(originalXml,
MeaEgressPortInfo.class)));
      }

      @Test
      public void meaPmCounterSerializationTest() throws JAXBException,
IOException, SAXException {

            MeaPmCounter src = new MeaPmCounter();

            MeaPmCounter.PmCounter data = src.new PmCounter();
            src.setPmCounter(data);
```

```
            // sample data, according with the content of sample xml
            data.setDisGreen_bytes(0);
            data.setDisGreen_pkts(0);
            data.setDisMtu(0);
            data.setDisOther(0);
            data.setDisRed_pkts(0);
            data.setDisYellow_pkts(0);
            data.setDisYellowRed_bytes(0);
            data.setFwdGreen_bytes(4862958000L);
            data.setFwdGreen_pkts(4862959L);
            data.setFwdGreenRate(29920000L);
            data.setFwdYellow_bytes(0);
            data.setFwdYellow_pkts(0);
            data.setFwdYellowRate(0);
            data.setPmId(10);

            String xml = SerializationUtils.toXml(src);

            String originalXml =
IOUtils.toString(this.getClass().getResourceAsStream(meaPmCounterSample));
            XMLAssert.assertXMLEqual(originalXml, xml);

            XMLAssert.assertXMLEqual(originalXml,
SerializationUtils.toXml(SerializationUtils.fromXml(originalXml,
MeaPmCounter.class)));
      }

      @Test
      public void meaPolicerAllProfilesSerializationTest() throws
JAXBException, SAXException, IOException {

            MeaPolicerAllProfiles src = new MeaPolicerAllProfiles();

            // sample data, according with the content of sample xml
            List<MeaPolicerAllProfiles.PolicerProfId> ids = new
ArrayList<MeaPolicerAllProfiles.PolicerProfId>();
            for (int i=0; i<5; i++) {
                  MeaPolicerAllProfiles.PolicerProfId id = src.new
PolicerProfId();
                  id.setUnit(0);
                  id.setProfileId(i+1);
                  ids.add(id);
            }
            src.setPolicerProfId(ids);

            String xml = SerializationUtils.toXml(src);
```

```
            String originalXml =
IOUtils.toString(this.getClass().getResourceAsStream(meaPolicerAllProfilesSampl
e));
            XMLAssert.assertXMLEqual(originalXml, xml);

            XMLAssert.assertXMLEqual(originalXml,
SerializationUtils.toXml(SerializationUtils.fromXml(originalXml,
MeaPolicerAllProfiles.class)));
      }

      @Test
      public void meaPolicerProfileSerializationTest() throws JAXBException,
IOException, SAXException {

            MeaPolicerProfile src = new MeaPolicerProfile();

            // sample data, according with the content of sample xml
            MeaPolicerProfile.PolicerData data = src.new PolicerData();
            src.setPolicerData(data);

            data.setCbs(64000L);
            data.setCir(100000000L);
            data.setColor(0);
            data.setComp(0);
            data.setEbs(0);
            data.setEir(0);
            data.setGn_sign(0);
            data.setGn_type(5);
            data.setType(0);

            String xml = SerializationUtils.toXml(src);

            String originalXml =
IOUtils.toString(this.getClass().getResourceAsStream(meaPolicerProfileSample));
            XMLAssert.assertXMLEqual(originalXml, xml);

            XMLAssert.assertXMLEqual(originalXml,
SerializationUtils.toXml(SerializationUtils.fromXml(originalXml,
MeaPolicerProfile.class)));
      }

      @Test
      public void meaPortMappingSerializationTest() throws JAXBException,
IOException, SAXException {

            MeaPortMapping src = new MeaPortMapping();
```

```
            // sample data, according with the content of sample xml
            List<String> ports = Arrays.asList("1", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "11", "12", "13", "14", "15", "100", "101", "102", "103",
"104", "105", "108", "109", "110", "111");
            List<MeaPortMapping.PortMapping> portMappings = new
ArrayList<MeaPortMapping.PortMapping>(ports.size());
            for (String port : ports) {
                  MeaPortMapping.PortMapping portMapping = src.new
PortMapping();
                  portMapping.setUnit(0);
                  portMapping.setPort(Integer.parseInt(port));
                  portMappings.add(portMapping);
            }

            String xml = SerializationUtils.toXml(src);

            String originalXml =
IOUtils.toString(this.getClass().getResourceAsStream(meaPortMappingSample));
            XMLAssert.assertXMLEqual(originalXml, xml);

            XMLAssert.assertXMLEqual(originalXml,
SerializationUtils.toXml(SerializationUtils.fromXml(originalXml,
MeaPortMapping.class)));
      }

      @Test
      public void meaGetServiceVlanIdSerializationTest() throws JAXBException,
IOException, SAXException {

            MeaGetServiceVlanId src = new MeaGetServiceVlanId();

            // sample data, according with the content of sample xml
            MeaGetServiceVlanId.ServiceDataBase data = src.new
ServiceDataBase();
            src.setServiceDataBase(data);

            MeaGetServiceVlanId.ServiceDataBase.ServiceOutputPort
serviceOutputPort = data.new ServiceOutputPort();
            data.setServiceOutputPort(serviceOutputPort);

            MeaGetServiceVlanId.ServiceDataBase.ServiceOutputPort.Cluster
cluster = serviceOutputPort.new Cluster();
            serviceOutputPort.setCluster(cluster);

            cluster.setClusterId(104);
```

```
            data.seteIngressType(1);
            data.setInner_vlanId(0);
            data.setOuter_vlanId(10);
            data.setPmId(3);
            data.setPolicerId(2);
            data.setServiceId(6);
            data.setServiceOutputPort(serviceOutputPort);
            data.setSrc_port(105);
            data.setUnit(0);
            data.setVlanEdit_flowtype(2);
            data.setVlanEdit_inner_command(3);
            data.setVlanEdit_inner_vlan(20);
            data.setVlanEdit_outer_command(0);
            data.setVlanEdit_outer_vlan(0);

            String xml = SerializationUtils.toXml(src);

            String originalXml =
IOUtils.toString(this.getClass().getResourceAsStream(meaGetServiceVlanIdSample)
);
            XMLAssert.assertXMLEqual(originalXml, xml);

            XMLAssert.assertXMLEqual(originalXml,
SerializationUtils.toXml(SerializationUtils.fromXml(originalXml,
MeaGetServiceVlanId.class)));
      }


}
```

5 Software release and deployment

All the software developed, except the Web Services APIs for the corresponding hardware devices, which is part of the commercial devices itself, is located in the corresponding repository

```
http://stash.i2cat.net/
```

To access and install the code there is the need to use the aforementioned user and password

```
User: EC_reviewer
Password: 2ndreview
```

You need to run the following commands to download the code. First, download and install the Open NaaS platform from source code

```
$ git clone https://github.com/dana-i2cat/mqnaas/
$ cd mqnaas
$ mvn clean install -DskipTests
```

Download and install the reservation capability feature from source code

```
$ git clone http://username@git.i2cat.net/scm/SODALES/reservation-
capability.git
$ cd reservation-capability
$ mvn clean install -DskipTests
```

Download and install the sodales clients features

```
$ git clone http://username@git.i2cat.net/scm/SODALES/enet-38xx
$ cd enet-38xx
$ mvn clean install -DskipTests
```

```
$ git clone http://username@git.i2cat.net/scm/SODALES/mqnaas-olt
$ cd mqnaas-olt
$ mvn clean install -DskipTests
```

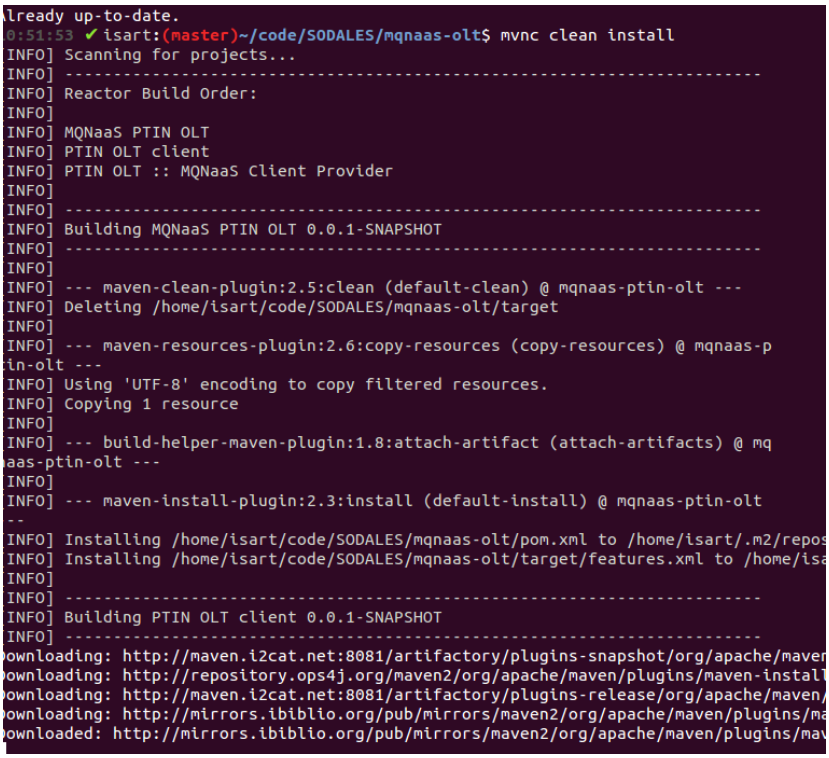

Figure 10: Compile and install mqnaas-ptin-olt

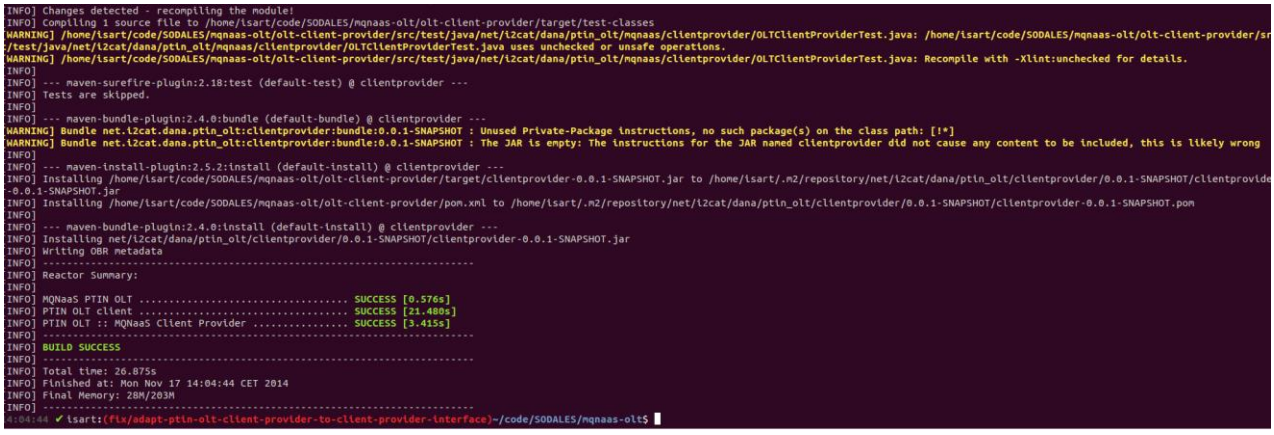

Figure 11: Compile and install mqnaas-ptin-olt

Run MQNaaS

MQNaaS runs inside an OSGI container called Apache Karaf. In order to run MQNaaS features, you will need to download and run Apache Karaf 3.0.1 first.

Download Apache Karaf 3.0.1 Binary Distribution for your specific OS from the official website

```
http://karaf.apache.org/index/community/download/archives.html#Karaf3.0.1.
```

Extract Karaf

```
Windows version
$ unzip apache-karaf-3.0.1.zip
Unix version
$ tar -xvf apache-karaf-3.0.1.tar.gz
```

Run Karaf

```
Windows version
$ Execute apache-karaf/3.0.1/bin/karaf.bat
Unix version
$ ./apache-karaf-3.0.1/bin/karaf
```

Install MQNaaS bundles in karaf

```
karaf@root()> feature:repo-add mvn:org.mqnaas/mqnaas/0.0.1-
SNAPSHOT/xml/features
karaf@root()> feature:install mqnaas
```

Install MQNaaS reservation capability

```
karaf@root()> feature:repo-add mvn:net.i2cat.dana.mqnaas/reservation-
capability/0.0.1-SNAPSHOT/xml/features
karaf@root()> feature:install reservation-capability
```

Install mqnaas-olt

```
karaf@root()> feature:repo-add mvn:net.i2cat.dana.mqnaas-olt/mqnaas-ptin-
olt/0.0.1-SNAPSHOT/xml/features
karaf@root()> feature:install mqnaas-ptin-olt
```

Install enet-38xx

```
karaf@root()> feature:repo-add mvn:net.i2cat.dana.enet-38xx/enet-38xx/0.0.1-
SNAPSHOT/xml/features
karaf@root()> feature:install enet-38xx
```

In any case, we have integrated everygthing and created an appliance ready to be deployed. It is released as a Virtual Machine (VM) with all the software bundles installed. The VM is based on an Ubuntu 14.04 distribution (64bits). It can be downloaded from the following link:

```
ftp://84.88.40.143/home/efetepe/sodales-d3-prototype.ova
```

The user and password required for downloading the VM are the following

```
User: efetepe
Password: 3f3t3p314
```

In order to use the software it is required to download and execute the VM, for example using Virtual Box. Once installed, the login credentials for the VM follow

```
User: sodales
Password: reverse
```

Once logged into the VM, there is one automated script used in order to execute the SODALES software layer. The script can be found in the following folder:

```
~/software/scripts/
```

6 Conclusions and lessons learnt

This deliverable comprises the prototype for the implemented SODALES software layer, responsible for the control and management of the physical devices populating the SODALES environment. The report is the documentation for the prototype deliverable.

The SODALES software has been focused mainly in covering the requirements identified during the initial stages of the work package execution, and reported in the corresponding deliverable D3.1 [SODALES-D3.1]. The implementation work has been mainly focused on covering the two following aspects:

- To provide an open access network model over the physical infrastructure considered, enabling the infrastructure provider to create independent slices of the infrastructure.
- To provide an unified platform for management and operation of the different devices, both virtual and physical, for the infrastructure providers and/or the service providers, depending on the type of resource addressed.

The report contains the most important software modules built for both the Active Remote Node, which is being built by PTIN, and the Customer Premises Equipment, which is being built by Ethernity Networks; as well as the most important features designed and implemented in order to provide a generic open access enabler mechanism. It is worth to remark that through the reservation capability implemented in the OpenNaaS system any technology can be utilized to provide open access network capabilities, by means of the slicing units concept.

In terms of validation, the report contains very brief descriptions of some of the JUnit tests performed within the lab in order to validate the specific features developed. However, there is still place to improvents within the software. Real validation will be performed within work package five by means of the field trials to be performed and deployed in Portugal. It is expected that software bugs are reported, and that the development team focuses its efforts in solving these reported bugs. Work package three has officially ended, however, development effort will be carried within work package five out in order to address the different problems (if any) identified within the real field trial deployment.

Last but not least, it is worth to remark that the software development process within the SODALES project has encountered several problems that may be solved for future projects focusing on software development. First, having distributed development teams is not an easy-to-deal situation when aiming at applying any product-oriented development methodology, since communication typically does not happen as expected. Secondly, developing a software layer for hardware that it is under development, and which flexible architecture, as demonstrated in several WP2 devlierables, makes that features are not clearly stabilised, is not again an easy-to-deal situation for software development teams.

7 Acronyms

ARN	Active Remote Node
API	Application Programming Interface
CPE	Customer Premises Equipment
FFP	Fabric Flow Processor
JAXB	Java Architecture for XML Binding
JVM	Java Virtual Machine
NMS	Network Management System
ODL	OpenDaylight
OSGi	Open Services Gateway Alliance
REST	Representational State Transfer
SCN	Signaling Communication Network
SLU	Slicing Managmenet Unit
SDN	Software Defined Networking
SODALES	Software-defined Access using Low-Energy Sub-systems
VM	Virtual Machine
VTN	Virtual Tenant Network
XML	Extensible Markup Language
XSD	XML Schema Data

8 References

[SODALES-D3.3] Ferrer Riera, J., Bock C., et al. SODALES Deliverable D3.3: Control and Management Plane Design

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

[Scrum] http://www.scrum.org

[Jira] http://www.atlassian.com/software/jira

[Godzilla] What does JIRA mean? Atlassian Forums. Online: https://confluence.atlassian.com/pages/viewpage.action?pageId=22321995

[Git] http://git-scm.com

[Bamboo] http://www.atlassian.com/software/bamboo

[MANTYCHORE-FP7] http://mantychore.eu

[REST] Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures. PhD Dissertation. University of California. Online: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[SODALES-D2.1] Parker, M., et al: SODALES D2.1: ARN Design and Interfacing

[SODALES-D2.4] Jungnickel, V. et al: SODALES D2.4: ARN Interface for LTE and beyond mobile services with support for legacy services

[Chowdury] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. 2009. Network virtualization: state of the art and research challenges. *Comm. Mag.* 47, 7 (July 2009), 20-26. DOI=10.1109/MCOM.2009.5183468 http://dx.doi.org/10.1109/MCOM.2009.5183468

[Botero] Fischer, A.; Botero, J.F.; Till Beck, M.; de Meer, H.; Hesselbach, X., "Virtual Network Embedding: A Survey," *Communications Surveys & Tutorials, IEEE* , vol.15, no.4, pp.1888,1906, Fourth Quarter 2013 doi: 10.1109/SURV.2013.013013.00155

[ODL] http://www.opendaylight

[OPENDOVE] https://wiki.opendaylight.org/view/Open_DOVE:Main

[CONTENT-FP7] http://content-fp7.eu

9 Appendix A – Complete Reservation specification

Reservation Model

```
package net.i2cat.dana.mqnaas.capability.reservation.model;

import java.util.Set;

import org.mqnaas.core.api.IResource;
import org.mqnaas.core.api.IRootResource;
import org.mqnaas.core.api.Identifiable;

/**
 * <p>
 * Class representing the main model of the reservation capability.
 * </p>
 * <p>
 * A set of {@link Device devices} could be reserved during a period for time,
established by the {@link #getStartDate() start} and
 * {@link #getEndDate() final} dates. This reservation would have a {@link
ReservationState state}, depending on the reservation period and the user
 * interaction. During the {@link ReservationState#RESERVED} state, the reservation is
mapped and transfered into MqNaaS {@link IResource}.
 * </p>
 *
 * @author Adrián Roselló Rey (i2CAT)
 *
 */
public class Reservation implements Identifiable {

	ReservationState	reservationState;
	Set<Device>			devices;
	Set<IRootResource>	resources;
	Period				period;
	String				id;

	public Reservation(String id) {
		this.id = id;
	}

	@Override
	public String getId() {
		return id;
	}
```

```
	public ReservationState getReservationState() {
		return reservationState;
	}

	public void setReservationState(ReservationState reservationState) {
		this.reservationState = reservationState;
	}

	public Set<Device> getDevices() {
		return devices;
	}

	public void setDevices(Set<Device> devices) {
		this.devices = devices;
	}

	public Set<IRootResource> getResources() {
		return resources;
	}

	public void setResources(Set<IRootResource> resources) {

		this.resources = resources;
	}

	public Period getPeriod() {
		return period;
	}

	public void setPeriod(Period period) {
		this.period = period;
	}

	@Override
	public String toString() {
		return "Reservation [reservationState=" + reservationState + ", devices="
+ devices + ", resources=" + resources + ", period=" + period + "]";
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
```

```
            result = prime * result + ((devices == null) ? 0 : devices.hashCode());
            result = prime * result + ((period == null) ? 0 : period.hashCode());
            result = prime * result + ((reservationState == null) ? 0 :
reservationState.hashCode());
            result = prime * result + ((resources == null) ? 0 :
resources.hashCode());
            return result;
      }

      @Override
      public boolean equals(Object obj) {
            if (this == obj)
                  return true;
            if (obj == null)
                  return false;
            if (getClass() != obj.getClass())
                  return false;
            Reservation other = (Reservation) obj;
            if (devices == null) {
                  if (other.devices != null)
                        return false;
            } else if (!devices.equals(other.devices))
                  return false;
            if (period == null) {
                  if (other.period != null)
                        return false;
            } else if (!period.equals(other.period))
                  return false;
            if (reservationState != other.reservationState)
                  return false;
            if (resources == null) {
                  if (other.resources != null)
                        return false;
            } else if (!resources.equals(other.resources))
                  return false;
            return true;
      }
}
```

Reservation Request

```
package net.i2cat.dana.mqnaas.capability.reservation.model;

import java.util.Date;
```

```
import java.util.Set;

/**
 * <p>
 * Class to request {@link Devices} reservations during a specific period of time.
 * </p>
 *
 * @author Adrián Roselló Rey (i2CAT)
 *
 */
public class ReservationRequest {

	Set<Device>  devices;
	Date         startDate;
	Date         endDate;

	public Set<Device> getDevices() {
		return devices;
	}

	public void setDevices(Set<Device> devices) {
		this.devices = devices;
	}

	public Date getStartDate() {
		return startDate;
	}

	public void setStartDate(Date startDate) {
		this.startDate = startDate;
	}

	public Date getEndDate() {
		return endDate;
	}

	public void setEndDate(Date endDate) {
		this.endDate = endDate;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
```

```
		result = prime * result + ((devices == null) ? 0 : devices.hashCode());
		result = prime * result + ((endDate == null) ? 0 : endDate.hashCode());
		result = prime * result + ((startDate == null) ? 0 :
startDate.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		ReservationRequest other = (ReservationRequest) obj;
		if (devices == null) {
			if (other.devices != null)
				return false;
		} else if (!devices.equals(other.devices))
			return false;
		if (endDate == null) {
			if (other.endDate != null)
				return false;
		} else if (!endDate.equals(other.endDate))
			return false;
		if (startDate == null) {
			if (other.startDate != null)
				return false;
		} else if (!startDate.equals(other.startDate))
			return false;
		return true;
	}

	@Override
	public String toString() {
		return "ReservationRequest [devices=" + devices + ", startDate=" +
startDate + ", endDate=" + endDate + "]";
	}
}
```

Period

```
package net.i2cat.dana.mqnaas.capability.reservation.model;
```

```
import java.util.Date;

/**
 * <p>
 * Class representing a period of time, established by a start and an end date.
 * </p>
 *
 * @author Adrián Roselló Rey (i2CAT)
 *
 */
public class Period implements Comparable<Period> {

	private Date startDate;
	private Date endDate;

	public Period(Date startDate, Date endDate) {

		if (startDate == null || endDate == null)
			throw new IllegalArgumentException("Start and end date should not
be null");
		if (endDate.before(startDate))
			throw new IllegalArgumentException("Start date should be before the
end date");

		this.startDate = startDate;
		this.endDate = endDate;

	}

	public Date getStartdate() {
		return startDate;
	}

	public Date getEndDate() {
		return endDate;
	}

	/**
	 * @return <ul>
	 *         <li>"-1", if the starting and end date are smaller than the starting
date of the <code>other</code> period.</li>
	 *         <li>"1", if the starting and end date are smaller than the end date
of the <code>other</code>period.</li>
```

```
         *          <li>"0", otherwise
         *          </ul>
         *
         */
        @Override
        public int compareTo(Period other) {

                // both are equals
                if (startDate.equals(other.getStartdate()) &&
endDate.equals(other.getEndDate()))
                        return 0;

                // Both have same endDate, but different startDate
                if (endDate.equals(other.getEndDate())) {
                        if (startDate.before(other.startDate))
                                return -1;
                        return 1;
                }

                // Both have same startDate, but different endDate
                if (startDate.equals(other.getEndDate())) {
                        if (endDate.before(other.getEndDate()))
                                return -1;
                        return 1;
                }

                // They don't overlap
                if (startDate.after(other.getEndDate()))
                        return 1;
                if (endDate.before(other.getStartdate()))
                        return -1;

                // They overlap
                if (startDate.before(other.getStartdate()) &&
endDate.before(other.getEndDate()))
                        return -1;
                return 1;
        }

        @Override
        public String toString() {
                return "Period [startDate=" + startDate + ", endDate=" + endDate + "]";
        }
```

```
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((endDate == null) ? 0 : endDate.hashCode());
		result = prime * result + ((startDate == null) ? 0 :
startDate.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Period other = (Period) obj;
		if (endDate == null) {
			if (other.endDate != null)
				return false;
		} else if (!endDate.equals(other.endDate))
			return false;
		if (startDate == null) {
			if (other.startDate != null)
				return false;
		} else if (!startDate.equals(other.startDate))
			return false;
		return true;
	}
}
```

10 Appendix B – Slicing Unit Utils

Range

A range is defined by two integer bounds, considering that both bounds are included in the range.

```
public class Range {
	private int lowerBound, upperBound;
}
```

SliceCube

A cube represents a multi-dimensional cube and is represented by an array of ranges

```
public class SliceCube {

	private Range[] ranges;

	SliceCube(Range[] cube) {
		this.ranges = cube;
	}

	public Range[] getRanges() {
		return ranges;
	}
}
```

11 Appendix C – Graphical User Interface Sketches

This appendix contains the design process of the web-enabled graphical user interface, which provides the single point-of-entry to the different features offered by the SODALES management plane. There are two main components in the design process, the simplified navigation maps for both the infrastructure and the service provider, which are similar with different actions and options; and the different sketches used to build the web views. Finally, we also include a set of preliminary web views that were created in the process.

The Appendix does not include all the sketches and draft web designs performed, it only contains a set of the most representative ones.

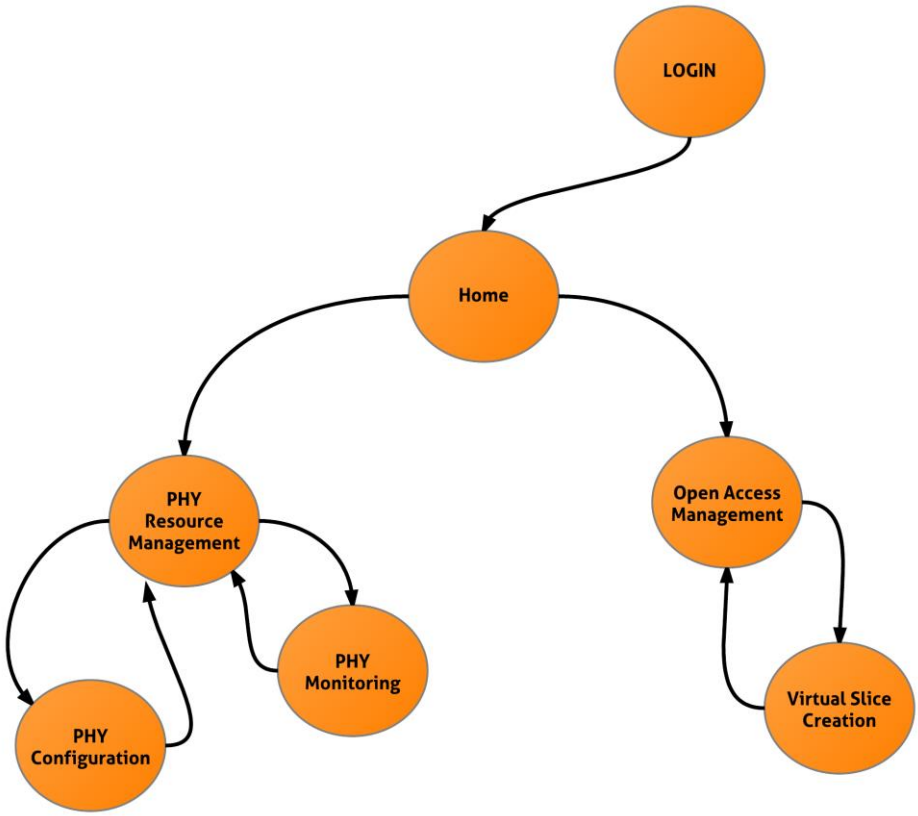

Figure 12: SODALES Infrastructure Provider Navigation Map

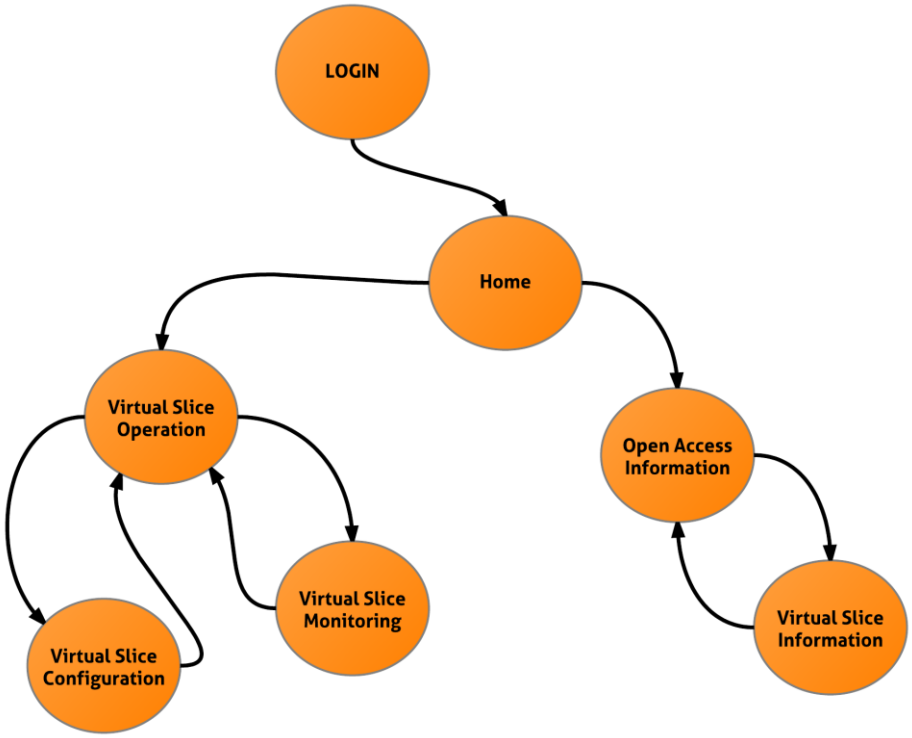

Figure 13: SODALES Service Provider Navigation Map

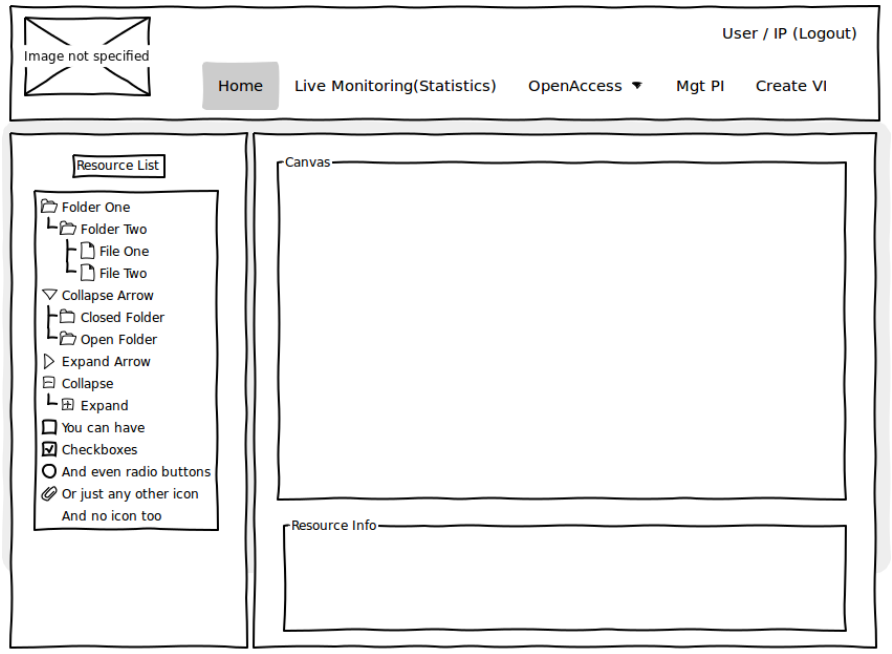

Figure 14: SODALES Home Page Sketch

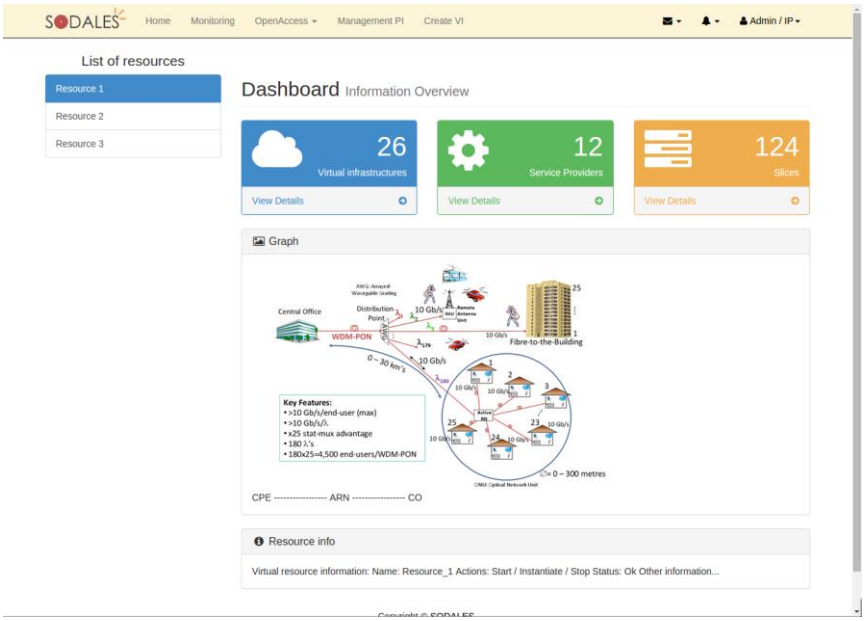

Figure 15: SODALES Home Page Draft Design

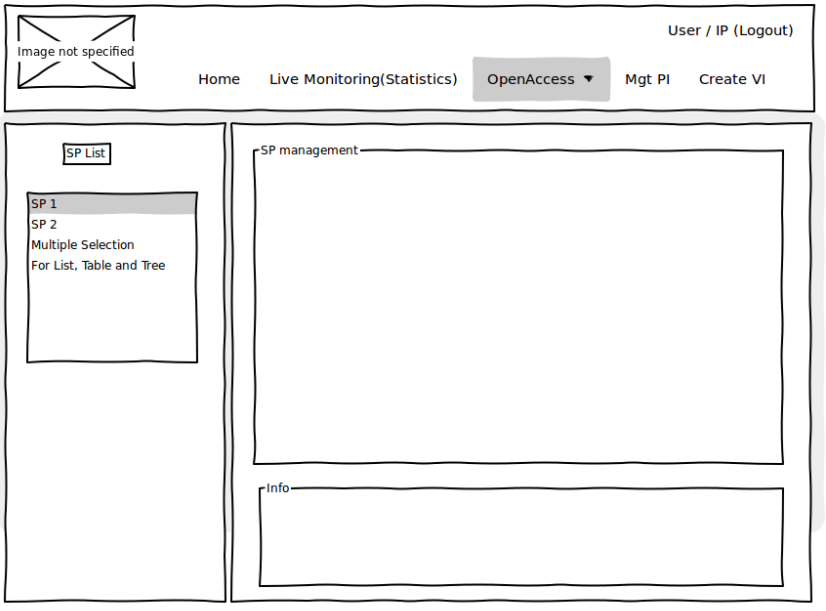

Figure 16: SODALES Open Access Page Sketch

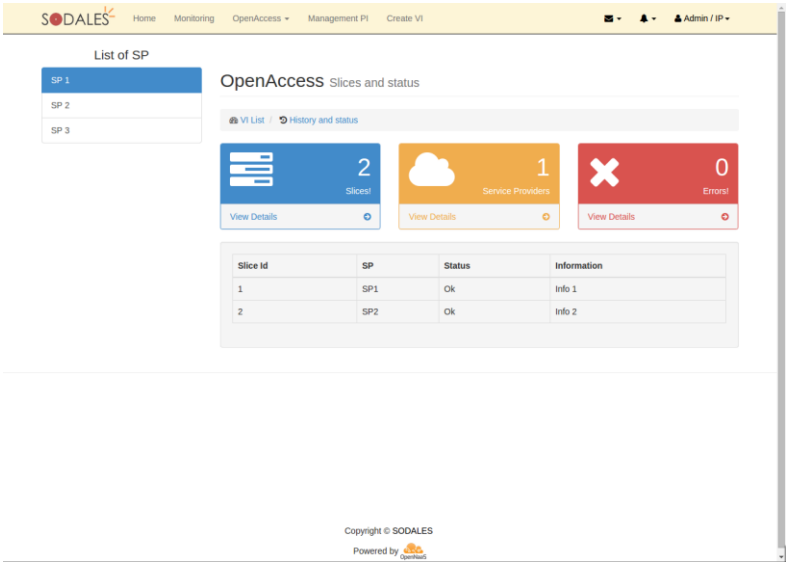

Figure 17: SODALES Open Access Page Draft Design

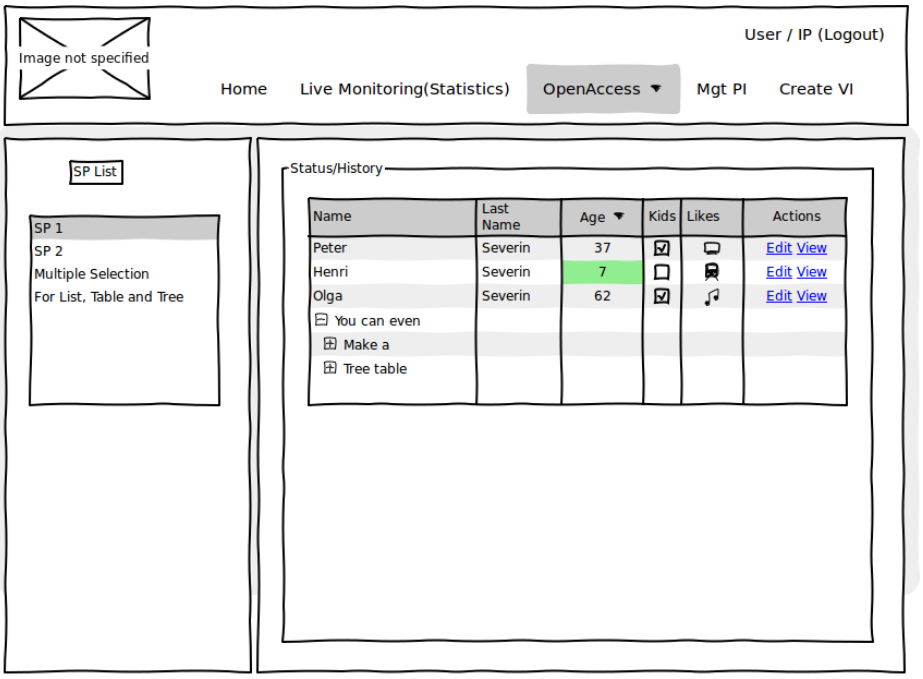

Figure 18: SODALES Monitoring Sketch

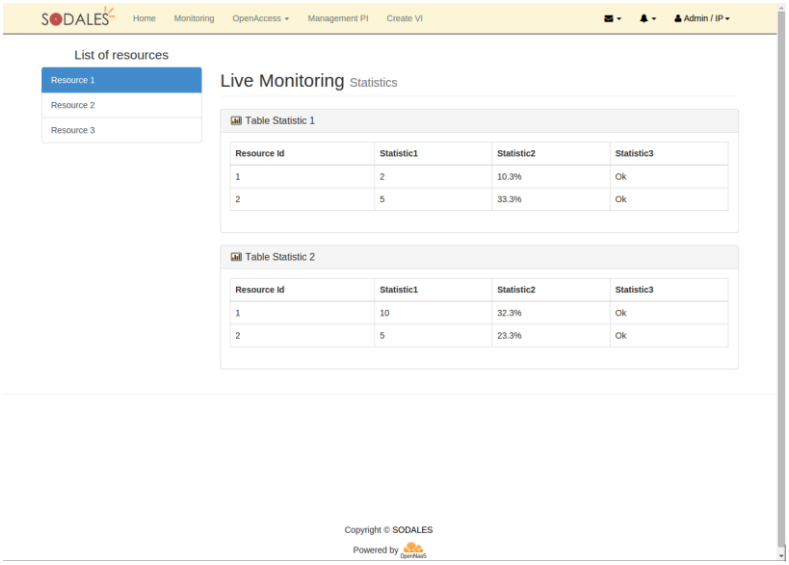

Figure 19: SODALES Monitoring Page draft design

12 Appendix D – CPE Implementation

12.1 XML Models

meaIngressPortInfo.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<meaIngressPort xmlns="http://www.dana.i2cat.net/mea-api/igressPort">
  <ingress_Data>
    <unit>0</unit>
    <port_id>111</port_id>
    <rx_enable>0</rx_enable>
    <crc_check>1</crc_check>
  </ingress_Data>
</meaIngressPort>
```

meaPolicerAllProfiles.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<meaPolicerAllProfiles xmlns="http://www.ethernity-net.com/enet/PolicerAllProfiles">
  <policerProfId>
    <unit>0</unit>
    <profileId>1</profileId>
  </policerProfId>
  <policerProfId>
    <unit>0</unit>
    <profileId>2</profileId>
  </policerProfId>
  <policerProfId>
    <unit>0</unit>
    <profileId>3</profileId>
  </policerProfId>
  <policerProfId>
    <unit>0</unit>
    <profileId>4</profileId>
  </policerProfId>
  <policerProfId>
    <unit>0</unit>
    <profileId>5</profileId>
  </policerProfId>
</meaPolicerAllProfiles>
```

meaPolicerProfile.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<meaPolicerProfile xmlns="http://www.ethernity-net.com/enet/PolicerProfile">
  <PolicerData>
    <Type>0</Type>
    <CIR>100000000</CIR>
    <CBS>64000</CBS>
    <EIR>0</EIR>
    <EBS>0</EBS>
    <gn_sign>0</gn_sign>
    <gn_type>5</gn_type>
    <comp>0</comp>
    <Color>0</Color>
  </PolicerData>
</meaPolicerProfile>
```

meaPortMapping.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<meaPortMapping xmlns="http://www.ethernity-net.com/enet/igressPortMapping">
  <portMapping>
    <unit>0</unit>
    <port>1</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>2</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>3</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>4</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>5</port>
  </portMapping>
  <portMapping>
```

```
    <unit>0</unit>
    <port>6</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>7</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>8</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>9</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>10</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>11</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>12</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>13</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>14</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>15</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>100</port>
  </portMapping>
  <portMapping>
```

```
    <unit>0</unit>
    <port>101</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>102</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>103</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>104</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>105</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>108</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>109</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>110</port>
  </portMapping>
  <portMapping>
    <unit>0</unit>
    <port>111</port>
  </portMapping>
</meaPortMapping>
```

12.2 Software Objects (Java)

meaEgressPortInfo.java

```
package net.i2cat.enet38xx.client.model;
```

```
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
<meaIngressPort xmlns="http://www.dana.i2cat.net/mea-api/igressPort">
  <ingress_Data>
    <unit>0</unit>
    <port_id>111</port_id>
    <rx_enable>0</rx_enable>
    <crc_check>1</crc_check>
  </ingress_Data>
</meaIngressPort>
 *
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@XmlRootElement(name="meaIngressPort", namespace="http://www.dana.i2cat.net/mea-
api/igressPort")
@XmlAccessorType(XmlAccessType.FIELD)
public class MeaIngressPortInfo {

	private IngressData ingress_Data;

	public IngressData getIngress_Data() {
		return ingress_Data;
	}

	public void setIngress_Data(IngressData ingress_Data) {
		this.ingress_Data = ingress_Data;
	}

	@XmlType(namespace="http://www.dana.i2cat.net/mea-api/igressPort",
propOrder={"unit", "port_id", "rx_enable", "crc_check", "My_mac"})
	@XmlAccessorType(XmlAccessType.FIELD)
	class IngressData {

		private int unit;
		private int port_id;
		private int rx_enable;
		private int crc_check;
```

```
		private String My_mac;

		public int getUnit() {
			return unit;
		}

		public void setUnit(int unit) {
			this.unit = unit;
		}

		public int getPort_id() {
			return port_id;
		}

		public void setPort_id(int port_id) {
			this.port_id = port_id;
		}

		public int getRx_enable() {
			return rx_enable;
		}

		public void setRx_enable(int rx_enable) {
			this.rx_enable = rx_enable;
		}

		public int getCrc_check() {
			return crc_check;
		}

		public void setCrc_check(int crc_check) {
			this.crc_check = crc_check;
		}

		public String getMy_mac() {
			return My_mac;
		}

		public void setMy_mac(String my_mac) {
			My_mac = my_mac;
		}
	}
}
```

meaIngressPortInfo.java

```
package net.i2cat.enet38xx.client.model;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
* @author Isart Canyameres Gimenez (i2cat)
 *
 */
@XmlRootElement(name="meaIngressPort", namespace="http://www.dana.i2cat.net/mea-
api/igressPort")
@XmlAccessorType(XmlAccessType.FIELD)
public class MeaIngressPortInfo {

	private IngressData ingress_Data;

	public IngressData getIngress_Data() {
		return ingress_Data;
	}

	public void setIngress_Data(IngressData ingress_Data) {
		this.ingress_Data = ingress_Data;
	}

	@XmlType(namespace="http://www.dana.i2cat.net/mea-api/igressPort",
propOrder={"unit", "port_id", "rx_enable", "crc_check", "My_mac"})
	@XmlAccessorType(XmlAccessType.FIELD)
	class IngressData {

		private int unit;
		private int port_id;
		private int rx_enable;
		private int crc_check;
		private String My_mac;

		public int getUnit() {
			return unit;
		}

		public void setUnit(int unit) {
			this.unit = unit;
```

```
		}

		public int getPort_id() {
			return port_id;
		}

		public void setPort_id(int port_id) {
			this.port_id = port_id;
		}

		public int getRx_enable() {
			return rx_enable;
		}

		public void setRx_enable(int rx_enable) {
			this.rx_enable = rx_enable;
		}

		public int getCrc_check() {
			return crc_check;
		}

		public void setCrc_check(int crc_check) {
			this.crc_check = crc_check;
		}

		public String getMy_mac() {
			return My_mac;
		}

		public void setMy_mac(String my_mac) {
			My_mac = my_mac;
		}
	}
}
```

meaPolicerAllProfiles.java

```
package net.i2cat.enet38xx.client.model;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
```

```
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@XmlRootElement(name="meaPolicerAllProfiles", namespace="http://www.ethernity-net.com/enet/PolicerAllProfiles")
@XmlAccessorType(XmlAccessType.FIELD)
public class MeaPolicerAllProfiles {

	private List<PolicerProfId> policerProfId;

	public List<PolicerProfId> getPolicerProfId() {
		return policerProfId;
	}

	public void setPolicerProfId(List<PolicerProfId> policerProfId) {
		this.policerProfId = policerProfId;
	}

	@XmlType(namespace="http://www.ethernity-net.com/enet/PolicerAllProfiles", propOrder={"unit", "profileId"})
	@XmlAccessorType(XmlAccessType.FIELD)
	class PolicerProfId {

		private int unit;
		private int profileId;

		public int getUnit() {
			return unit;
		}
		public void setUnit(int unit) {
			this.unit = unit;
		}
		/**
		 *
		 * @return identifier of a @link{MeaPolicerProfile}
		 */
		public int getProfileId() {
			return profileId;
```

```
		}
		public void setProfileId(int profileId) {
			this.profileId = profileId;
		}
	}

}
```

meaPolicerProfile.java

```
package net.i2cat.enet38xx.client.model;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@XmlRootElement(name="meaPolicerProfile", namespace="http://www.ethernity-
net.com/enet/PolicerProfile")
@XmlAccessorType(XmlAccessType.FIELD)
public class MeaPolicerProfile {

	@XmlElement(name="PolicerData")
	private PolicerData policerData;

	public PolicerData getPolicerData() {
		return policerData;
	}

	public void setPolicerData(PolicerData policerData) {
		this.policerData = policerData;
	}


	@XmlType(namespace="http://www.dana.i2cat.net/mea-api/egressPort",
propOrder={"type", "cir", "cbs", "eir", "ebs", "gn_sign", "gn_type", "comp", "color"})
	@XmlAccessorType(XmlAccessType.FIELD)
	class PolicerData {
```

```
@XmlElement(name="Type")
private int type;
@XmlElement(name="CIR")
private long cir;
@XmlElement(name="CBS")
private long cbs;
@XmlElement(name="EIR")
private long eir;
@XmlElement(name="EBS")
private long ebs;
private int gn_sign;
private int gn_type;
private int comp;
@XmlElement(name="Color")
private int color;

public int getType() {
	return type;
}
public void setType(int type) {
	this.type = type;
}
public long getCir() {
	return cir;
}
public void setCir(long cir) {
	this.cir = cir;
}
public long getCbs() {
	return cbs;
}
public void setCbs(long cbs) {
	this.cbs = cbs;
}
public long getEir() {
	return eir;
}
public void setEir(long eir) {
	this.eir = eir;
}
public long getEbs() {
	return ebs;
}
}
```

```
		public void setEbs(long ebs) {
			this.ebs = ebs;
		}
		public int getGn_sign() {
			return gn_sign;
		}
		public void setGn_sign(int gn_sign) {
			this.gn_sign = gn_sign;
		}
		public int getGn_type() {
			return gn_type;
		}
		public void setGn_type(int gn_type) {
			this.gn_type = gn_type;
		}
		public int getComp() {
			return comp;
		}
		public void setComp(int comp) {
			this.comp = comp;
		}
		public int getColor() {
			return color;
		}
		public void setColor(int color) {
			this.color = color;
		}
	}

}
```

meaPortMapping.java

```
package net.i2cat.enet38xx.client.model;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
```

```
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
@XmlRootElement(namespace="http://www.ethernity-net.com/enet/igressPortMapping")
@XmlAccessorType(XmlAccessType.FIELD)
public class MeaPortMapping {

	private List<PortMapping> portMapping;

	public List<PortMapping> getPortMapping() {
		return portMapping;
	}

	public void setPortMapping(List<PortMapping> portMapping) {
		this.portMapping = portMapping;
	}

	@XmlType(name="portMapping", namespace="http://www.ethernity-
net.com/enet/igressPortMapping", propOrder={"unit", "port"})
	@XmlAccessorType(XmlAccessType.FIELD)
	class PortMapping {

		private int unit;
		private int port;

		public int getUnit() {
			return unit;
		}

		public void setUnit(int unit) {
			this.unit = unit;
		}

		public int getPort() {
			return port;
		}

		public void setPort(int port) {
			this.port = port;
		}
	}
}
```

13 Appendix E – ARN JXB bindings

```
<?xml version="1.0" encoding="utf-8"?>
<jxb:bindings
xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
jxb:extensionBindingPrefixes="xjc"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/jaxb
http://java.sun.com/xml/ns/jaxb/bindingschema_2_0.xsd"
version="2.1" >
      <jxb:globalBindings localScoping="toplevel">
            <xjc:simple />
      </jxb:globalBindings>

      <!-- Force different class names for types with same name (lagMemberPort) -->
      <jxb:bindings schemaLocation="../xml-schema/EquipmentEntities.xsd">
            <jxb:bindings
node="//xs:complexType[@name='InterfaceLag']/xs:choice/xs:element[@name='lagMemberPort
']/xs:complexType">
                  <jxb:class name="LagMemberPortInInterfaceLag"/>
            </jxb:bindings>
      </jxb:bindings>
      <jxb:bindings schemaLocation="../xml-schema/EquipmentEntities.xsd">
            <jxb:bindings
node="//xs:complexType[@name='TLagMemberAttach']/xs:sequence/xs:element[@name='lagMemb
erPort']/xs:complexType">
                  <jxb:class name="LagMemberPortInTLagMemberAttach"/>
            </jxb:bindings>
      </jxb:bindings>

      <!-- Use valid enum member names -->
      <!-- This is a work-around for issue  https://java.net/jira/browse/JAXB-961 -->
      <jxb:bindings schemaLocation="../xml-schema/Main_configurations.xsd">
            <!-- ToperationEntity valid enum values (without slashes '/') -->
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
equipment/system']">
                  <jxb:typesafeEnumMember name="EQUIPMENT_SYSTEM"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
card/fan']">
                  <jxb:typesafeEnumMember name="CARD_FAN"/>
```

```
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/lag']">
                  <jxb:typesafeEnumMember name="INTERFACE_LAG"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/lagPort']">
                  <jxb:typesafeEnumMember name="INTERFACE_LAGPORT"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/ethernet']">
                  <jxb:typesafeEnumMember name="INTERFACE_ETHERNET"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/rf']">
                  <jxb:typesafeEnumMember name="INTERFACE_RT"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/gpon']">
                  <jxb:typesafeEnumMember name="INTERFACE_GPON"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/lag/port']">
                  <jxb:typesafeEnumMember name="INTERFACE_LAG_PORT"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/gpon/ontList']">
                  <jxb:typesafeEnumMember name="INTERFACE_GPON_ONTLIST"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/cos']">
                  <jxb:typesafeEnumMember name="INTERFACE_COS"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
interface/xsfp']">
                  <jxb:typesafeEnumMember name="INTERFACE_XSFP"/>
            </jxb:bindings>
```

```
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
card/rfOverlayModule']">
                  <jxb:typesafeEnumMember name="CARD_RFOVERLAYMODULE"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
clock/ntp']">
                  <jxb:typesafeEnumMember name="CLOCK_NTP"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
clock/timeZone']">
                  <jxb:typesafeEnumMember name="CLOCK_TIMEZONE"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
clientService/gem']">
                  <jxb:typesafeEnumMember name="CLIENTSERVICE_GEM"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
clientService/igmp']">
                  <jxb:typesafeEnumMember name="CLIENTSERVICE_IGMP"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
clientService/dhcp']">
                  <jxb:typesafeEnumMember name="CLIENTSERVICE_DHCP"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
clientService/byRequest']">
                  <jxb:typesafeEnumMember name="CLIENTSERVICE_BYREQUEST"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
clientService/tContMap']">
                  <jxb:typesafeEnumMember name="CLIENTSERVICE_TCONTMAP"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
networkService/igmp']">
                  <jxb:typesafeEnumMember name="NETWORKSERVICE_IGMP"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
networkService/dhcp']">
```

```
                    <jxb:typesafeEnumMember name="NETWORKSERVICE_DHCP"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
networkService/byRequest']">
                    <jxb:typesafeEnumMember name="NETWORKSERVICE_BYREQUEST"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
multicast/probe']">
                    <jxb:typesafeEnumMember name="MULTICAST_PROBE"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
multicast/group']">
                    <jxb:typesafeEnumMember name="MULTICAST_GROUP"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
multicast/igmpProxy']">
                    <jxb:typesafeEnumMember name="MULTICAST_IGMPPROXY"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
multicast/igmpQuerier']">
                    <jxb:typesafeEnumMember name="MULTICAST_IGMPQUERIER"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
multicast/activeGroup']">
                    <jxb:typesafeEnumMember name="MULTICAST_ACTIVEGROUP"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
multicast/activeGroupUser']">
                    <jxb:typesafeEnumMember name="MULTICAST_ACTIVEGROUPUSER"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
firmware/ontList']">
                    <jxb:typesafeEnumMember name="FIRMWARE_ONTLIST"/>
                </jxb:bindings>
                <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
acl/rule']">
                    <jxb:typesafeEnumMember name="ACL_RULE"/>
                </jxb:bindings>
```

```
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
card/port']">
                  <jxb:typesafeEnumMember name="CARD_PORT"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
mgmtAcces/defaultGw']">
                  <jxb:typesafeEnumMember name="MGMTACCESS_DEFAULTGW"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
mgmtAcces/localAccess']">
                  <jxb:typesafeEnumMember name="MGMTACCESS_LOCALACCESS"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
mgmtAcces/inBandAccess']">
                  <jxb:typesafeEnumMember name="MGMTACCESS_INBANDACCESS"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
onu/igmp']">
                  <jxb:typesafeEnumMember name="ONU_IGMP"/>
            </jxb:bindings>
            <jxb:bindings
node="//xs:simpleType[@name='ToperationEntity']/xs:restriction/xs:enumeration[@value='
ethernet/profile/trafficClass']">
                  <jxb:typesafeEnumMember name="ETHERNET_PROFILE_TRAFFICCLASS"/>
            </jxb:bindings>
      </jxb:bindings>

</jxb:bindings>
```

14 Appendix F – Mock ARN test

```
package net.i2cat.dana.ptin_olt.client.api;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringWriter;
import java.net.URISyntaxException;
import java.nio.charset.Charset;
import java.util.List;

import javax.ws.rs.WebApplicationException;
import javax.ws.rs.core.MediaType;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;

import net.i2cat.dana.ptin_olt.client.api.model.Equipment;
import net.i2cat.dana.ptin_olt.client.api.model.Request;
import net.i2cat.dana.ptin_olt.client.api.model.Response;
import net.i2cat.dana.ptin_olt.client.api.model.ToperationEntity;
import net.i2cat.dana.ptin_olt.client.api.model.ToperationType;
import net.i2cat.dana.ptin_olt.client.api.model.TrequestOperation;
import net.i2cat.dana.ptin_olt.client.api.model.TresponseOperation;
import net.i2cat.dana.ptin_olt.client.api.model.Tresult;

import org.apache.commons.io.IOUtils;
import org.apache.cxf.jaxrs.client.JAXRSClientFactory;
import org.custommonkey.xmlunit.XMLAssert;
import org.custommonkey.xmlunit.XMLUnit;
import org.eclipse.jetty.http.HttpStatus;
import org.junit.Assert;
import org.junit.BeforeClass;
import org.junit.Rule;
import org.junit.Test;
import org.xml.sax.SAXException;

import com.github.tomakehurst.wiremock.client.WireMock;
import com.github.tomakehurst.wiremock.junit.WireMockRule;
```

```
/**
 *
 * @author Isart Canyameres Gimenez (i2cat)
 *
 */
public class ClientTest {

	private static final String 			MOCKTESTSERVER_BASE_URL
			= "http://localhost:8080";
	private static final String 			MOCKTESTSERVER_PATH
			= "/cgi-bin/xml-parser.cgi";
	private static final String 			MOCKTESTSERVER_URL
			= MOCKTESTSERVER_BASE_URL + MOCKTESTSERVER_PATH;

	private static final String 			REQUESTS_DIRECTORY
			= "/requests";
	private static final String 			RESPONSES_DIRECTORY
			= "/responses";

	private static final String 			SHOW_EQUIPMENT_WITH_SIMULATOR_REQUEST
	= "/showEquipmentWithSimulator/request.xml";
	private static final String 			SHOW_EQUIPMENT_WITH_SIMULATOR_RESPONSE
	= "/showEquipmentWithSimulator/response.xml";

	private static PTIN_OLT_RPCClientAPI	client;

	@Rule
	public WireMockRule 					wireMockRule
			= new WireMockRule(8080);

	@BeforeClass
	public static void initClient() {
		client = JAXRSClientFactory.create(MOCKTESTSERVER_URL,
PTIN_OLT_RPCClientAPI.class);

		// configuration for XMLUnit and XMLAssert
		XMLUnit.setIgnoreWhitespace(true);
		XMLUnit.setIgnoreAttributeOrder(true);
		XMLUnit.setIgnoreComments(true);
		XMLUnit.setNormalize(true);
		XMLUnit.setNormalizeWhitespace(true);
	}

	/**
	 * Sends a request asking for the equipment to a MOCK TESTSERVER and parses response.
	 *
	 * This test illustrates how the client is meant to be used.
	 *
	 * @throws Exception
```

```
	 */
	@Test
	public void requestShowEquipments() throws Exception {

		String requestBody =
IOUtils.toString(this.getClass().getResource(SHOW_EQUIPMENT_WITH_SIMULATOR_REQUEST));
		String responseBody =
IOUtils.toString(this.getClass().getResource(SHOW_EQUIPMENT_WITH_SIMULATOR_RESPONSE));
		mockServer(requestBody, responseBody);

		List<Equipment> equipments = getEquipment();
		Assert.assertNotNull(equipments);
		Assert.assertFalse(equipments.isEmpty());
	}

	/**
	 * Test serialization and deserialization for each sample request in REQUESTS_DIRECTORY
folder. This test checks for each sample that request
	 * serialization produces XML that's equivalent to the sample one.
	 *
	 * @throws JAXBException
	 * @throws IOException
	 * @throws SAXException
	 * @throws TransformerException
	 * @throws ParserConfigurationException
	 * @throws URISyntaxException
	 */
	@Test
	public void requestSerializationDeserializationTest()
			throws JAXBException, IOException, SAXException, TransformerException,
ParserConfigurationException, URISyntaxException {

		java.net.URL folderUrl = this.getClass().getResource(REQUESTS_DIRECTORY);
		File folder = new File(folderUrl.toURI());
		StringBuffer report = new StringBuffer();
		report.append("Proccessed samples: " + System.getProperty("line.separator"));
		try {
			for (File fileEntry : folder.listFiles()) {
				if (!fileEntry.isDirectory()) {
					if (fileEntry.getName().endsWith(".xml")) {
						InputStream in = new FileInputStream(fileEntry);
						String xmlRequest = IOUtils.toString(in);
						Request request = loadRequestFromXML(new
ByteArrayInputStream(xmlRequest.getBytes(Charset.forName("UTF-8"))));
						String generatedXmlRequest = toXml(request);
						if (!XMLUnit.compareXML(xmlRequest,
generatedXmlRequest).similar()) {
							System.out.println("ORIGINAL: " +
xmlRequest);
```

```
                                                System.out.println("GENERATED: " +
generatedXmlRequest);
                                                System.out.println("Differences found in
request " + fileEntry.getName());
                                        }
                                        XMLAssert.assertXMLEqual(xmlRequest,
generatedXmlRequest);
                                        report.append(fileEntry.getName() +
System.getProperty("line.separator"));
                                    }
                            }
                    }
            } finally {
                    System.out.println(report.toString());
            }
    }

    /**
     * Test serialization and deserialization for each sample response in
RESPONSES_DIRECTORY folder. This test checks for each sample that response
     * serialization produces XML that's equivalent to the sample one.
     *
     * @throws JAXBException
     * @throws IOException
     * @throws SAXException
     * @throws TransformerException
     * @throws ParserConfigurationException
     * @throws URISyntaxException
     */
    @Test
    public void responseSerializationDeserializationTest()
                    throws JAXBException, IOException, SAXException, TransformerException,
ParserConfigurationException, URISyntaxException {

            java.net.URL folderUrl = this.getClass().getResource(RESPONSES_DIRECTORY);
            File folder = new File(folderUrl.toURI());
            StringBuffer report = new StringBuffer();
            report.append("Proccessed samples: " + System.getProperty("line.separator"));
            try {
                    for (File fileEntry : folder.listFiles()) {
                            if (!fileEntry.isDirectory()) {
                                    if (fileEntry.getName().endsWith(".xml")) {
                                            InputStream in = new FileInputStream(fileEntry);
                                            String xmlResponse = IOUtils.toString(in);
                                            Response response = loadResponseFromXML(new
ByteArrayInputStream(xmlResponse.getBytes(Charset.forName("UTF-8"))));
                                            String generatedXml = toXml(response);
                                            if (!XMLUnit.compareXML(xmlResponse,
generatedXml).similar()) {
```

```
                                                System.out.println("ORIGINAL: " +
xmlResponse);
                                                System.out.println("GENERATED: " +
generatedXml);
                                                System.out.println("Differences found in
response " + fileEntry.getName());
                                        }
                                        XMLAssert.assertXMLEqual(xmlResponse, generatedXml);
                                        report.append(fileEntry.getName() +
System.getProperty("line.separator"));
                                    }
                            }
                    }
            } finally {
                    System.out.println(report.toString());
            }
    }

    private Request loadRequestFromXML(InputStream in) throws JAXBException {
            return fromXml(in, Request.class);
    }

    private Response loadResponseFromXML(InputStream in) throws JAXBException {
            return fromXml(in, Response.class);
    }

    /**
     * Deserialize the XML InputStream into an instance of provided class
     *
     * @param xml
     * @param objectClass
     * @return
     * @throws JAXBException
     */
    private static <T> T fromXml(InputStream xml, Class<T> objectClass) throws JAXBException
{
            JAXBContext context = JAXBContext.newInstance(objectClass);
            Object obj = context
                            .createUnmarshaller().unmarshal(xml);
            return (T) obj;
    }

    private static String toXml(Object obj) throws JAXBException {
            StringWriter sw = new StringWriter();
            JAXBContext context = JAXBContext.newInstance(obj.getClass());
            Marshaller m = context.createMarshaller();

            m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
            m.marshal(obj, sw);
```

```
		return sw.toString();
	}

	/**
	 * FIXME This method should probably be a service itself.
	 *
	 * @return
	 * @throws Exception
	 */
	private List<Equipment> getEquipment() throws Exception {

		Long opId = 1L;

		TrequestOperation op = new TrequestOperation();
		op.setToken(opId);
		op.setType(ToperationType.SHOW);
		op.setEntity(ToperationEntity.EQUIPMENT_SYSTEM);
		op.setEquipment(new Equipment());

		Request request = new Request();
		request.getOperations().add(op);

		System.out.println(toXml(request));

		Response response = null;
		try {
			response = client.sendRequest(request);
		} catch (WebApplicationException ex) {
			// following response will only be available
			// when the exception has been produced after receiving the response (i.e.
while parsing inside the client proxy)
			javax.ws.rs.core.Response r = ex.getResponse();
			System.out.println(r);
			// FIXME probably better to throw a protocol exception or something
similar
			throw ex;
		}

		System.out.println(toXml(response));

		TresponseOperation rop = getOperationByToken(response, opId);
		if (!isOkResult(rop.getResult())) {
			// FIXME report operation failure, probably throwing a specific exception
for this purpose
		}

		return rop.getEquipmentList().getEquipments();
	}
```

```
	/**
	 * Returns the TresponseOperation with given token that is part of given response, or
null if any.
	 *
	 * FIXME This method should probably be placed in a controller or utils class
	 *
	 * @param response
	 * @param token
	 * @return TresponseOperation with given token in given response. null if there is no
such TresponseOperation in given response.
	 */
	private static TresponseOperation getOperationByToken(Response response, Long token) {
		for (TresponseOperation op : response.getOperations()) {
			if (op.getToken().equals(token))
				return op;
		}
		return null;
	}

	/**
	 * FIXME This method should probably be plpaced in a controller or utils class
	 *
	 * @param result
	 * @return
	 */
	private static boolean isOkResult(Tresult result) {
		String okCode = "00000000";
		if (okCode.equals(result.getError()))
			return true;
		return false;
	}

	private void mockServer(String requestMessageBody, String responseMessageBody) {
		WireMock.stubFor(
				WireMock.post(
					WireMock.urlEqualTo(MOCKTESTSERVER_PATH))
					.withHeader("Content-Type",
WireMock.equalTo(MediaType.APPLICATION_XML))

	.withRequestBody(WireMock.equalToXml((requestMessageBody)))
					.willReturn(WireMock.aResponse()
							.withStatus(HttpStatus.OK_200)
							.withHeader("Content-Type",
MediaType.APPLICATION_XML)
							.withBody(responseMessageBody))
				);
	}
```

```
}
```