(FP7 614100)

D7.2.2 Integrated Component Platform

Published by the IMPReSS Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme and the Conselho Nacional de Desenvolvimento Científico e Tecnológico
Objective ICT-2013.10.2 EU-Brazil research and development Cooperation

Document control page

Document file: D7.2.2_Integrated_Component_Platform_1.0.docx
Document version: 1.0
Document owner: Peeter Kool (CNET)

Work package: WP7 – IDE Framework for Model-driven development
Task: T7.2 – Components Integration
Deliverable type: P

Document status: x approved by the document owner for internal review
x approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Peeter Kool (CNET)	2015-12-01	Initial ToC
0.5	Peeter Kool, Peter Rosengren (CNET)	2015-12-20	Initial Content
0.9	Peeter Kool (CNET)	2016-03-30	Additional content, editing and spell check, versions for internal review
1.0	Peter Rosengren, Peeter Kool (CNET)	2016-03-31	Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Enrico Ferrera (ISMB)	2016-03-31	Accepted with minor comments

Index:

1. Executive summary

This deliverable describes the final prototype deliverable developed within task T7.2 Components Integration. The first section is a short introduction. The second section gives an overview of the available components and their interfaces. The third section describes the tools that were specifically created or extended but are not reported in other deliverables. Finally, there is a section with some conclusions.

2. Introduction

The aim of the task T7.2 Components Integration is to integrate the components from WP3-6 after having defined the interfaces and protocols for the information exchange between application systems and the IMPReSS middleware.

This prototype deliverable aims at providing pointers to the relevant deliverables containing the APIs and tools that is part of the IMPReSS platform. Furthermore, additional APIs and tools which have been developed but are not part of the other deliverables are described here.

Finally, there is a conclusions section which discusses some "lessons learned" with regards to the integration process. Taking into account that the project is distributed both geographically and in time zones.

3. Overview of components and interfaces

This section will describe the different integrated components and their interfaces. The full description of the components is available in their respective work package deliverables.

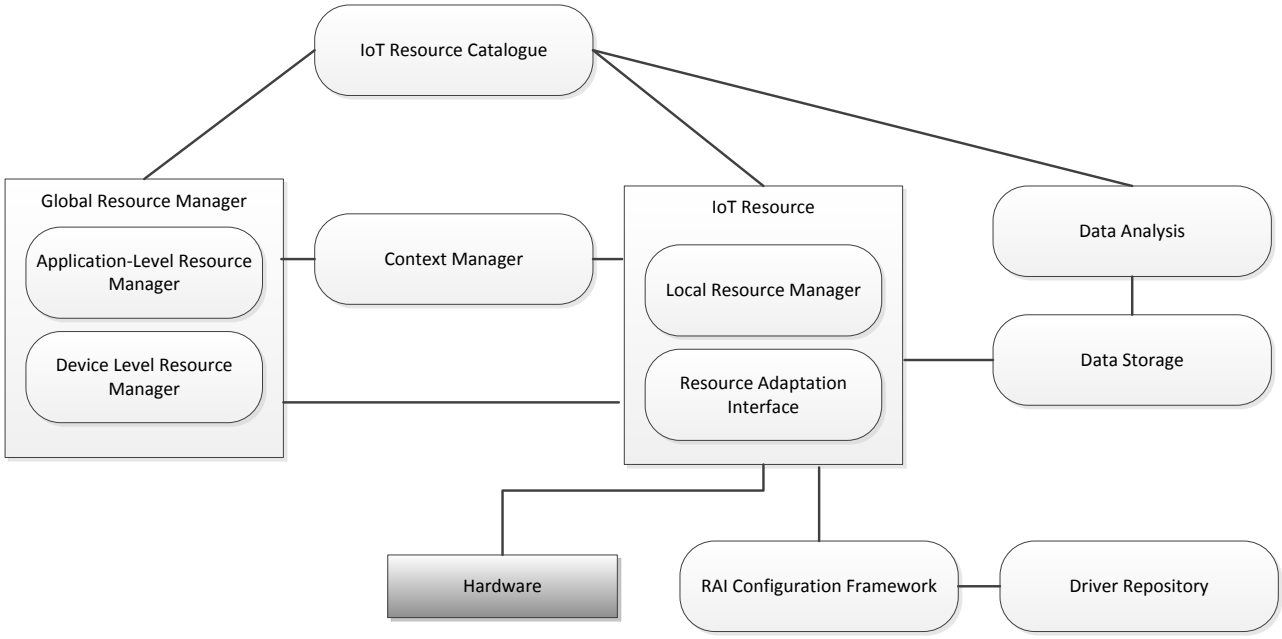

Figure 1: Overview of the main components in the IMPReSS architecture

Figure 1 shows the main important components in the IMPReSS platform, which consist of:

- IoT Resource Catalogue: The IoT Resource Catalogue provides mechanisms for routing requests regarding the physical world to the appropriate software resource that is able to fulfill the request, such as reporting the temperature in a particular office space or the current electricity consumption of a household appliance.
- Global Resource Manager: Consist of two main components:
  - Application-Level Resource Manager: Provides resource management at the application level.
  - Device-Level Resource Management: Provides resource management at the device level.
- Context Manager: This component encompasses all background software components that a typical context-aware middleware offers to its users such as context templates, context models, context reasoning engine, and algorithms for sensor and data fusion
- IoT Resource: Consists of two main components:
  - Local Resource Manager: Controls access and sharing of resources.
  - Resource Adaptation Interface (RAI): The actual interfacing with resources.
- RAI Configuration Framework and Integration Support Tool: Used for setting up the RAI with relevant drivers, also called Device Managers, and configuring the resources.
- Driver Repository: Repository of available RAI Device Managers for different protocols and devices.

- Data Analysis: This module provides all software components needed to implement data analysis and historic context information that will be used by IMPReSS applications.
- Data Storage: Provides data storage and data retrieval mechanism.

The common integration in-between the different modules have been done using APIs which were either REST-based, i.e. HTTP, or by using MQTT (MQTT, 2016). MQTT is a lightweight publish/subscribe messaging transport which is widely used and supported in the IoT World. It has the additional benefit that it can be used when needing to bridge networks, i.e. different networks using NAT can still connect and communicate with each other.

Most of these interfaces are described in previous deliverables and therefore we will provide an overview and refer to actual deliverable describing the API, though additions that have been made to the API during the integration phase will be described more in full, also including MQTT messages that are used.

3.1 IoT Resource Catalogue

Main API and documentation are available in deliverable D3.2 (IMPReSS_ D3.2, 2015) but there is some additional mechanism that have been added during the integration after the delivery of the document. The main additions are:

- Support for Resource Discovery using MQTT messaging.
- Additional MQTT messages that are created when the IoT Resource Catalogue state changes.
- Additions of new metadata the SCPD discovery document.

3.1.1 MQTT Discovery

In order to facilitate discovery over MQTT the IoT Resource Catalogue subscribes to the following topics:

- /<Application Name>/control/newiotresource/<Resource id> which is used for registering the resource. The payload of the message is the SCPD discovery document. See Code Listing 1.
- /<Application Name>/control/removeiotresource/<Resource id> which is used for deregistering the resource. This message lacks payload.

The <Application Name> can be used to distinguish different applications but is optional.

3.1.2 MQTT Messages from the IoT Resource Catalogue

Instead of always calling the IoTResource Catalogue to check the status of resources we added a messaging API that sends out messages when the state changes. For instance, it is possible to get a message each time a new resource is discovered by the IoT Resource Catalogue, instead of querying the catalogue at intervals. Table 1 shows the topics and the payloads for the messages created by the IoT Resource Catalogue.

Event	Topic	Payload
Resource Discovered	/IMPRESS/System/NewIoTResource	{ "resource id":

		"http://purl.oclc.org/impress/rai/occupanysensor54", "type": ["OccypancySensor"], "friendly name": "Occypancy sensor x" }

Resource removed	/IMPRESS/System/ ObsoleteIoTResource	{
		"resource id":
		"http://purl.oclc.org/impress/rai/kinectsensor_1"
		}

IoTEntity Created	/IMPRESS/System/NewIoTEntity	{
		"entity id":
		"urn:uuid:f47ac10b-58cc-4372-a567-0e02b2c3d479",
		"type": "Place",
		"friendly name": "Row 10 in classroom TS1204"
		}

IoTEntity removed	/IMPRESS/System/ObsoleteIoTEntity	{
		"entity id":
		"urn:uuid:f47ac10b-58cc-4372-a567-0e02b2c3d479"
		}

Resource Associated to IoTEntity	/IMPRESS/System/NewAssociation	{
		"resource id":
		"http://purl.oclc.org/impress/rai/occupanysensor54",
		"entity id ":
		"urn:uuid:f47ac10b-58cc-4372-a567-0e02b2c3d479",
		"description":
		"Occupancy sensor monitors row 10 in the classroom TS1204"
		}

Resource association To IoTEntity removed	/IMPRESS/System/ObsoleteAssociation	{
		"resource id":
		"http://purl.oclc.org/impress/rai/occupanysensor54",
		"entity id ":
		"urn:uuid:f47ac10b-58cc-4372-a567-0e02b2c3d479"
		}

Table 1: New IoT Resource Catalogue MQTT messages

3.1.3 Additions to the SCPD discovery document

The SCPD discovery document is used when registering a resource in the IoT Resource Catalogue. The document describes the resource and its services, see Code Listing 1 below.

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
        <specVersion>
                <major>1</major>
                <minor>0</minor>
        </specVersion>
        <device>
```

```
                <!--UPnP Elements-->
                <deviceType>urn:schemas-upnp-org:IoTdevice:meter:1</deviceType>
                <presentationURL>http://130.192.85.32:8080/</presentationURL>
                <friendlyName>Plugwise Meter 11EAF9-INSTANT_POWER</friendlyName>
                <manufacturer>Plugwise</manufacturer>
                <manufacturerURL>http://www.ismb.it</manufacturerURL>
                <modelDescription>A Meter device connected to a Plugwise network.</modelDescription>
                <modelName>meter</modelName>
                <modelNumber>1</modelNumber>
                <UDN>uuid:e5268dbd-4444-32dc-bfc9-f2721ea60a59</UDN>

                <!--CNet extension Elements-->
                <IoTresourceid xmlns="IoT">e5268dbd-4444-32dc-bfc9-f2721ea60a59</IoTresourceid>
                <gateway xmlns="IoT">130.192.85.32:8080</gateway>
                <errormessage xmlns="IoT"></errormessage>
                <supportedCommand xmlns="IoT">
                {turnOn=it.ismb.pertlab.pwal.api.devices.commands.impl.TurnOnMeter@4b34d86b,
                turnOff=it.ismb.pertlab.pwal.api.devices.commands.impl.TurnOffMeter@7a6c7f5f}
                </supportedCommand>
                <id xmlns="IoT">11EAF9-INSTANT_POWER</id>
                <configuration xmlns="IoT">it.ismb.pertlab.pwal.api.xmpp.PicForm@199a9e30</configuration>
                <power xmlns="IoT">0.0</power>
                <location xmlns="IoT">106.0 718.0</location>
                <networkType xmlns="IoT">Plugwise</networkType>
                <expiresAt xmlns="IoT">2015-11-16T09:50:57.325Z</expiresAt>
                <pwalId xmlns="IoT">e5268dbd-4444-32dc-bfc9-f2721ea60a59</pwalId>
                <type xmlns="IoT">Meter</type>
                <updatedAt xmlns="IoT">2015-11-16T10:49:57.356+01:00</updatedAt>
                <lastMeasurement xmlns="IoT">2015-11-16T10:49:57.356+01:00</lastMeasurement>
                <mqttBrokerIP xmlns="IoT">tcp://130.192.85.32</mqttBrokerIP>
                <mqttBrokerPort xmlns="IoT">1883</mqttBrokerPort>
                <mqttTopics xmlns="IoT">
                        <mqttTopic type="metadata" xmlns="IoT">
                        /impress/metadata/e5268dbd-4444-32dc-bfc9-f2721ea60a59
                        </mqttTopic>
                        <mqttTopic type="observation" xmlns="IoT">
                        /impress/observation/e5268dbd-4444-32dc-bfc9-f2721ea60a59
                        </mqttTopic>
                </mqttTopics>
                <serviceList>
                        <service>
                                <serviceType>urn:schemas-upnp-org:smartplugservice::1</serviceType>
                                <serviceId>urn:schemas-upnp-org:meterservice_e5268dbd-4444-32dc-bfc9-
f2721ea60a59:1</serviceId>
                                <SCPDURL>_urn:schemas-upnp-org:tmeterservice_e5268dbd-4444-32dc-bfc9-
f2721ea60a59:1_scpd.xml</SCPDURL>
                                <controlURL>_urn:schemas-upnp-org:meterservice_e5268dbd-4444-32dc-bfc9-
f2721ea60a59:1_control</controlURL>
                                <eventSubURL>http://127.0.0.1:44442/_urn:schemas-upnp-
org:meterservice_e5268dbd-4444-32dc-bfc9-f2721ea60a59:1_event</eventSubURL>
                        </service>
                </serviceList>
        </device>
</root>
```

Code Listing 1: Example of an SCPD discovery document

In the SCPD discovery document the MQTT metadata for the resource was added. This includes which topics the resource publishes data as well as which MQTT broker is used by the resource for publishing. Code Listing 1 highlights the additions in bold text.

The additions are:

- **mqttBrokerIP:** IP-address of the MQTT broker to which the resource publishes events
- **mqttBrokerPort:** The port used by the MQTT broker
- **mqttTopics/mqttTopic:** The topics that the resource uses when publishing events. Each of the topics have a @type attribute which indicates which type of messages are published using the topic. Currently two type are used:

- **metadata:** Messages containing meta data concerning the resource.
- **observation:** Messages containing actual observations/measurements from the resource

When a client finds a resource in the catalogue it can immediately start to subscribe to the published messages from the resource knowing which topics are used as well as the address of the MQTT broker where the messages are published.

3.2 Global Resource Manager

The APIs for the Global Resource Manager are described in the deliverable D4.3 (IMPReSS_D4.3, 2015). Additionally, the Global Resource Manager also subscribes to and supports the MQTT messages and topics described in Table 1.

3.3 Context Manager

The Context Manager API is described in appendix B of deliverable D6.4 Implementation of Context Reasoning Engine (IMPReSS_D6.4, 2015). The tools for creating and managing the context are described in deliverable D6.5 Implementation of Context Modelling Tool and Templates (IMPReSS_D6.5, 2015).

3.4 IoT Resource

IoT Resource consists of two main components:

- Local Resource Manager
- Resource Adaptation Interface (RAI)

3.4.1 Local Resource Manager

The Local Resource Manager API is described in deliverable D4.2 Device and Subsystem Resource Management (IMPReSS_D4.2, 2014). In addition to this API the Local Resource Manager also publishes messages to the MQTT broker, there are two main types of massages: metadata and observation.

Metadata messages contain information about the resource, such as a descriptive name, device type, properties, unit of measurement. The topic for the metadata messages are **/impress/metadata/<resourceid>** and an example payload is shown in Code Listing 2 below.

```
{
  "Name": "98A884-INSTANT_POWER",
  "Meta": [
    {
      "property": "impress:networkType",
      "Value": "Plugwise"
    },
    {
      "property": "geo:point",
      "Value": "102.0 862.0"
    }
  ],
  "Properties": [
    {
      "Name": "Power Consumed",
      "UnitOfMeasurement": {
        "property": "W",
        "TypeOf": [
          "Power Consumed"
        ]
      },
      "About": "f061b5da-7ddd-3510-95fc-15d61e870f26:Meter:getPower",
```

```
      "TypeOf": [
        "impress:Meter:getPower"
      ],
      "DataType": "Double"
    },
    {
      "Name": "Relay status",
      "UnitOfMeasurement": {
        "TypeOf": [
          "Relay status"
        ]
      },
      "About": "f061b5da-7ddd-3510-95fc-15d61e870f26:Meter:isOn",
      "TypeOf": [
        "impress:Meter:isOn"
      ],
      "DataType": "Boolean"
    }
  ],
  "About": "f061b5da-7ddd-3510-95fc-15d61e870f26",
  "TypeOf": [
    "Meter"
  ]
}
```


Code Listing 2: Example of a metadata message from the Local Resource Manager

Observation messages contain the values of the observations, such as a measured power consumption, status value et c. The topic for the metadata messages are **/impress/observation/<resourceid>** and an example payload is shown in Code Listing 3 below.


```
{
  "About": "f061b5da-7ddd-3510-95fc-15d61e870f26",
  "Properties": [
    {
      "IoTStateObservation": [
        {
          "Value": "0.0",
          "PhenomenonTime": "2015-09-25T09:18:24.005Z",
          "ResultTime": "2015-09-25T09:19:23.984Z"
        }
      ],
      "About": "f061b5da-7ddd-3510-95fc-15d61e870f26:Meter:getPower"
    },
    {
      "IoTStateObservation": [
        {
          "Value": "true",
          "PhenomenonTime": "2015-09-25T09:18:24.005Z",
          "ResultTime": "2015-09-25T09:19:23.984Z"
        }
      ],
      "About": "f061b5da-7ddd-3510-95fc-15d61e870f26:Meter:isOn"
    }
  ]
}
```


Code Listing 3: Example of an observation message from the Local Resource Manager

3.5 Resource Adaption Layer

The Resource Adaption Interface RAI API is not really exposed to the rest of the IMPReSS system since the communication is done through the Local Resource Manager which is the gatekeeper for the RAI. The following deliverables describe the API and tools used for configuring and managing the RAI and its drivers:

- D3.1 Resource Adaptation Interface Framework (IMPReSS_D3.1, 2014)

- D3.5 Templates and Integration Support Tool (IMPReSS_D3.5, 2016)

3.6 RAI Configuration Framework

The RAI Configuration Framework is in charge of the configuration and composition of the different components of the platform in order to initialize them and establish how they have to work together so as to set the correct workflow of the platform.

The framework is composed basically by three different components:

- The Configuration and Composition Manager (CC_M), which is the central component, in charge to compose and configure the different components of the platform, exposing the IMPReSS APIs for this purpose.
- The Configuration and Composition Agents (CC_A), which are the distributed entities of the architecture, located on the components that monitor the associated component and interact with the Manager.
- The Integration Support Tool GUI (see Figure 2), which is the interface provided to the System Managers in order to setup RAI downloading and installing drivers for relevant physical resources.

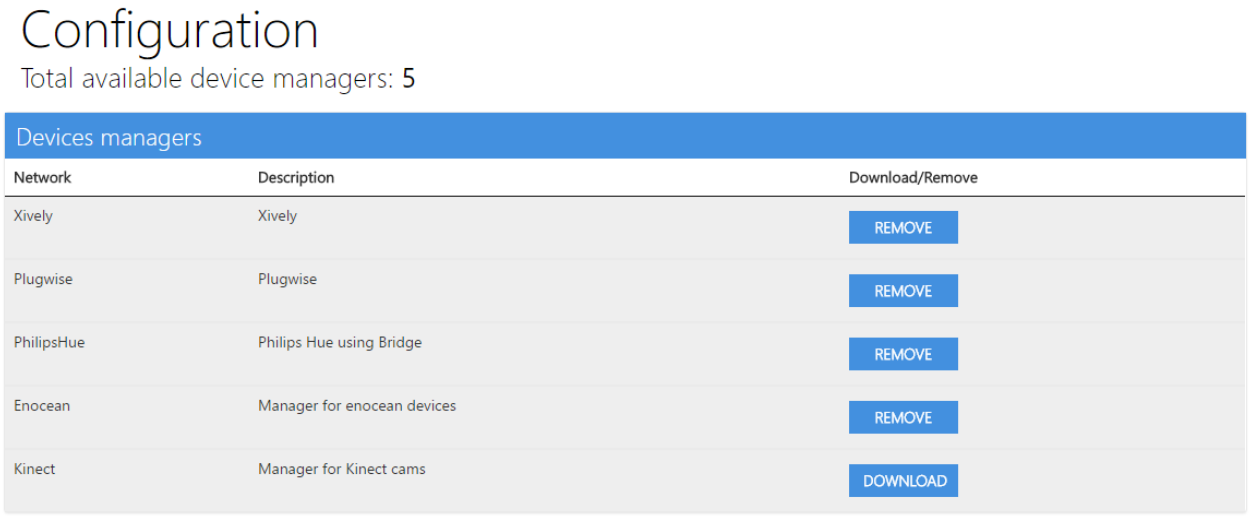

Figure 2: Integration Support Tool GUI

The RAI Configuration Framework is described in the following deliverable:

- D7.3.1 Initial Design and Implementation of the Configuration and Composition Manager (IMPReSS_D7.3.1, 2014)
- D7.3.2 Final Design and Implementation of the Configuration and Composition Manager (IMPReSS_D7.3.2, 2016)

The Integration Support Tool is described in:

- D3.5 Templates and Integration Support Tool (IMPReSS_D3.5, 2016)

3.7 Data Analysis

The Data Analysis API and tools are described in the following deliverables:

- D5.2 Data Analysis and Forecast for Energy Consumption (IMPReSS_D5.2, 2014)
- D5.3 Data Mining and Machine Learning Tools (IMPReSS_D5.3, 2015)
- D5.4 Machine Learning for User Behaviour and Occupancy Analysis (IMPReSS_D5.4, 2015)

3.8 Data Storage

The Data Storage and its API is described in D5.3 Data Mining and Machine Learning Tools (IMPReSS_D5.3, 2015). There is also an online version of the Data Storage API description http://storage.impress.gprt.ufpe.br:8080/ , see below.

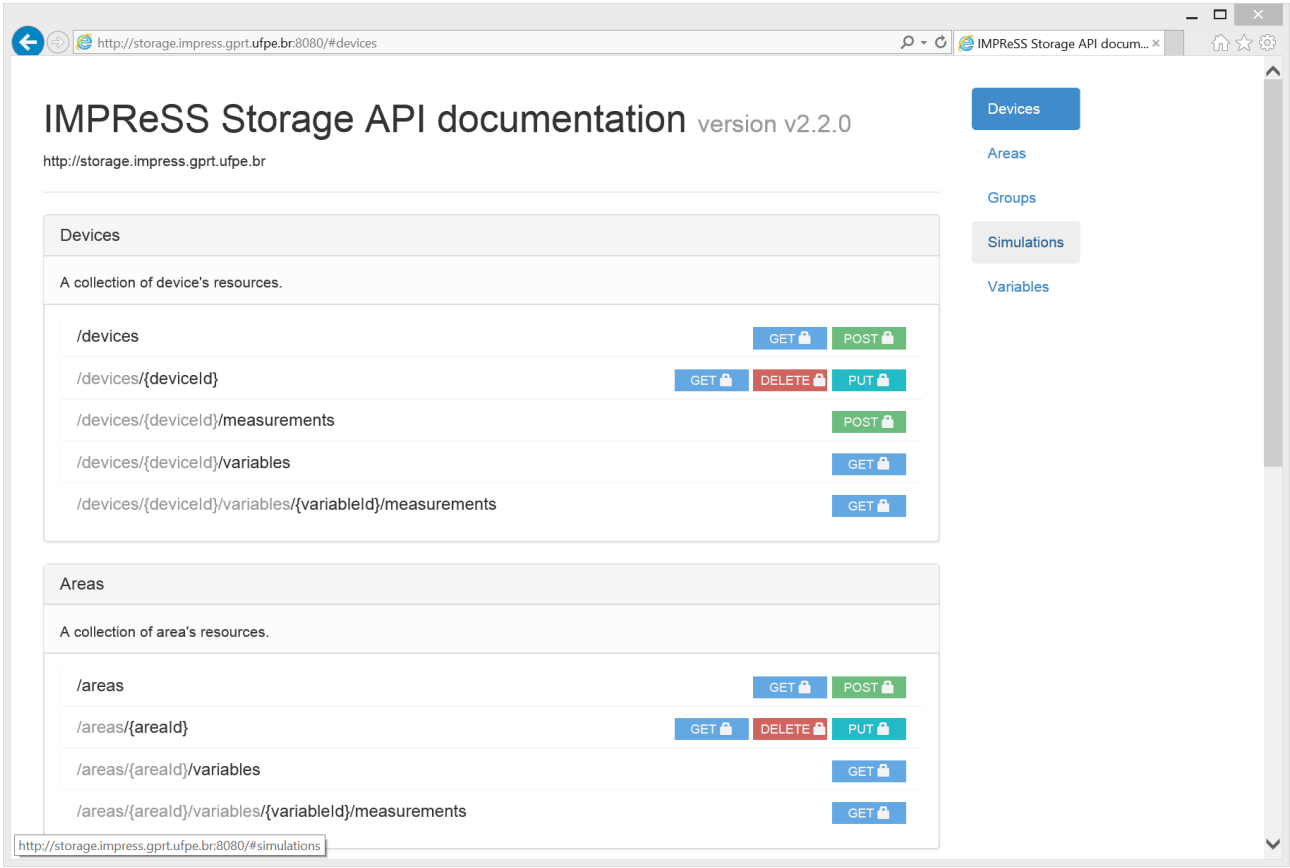

Figure 3: Data Storage online API documentation.

4. Tools

There are some tools that have been developed within the project that are not described in the deliverables and therefore we describe them in this section.

4.1 IoT Resource Catalogue Tools

Two tools have been developed to work with the IoT Resource Catalogue to help navigating the Catalogue contents and how to leverage the API. The two tools are:

- IoT World Gateway UI
- Deployment App

We will shortly describe these tools in the following sub-sections.

4.1.1 IoT World Gateway UI

This tool is a browser based tool that provides functionality to browse around and interact with contents of the IoT Resource Catalogue. The first tab is used for interacting with IoT Resources, see Figure 4.

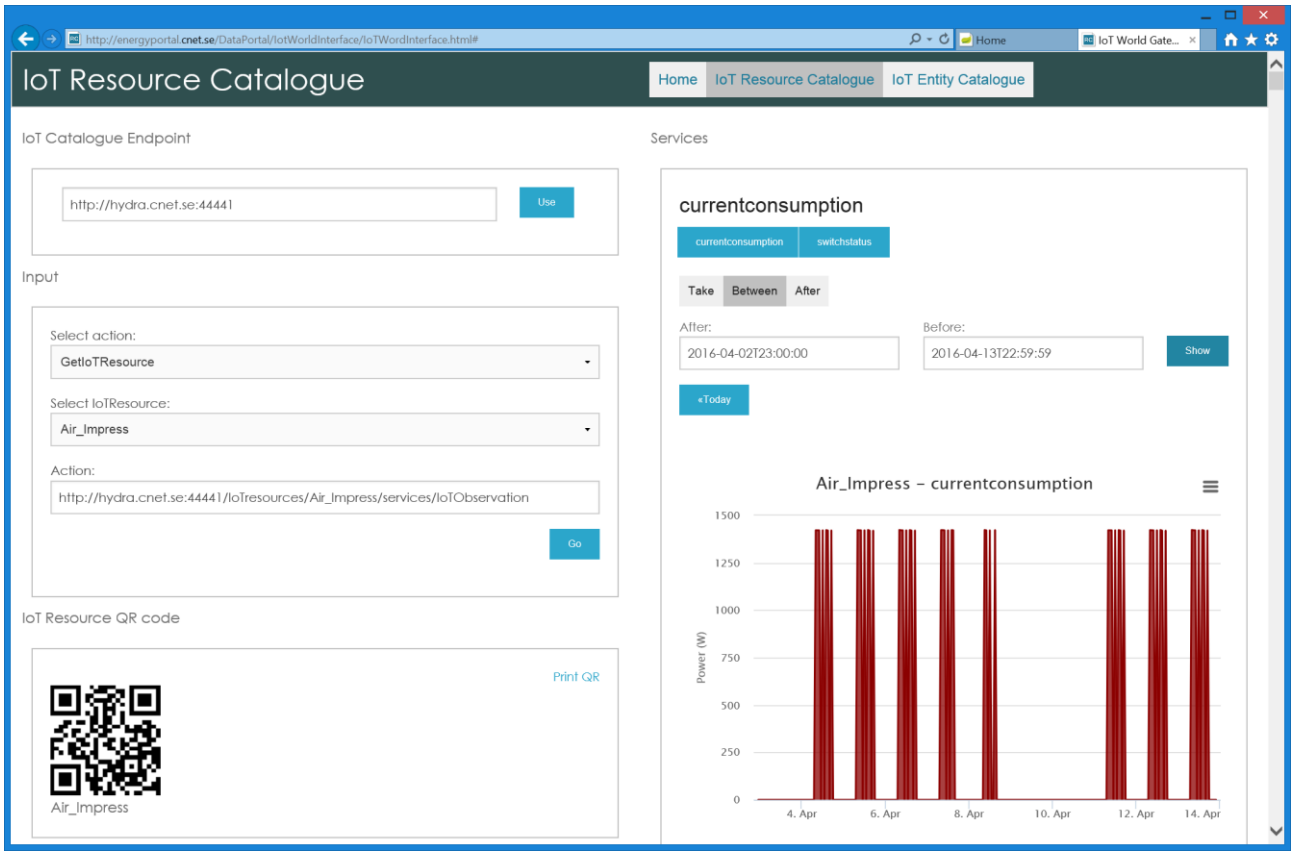

Figure 4: IoT World Gateway UI, IoT Resource Catalogue tab.

The user interface consists of a number of panes which are used for selecting and executing actions:

- IoT Catalogue Endpoint: Is used for selecting which IoT Resource Catalogue we want to work with. In this case it is http://linksmart.cnet.se:44441.
- Input: There one selects which action will be invoked as well as the IoT Resource we want to work with. The different inputs are:

    - Select action: The user selects the appropriate action in a drop down list.
    - Select IoTResource: The user selects an IoTResource from a drop down list that contains all the known IoT Resources in the catalogue. In the example in Figure 4, Air_impress is selected.
    - Clicking Go will perform the action.
- IoTResource QR code: Displays the IoTResources resource id as an QR code that can be printed or just be scanned.
- Services: This pane displays the available services in the selected IoT Resource, depending on the service selected there are different options. In this case we select the currentconsumption service which displays three options:
    - Take: Takes the last N observations
    - Between: Take observations between two dates
    - After: Take all observations after a specific date
    - Clicking Go will create a graph if historical values are available or even plot predicted values if the end date is in the future and prediction data is available.

There is also an additional pane that is not visible in Figure 4 called Output, see Figure 5 below.

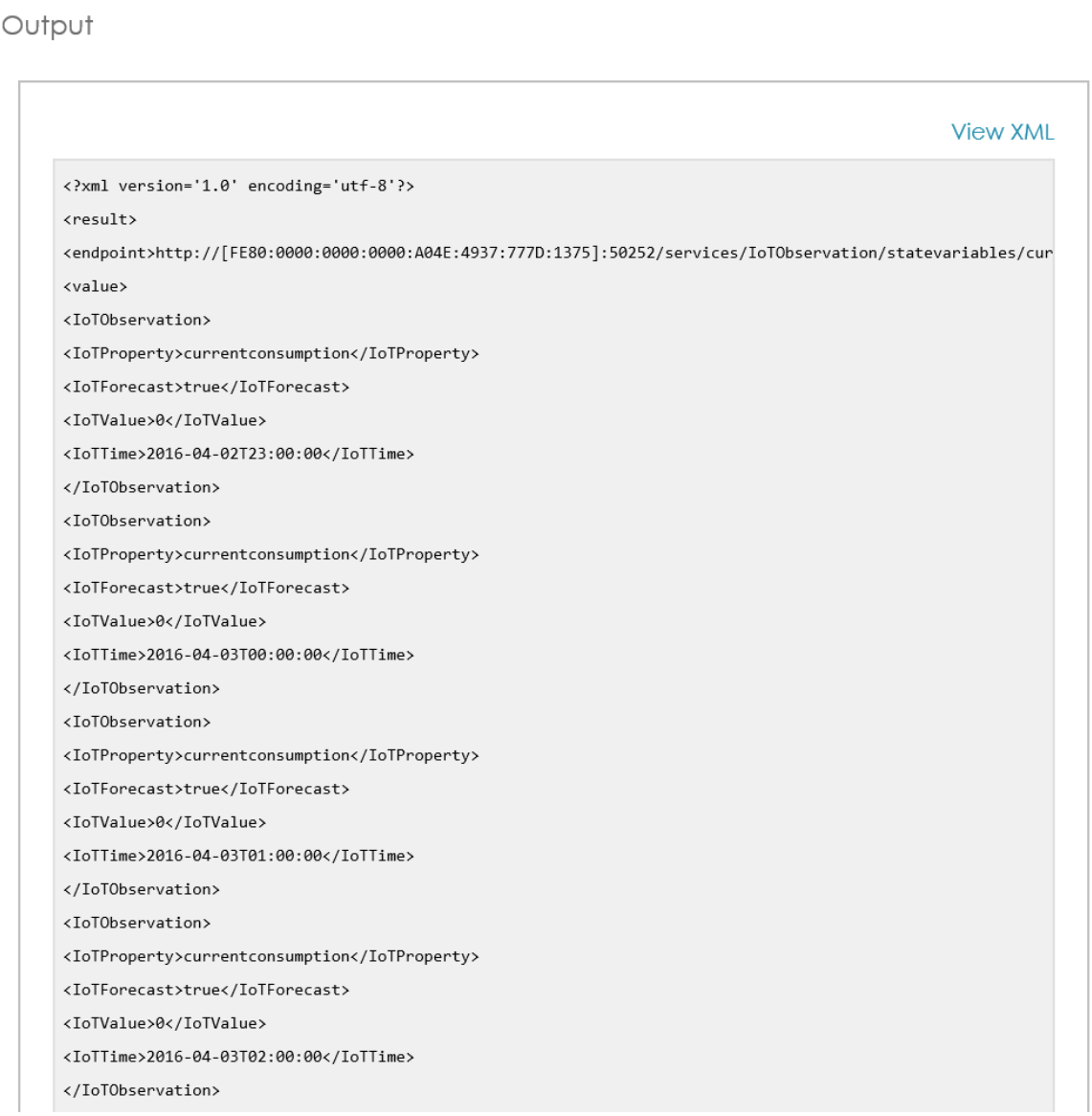

Figure 5: Output pane.

The output pane shows the actual result from invoking the service. In this case we can see the result in XML format. There is also a View XML link which can be used to display the data in a separate window or it can be copied and used in a program since it is the actual link needed to be executed to retrieve this dataset. Please note that the result can also be retrieved in JSON format as well if this is preferred.

The second tab in the IoT World Gateway UI is used for interaction with the IoTEntity model, see Figure 6 below.

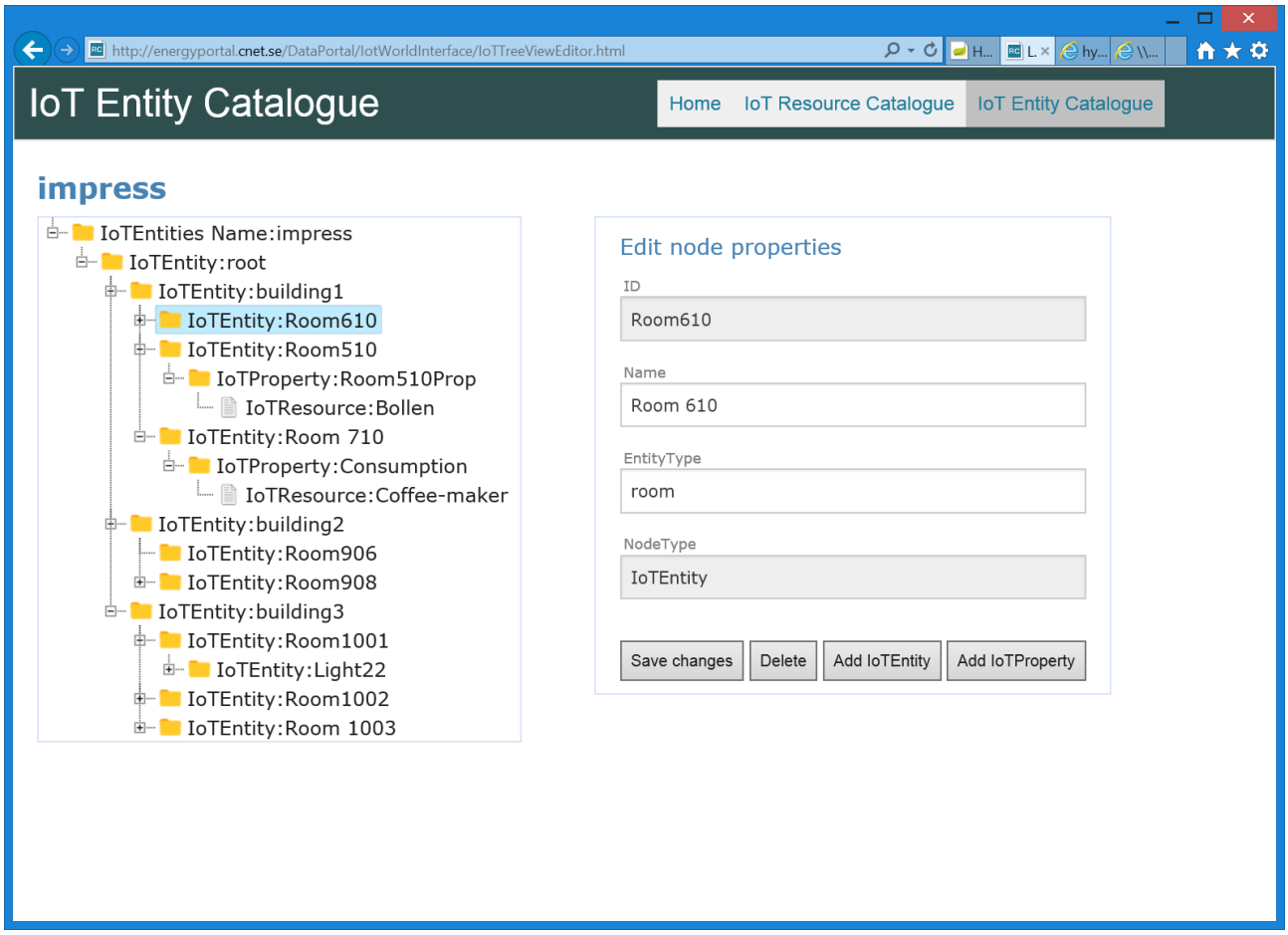

Figure 6: IoT Entity Catalogue Tab

The IoT Entity Catalogue Tool provides the following functionality for the user:

- Browsing the IoT Entity hierarchy, its properties and IoT Resources.
- The possibility to add/edit/delete IoT Entities
- The possibility to add/edit/delete IoT Entity properties and their link to IoT Resources

4.1.2 Deployment app

The deployment app is an iOS app that implements a tool that can be used when deploying IoT Resources and connecting them to the IoT Entity model.

The basic scenario for the deployment app is the following:

1. The user wants to deploy a sensor in a specific room
2. The user starts the deployment app and scan the room QR code
3. The deployment app shows the IoT Entity that matches the QR code, i.e. the IoT Entity ID matches the QR code.
    a. If the QR code does not exist, the user can browse the IoT Entity Catalogue manually in the app to select the right room.
4. The user creates a new property on the IoT Entity and gives it a name.
5. The user adds the IoT Resource to the property by scanning the QR code on the sensor.

6. The deployment app will show the matching IoT Resource that matches the QR code, i.e. The IoT Resource ID matches the QR code.
    a. If the QR code does not exist, the user can browse the IoT Resource Catalogue manually in the app to select the IoT Resource.
7. The user saves the change.

The user interface of the deployment app is quite straightforward as can be seen in the following screen dumps.

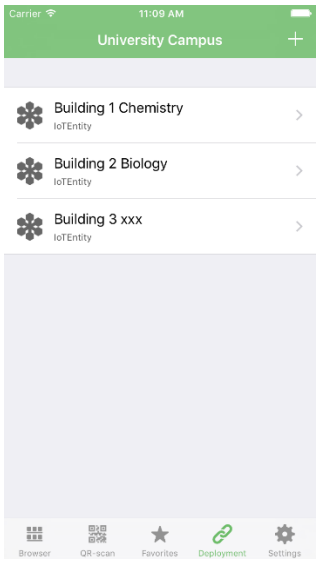

Figure 7: Deployment Tool IoT Entity browsing

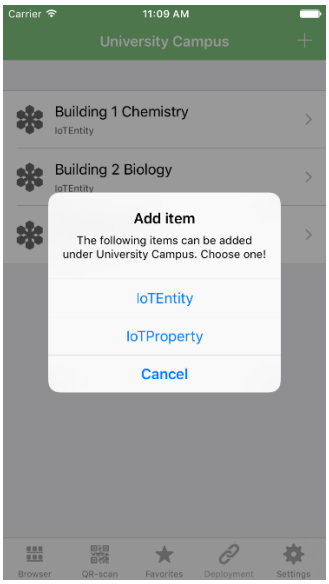

Figure 8: Deployment Tool IoT Entity add property

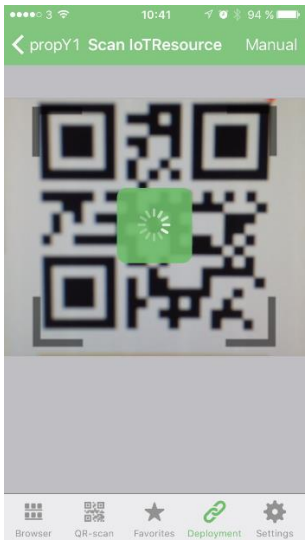

Figure 9: Deployment Tool Scan IoT Resource QR code

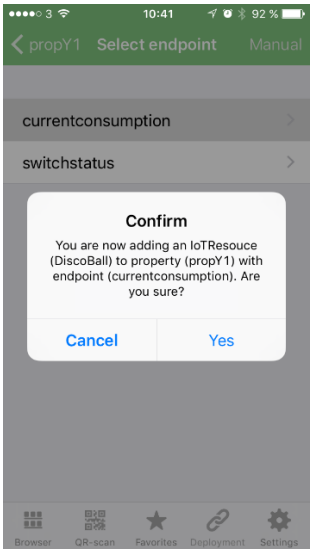

Figure 10: Deployment Tool Selecting IoT Resource and service for the property.

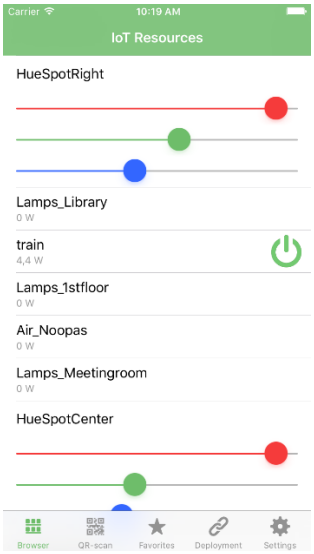

Figure 11: Deployment Tool IoT Resource Browser

4.2 IMPReSS Toolkit

The IMPReSS Toolkit is a tool that provides a single point for accessing the different tools in the IMPReSS system, see Figure 12 below. This tool makes it simpler to find the different resources needed when developing and deploying IMPReSS based systems.

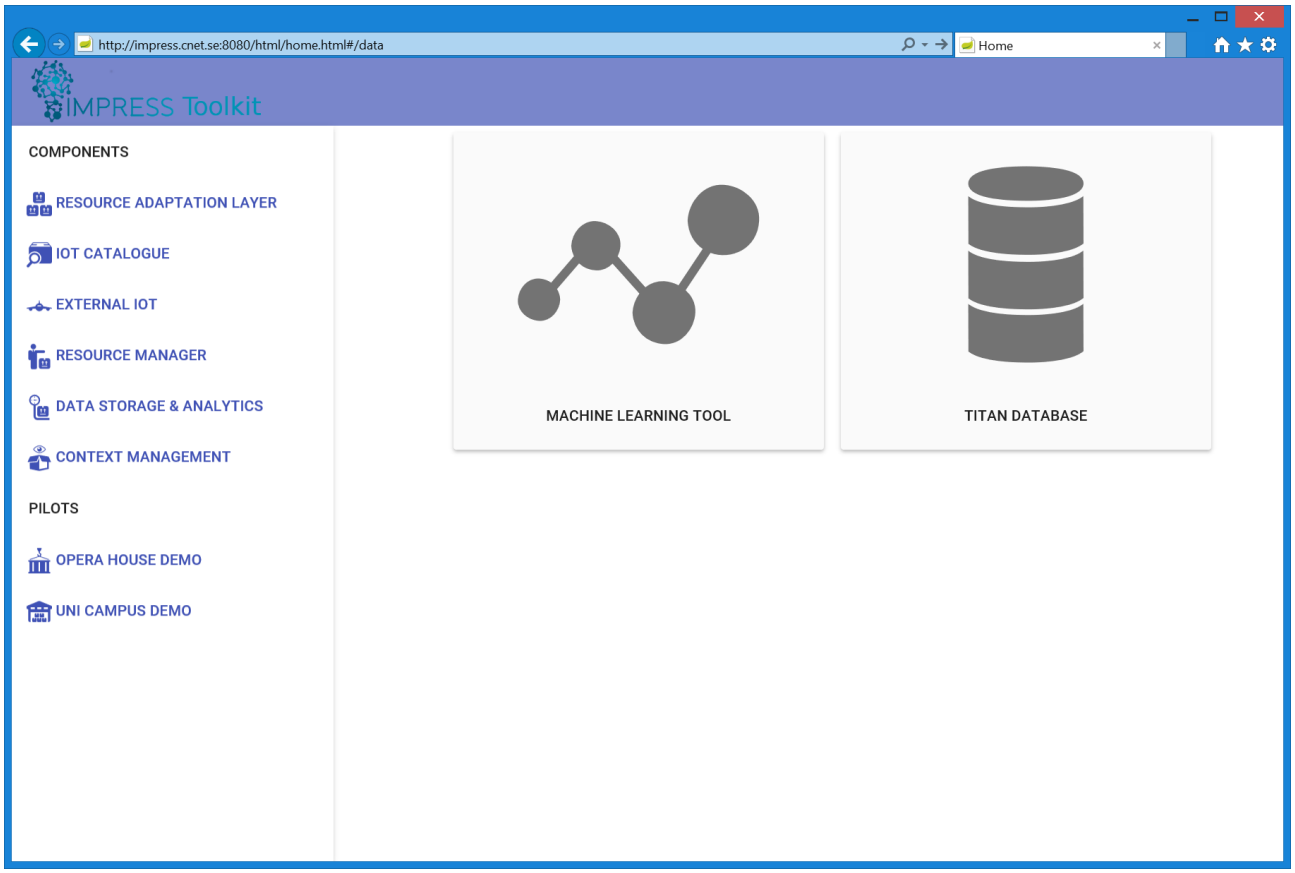

Figure 12: The IMPReSS Toolkit tool

5. Conclusions

As IMPReSS is a widely distributed both geographically and by time zone differences integration process has been quite challenging. But challenges also give the opportunity to gain experiences of what works well and what poses more problems.

What has worked well:

- MQTT, very stable platform and it works well even when the clients are distributed widely. Additionally, it gives the opportunity to use test messages to ease testing and development as well as intercepting/tracing calls.
- Distributed integration/test meetings using online conferencing systems also worked surprisingly well. The only challenge is the time zone differences which makes the possible time slots quite small.
- JSON has proved to be easy to use, even though it lacks a formal definition language. But it is humanly readable which aids in the debug situation.

What was more problematic:

- Actual sensors/actuators, the distributed nature of development meant that they needed to be deployed and maintained at one site. Also it is impossible to see if actuation works except when one is locally at the site of the sensors/actuators
- Keeping the whole distributed test environment alive at all times. Because of time zone differences it is not always easy to get hold of somebody to restart a failed server etc. This was partially relieved with the distributed integration/test meetings.

6. References

(MQTT, 2016) http://mqtt.org/ , visited 2016-02-15.

(IMPReSS_D3.1, 2014) D3.1 Resource Adaptation Interface Framework

(IMPReSS_D3.2, 2015) D3.2 Resource and Service Discovery Solutions

(IMPReSS_D3.5, 2016) D3.5 Templates and Integration Support Tool

(IMPReSS_D4.2, 2014) D4.2 Device and Subsystem Resource Management

(IMPReSS_D4.3, 2015) D4.3 Resource Management & Access Scheduler

(IMPReSS_D5.1, 2015) D5.1 Data and Knowledge Management Support

(IMPReSS_D5.2, 2014) D5.2 Data Analysis and Forecast for Energy Consumption

(IMPReSS_D5.3, 2015) D5.3 Data Mining and Machine Learning Tools

(IMPReSS_D5.4, 2015) D5.4 Machine Learning for User Behaviour and Occupancy Analysis

(IMPReSS_D6.4, 2015) D6.4 Implementation of Context Reasoning Engine

(IMPReSS_D6.5, 2015) D6.5 Implementation of Context Modelling Tool and Templates

(IMPReSS_D7.3.1, 2014) D7.3.1 Initial Design and Implementation of the Configuration and Composition Manager

(IMPReSS_D7.3.2, 2016) D7.3.2 Final Design and Implementation of the Configuration and Composition Manager