

**Private Public Partnership Project (PPP)**

Large-scale Integrated Project (IP)



### **D.7.3.3: FI-WARE Installation and Administration Guide**

**Project acronym:** FI-WARE

**Project full title:** Future Internet Core Platform

**Contract No.:** 285248

**Strategic Objective:** FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform

**Project Document Number:** ICT-2011-FI-285248-WP7-D.7.3.3

**Project Document Date:** 2014-07-10

**Deliverable Type and Security:** Public

**Author:** FI-WARE Consortium

**Contributors:** FI-WARE Consortium

## 1.1 Executive Summary

This document describes the installation and administration process of each Generic Enabler developed within in the "Interfaces to the Network and Devices" chapter. The system requirements for the installation of a Generic Enabler are outlined with respect to necessary hardware, operating system and software. Each GE has a section dedicated to the software installation and configuration process as well as a section, which describes sanity check procedures for the system administrator to verify that the GE installation was successful.

## 1.2 About This Document

The "FI-WARE Installation and Administration Guide" comes along with the software implementation of components, each release of the document referring to the corresponding software release (as per D.x.3), to facilitate the users/adopters in the installation (if any) and administration of components (including configuration, if any).

## 1.3 Intended Audience

The document targets system administrators as well as system operation teams of FI-WARE Generic Enablers from the FI-WARE project.

## 1.4 Chapter Context

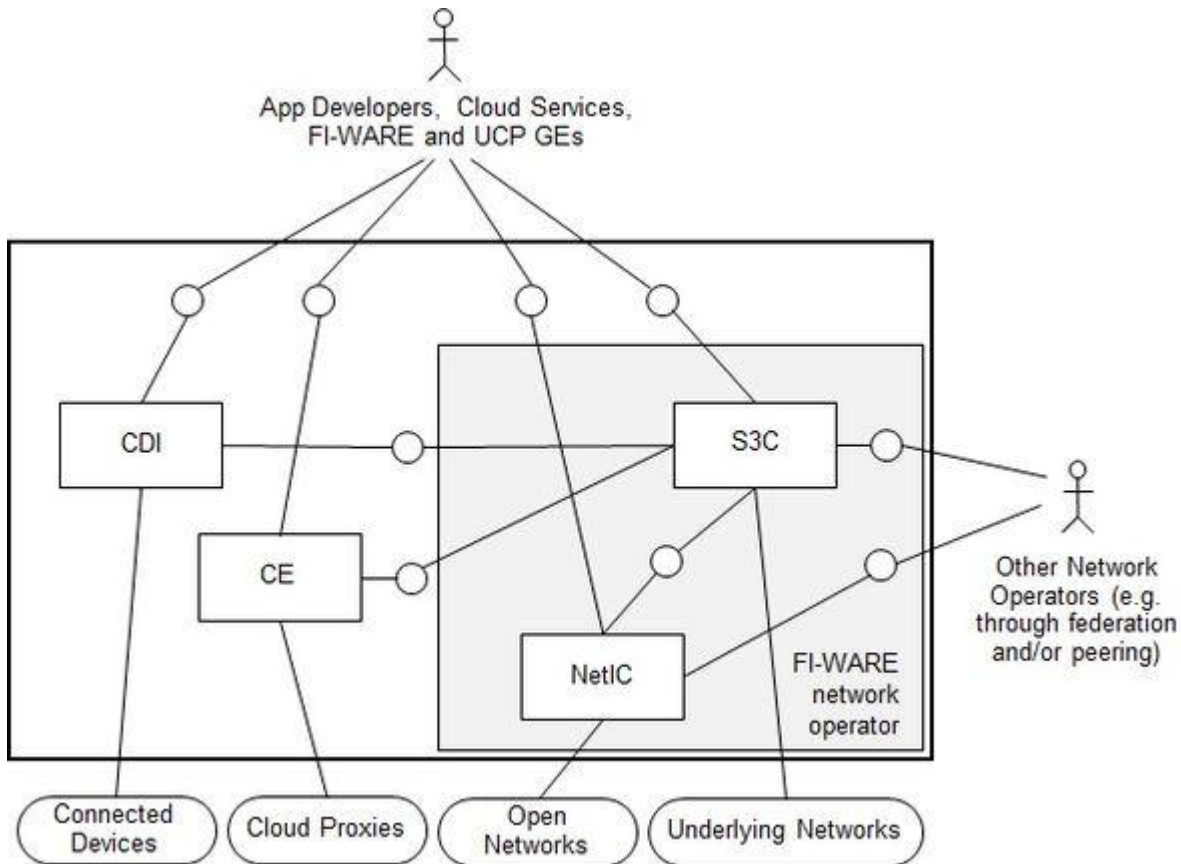
The I2ND chapter defines an enabler space for providing Generic Enablers (GEs) to run an open and standardised network infrastructure. The infrastructure deals with highly sophisticated terminals, as well as with highly sophisticated proxies, on one side, and with the network operator infrastructure on the other side. This latter will be implemented by physical network nodes, which typically will be under direct control of an operator, or the node functionality will be virtualised – in this case the I2ND functionality can be accessed by further potential providers, like virtual operators. The I2ND chapter defines four GEs, as represented in the figure:

- CDI (Connected Device Interface) towards the Connected Devices. These devices include, but are not limited to, mobile terminals, tablets, set top boxes and media phones.
- CE (Cloud Edge) towards the Cloud Proxies. Cloud Proxies are gateways, which will connect and control a set-up of nodes towards the Internet or/and an operator network.
- NetIC (Network Information and Control) towards Open Networks. Open Networks are following the idea of flow based controlled networks, and can be used for virtualisation of networks.
- S3C (Service Capability, Connectivity and Control) towards Underlying Networks. The underlying networks are following standards such as Next Generation Networks (NGNs) or Next Generation Mobile Networks (NGMNs). In the case of the S3C specified in I2ND the baseline underlying network will be the Evolved Packet Core (EPC) by 3GPP.

Each of the GEs of I2ND have specific interfaces which can be accessed by Application Developers, Cloud Services, FI-WARE and third party Enablers. Besides typical interfaces to functionality offered by such GEs, it is worth mentioning that:

- The GE S3C is the central point of the I2ND architecture. I2ND develops an enabling environment which can be used by network operators. Together with NetIC, both GEs build the environment of an operator, which might even be a virtual operator. S3C can be seen as the GE to run and steer the network infrastructure.

- The interfacing between S3C and CDI provides status and control information exchange of the device and remote control capabilities.
- Cloud proxies can be part of an operator infrastructure. Therefore it is necessary to have access to these network nodes through a standardised interface.



## 1.5 Structure of this Document

The document is generated out of a set of documents provided in `fiware` FI-WARE wiki. For the current version of the documents, please visit the wiki at <https://wiki.fi-ware.org/>.

The following resources were used to generate this document:

### D.7.3.3 Installation and Administration Guide front page

[Connected Device Interfacing - Installation and Administration Guide](#)

[Cloud Edge - Installation and Administration Guide](#)

[Cloud Edge OSGi - Installation and Administration Guide](#)

[NetIC GE - OFNIC - Installation and Administration Guide](#)

[NetIC GE - Altoclient - Installation and Administration Guide](#)

[NetIC GE - VNEIC - Installation and Administration Guide](#)

[S3C GE - EPC OTT API - Installation and Administration Guide](#)

[S3C GE - Network Identity Management - Installation and Administration Guide](#)

[S3C GE - Network Positioning Enabler - Installation and Administration Guide](#)

[S3C GE - Seamless Network Connectivity - Installation and Administration Guide](#)

[S3C GE - API Mediation - Installation and Administration Guide](#)

[S3C GE - Telecom AS - Installation and Administration Guide](#)

## 1.6 Typographical Conventions

Starting with October 2012 the FI-WARE project improved the quality and streamlined the submission process for deliverables, generated out of our wikis. The project is currently working on the migration of as many deliverables as possible towards the new system.

This document is rendered with semi-automatic scripts out of a MediaWiki system operated by the FI-WARE consortium.

### 1.6.1 Links within this document

The links within this document point towards the wiki where the content was rendered from. You can browse these links in order to find the "current" status of the particular content.

Due to technical reasons part of the links contained in the deliverables generated from wiki pages cannot be rendered to fully working links. This happens for instance when a wiki page references a section within the same wiki page (but there are other cases). In such scenarios we preserve a link for readability purposes but this points to an explanatory page, not the original target page.

In such cases where you find links that do not actually point to the original location, we encourage you to visit the source pages to get all the source information in its original form. Most of the links are however correct and this impacts a small fraction of those in our deliverables.

### 1.6.2 Figures

Figures are mainly inserted within the wiki as the following one:

```
[[Image:....|size|alignment|Caption]]
```

Only if the wiki-page uses this format, the related caption is applied on the printed document. As currently this format is not used consistently within the wiki, please understand that the rendered pages have different caption layouts and different caption formats in general. Due to technical reasons the caption can't be numbered automatically.

### 1.6.3 Sample software code

Sample API-calls may be inserted like the following one.

```
http://[SERVER_URL]?filter=name:Simth*&index=20&limit=10
```

## 1.7 Acknowledgements

The current document has been elaborated using a number of collaborative tools, with the participation of Working Package Leaders and Architects as well as those partners in their teams they have decided to involve.

## 1.8 Keyword list

FI-WARE, PPP, Architecture Board, Steering Board, Roadmap, Reference Architecture, Generic Enabler, Open Specifications, I2ND, Cloud, IoT, Data/Context Management, Applications/Services Ecosystem, Delivery Framework , Security, Developers Community and Tools , ICT, es. Internet, Latin American Platforms, Cloud Edge, Cloud Proxy.

## 1.9 Changes History

Release	Major changes description	Date	Editor
V1	First draft of deliverable submission generated	2014-07-09	TI (P. Garino)
V2	Final version for delivery	2014-07-11	TI (P. Garino)

## 1.10 Table of Contents

- 1.1 Executive Summary ..... 2
- 1.2 About This Document ..... 3
- 1.3 Intended Audience ..... 3
- 1.4 Chapter Context..... 3
- 1.5 Structure of this Document ..... 4
- 1.6 Typographical Conventions ..... 5
  - 1.6.1 Links within this document..... 5

- 1.6.2 Figures ..... 5
- 1.6.3 Sample software code ..... 6
- 1.7 Acknowledgements ..... 6
- 1.8 Keyword list ..... 6
- 1.9 Changes History ..... 6
- 1.10 Table of Contents ..... 6
- 2 Connected Device Interfacing - Installation and Administration Guide ..... 14
  - 2.1 Goal of the document..... 14
  - 2.2 Software and Hardware environment..... 14
    - 2.2.1 The CDI-Webinos platform ..... 14
    - 2.2.2 Distributed Compute Framework (Sisyphus) Setup ..... 24
    - 2.2.3 Mobility Management Setup..... 26
  - 2.3 Sanity check procedures..... 31
    - 2.3.1 End to End testing..... 31
    - 2.3.2 List of Running Processes ..... 33
    - 2.3.3 Network interfaces Up & Open ..... 34
    - 2.3.4 Databases ..... 35
  - 2.4 Diagnosis Procedures ..... 35
    - 2.4.1 Resource availability ..... 36
    - 2.4.2 Remote Service Access ..... 36
    - 2.4.3 Resource consumption ..... 36
    - 2.4.4 I/O flows ..... 38
- 3 Cloud Edge - Installation and Administration Guide ..... 39
  - 3.1 Installation ..... 39
    - 3.1.1 STEP1: Install the Cloud Proxy Software on the box ..... 40
    - 3.1.2 STEP2: Environment setup ..... 40
  - 3.2 Sanity check ..... 41
    - 3.2.1 End to end testing..... 41
    - 3.2.2 List of running process..... 42
    - 3.2.3 Network interfaces up & open ..... 42
    - 3.2.4 Databases ..... 42
  - 3.3 Diagnosis Procedures ..... 42
    - 3.3.1 Resource availability ..... 43
    - 3.3.2 Remote service access..... 43

- 3.3.3 Resources consumption ..... 44
- 3.3.4 I/O flows ..... 44
- 4 Cloud Edge OSGi - Installation and Administration Guide ..... 45
  - 4.1 Introduction ..... 45
  - 4.2 OSGi Equinox framework Installation..... 45
    - 4.2.2 Sanity check ..... 48
    - 4.2.3 Diagnosis Procedure ..... 49
  - 4.3 OSGi SDK Installation ..... 50
    - 4.3.1 Installation ..... 50
    - 4.3.2 Configuration of the PDE target platform ..... 50
    - 4.3.3 OSGi SDK –The mToolkit..... 52
    - 4.3.4 Sanity check..... 54
    - 4.3.5 Diagnosis Procedure ..... 55
    - 4.3.6 Activation of the IoT Gateway protocol Adapter - ZPA in the Cloud Edge OSGi Equinox framework ..... 56
- 5 NetIC GE - OFNIC - Installation and Administration Guide ..... 58
  - 5.1 Goal of the document..... 58
  - 5.2 OFNIC GEi..... 58
    - 5.2.1 Software and Hardware environment..... 58
    - 5.2.2 Prerequisites..... 59
    - 5.2.3 Getting source code..... 59
    - 5.2.4 Compilation ..... 60
    - 5.2.5 Configuration..... 60
    - 5.2.6 Running..... 60
  - 5.3 OFNIC GUI..... 61
    - 5.3.1 Software and Hardware environment..... 61
    - 5.3.2 Prerequisites..... 61
    - 5.3.3 Getting source code..... 61
    - 5.3.4 Running..... 62
  - 5.4 SNMP Sub-Agent ..... 62
    - 5.4.1 Software and Hardware environment..... 62
    - 5.4.2 Prerequisites..... 62
    - 5.4.3 Getting source code..... 62
    - 5.4.4 Compilation ..... 62



- 5.4.5 Configuration ..... 62
- 5.4.6 Running ..... 63
- 5.5 Sanity check procedures ..... 63
  - 5.5.1 End to End testing ..... 63
  - 5.5.2 List of Running Processes ..... 64
  - 5.5.3 Network interfaces Up & Open ..... 65
  - 5.5.4 Databases ..... 66
- 5.6 Diagnosis Procedures ..... 66
  - 5.6.1 Resource availability ..... 66
  - 5.6.2 Remote Service Access ..... 66
  - 5.6.3 Resource consumption ..... 66
  - 5.6.4 I/O flows ..... 67
- 5.7 References ..... 67
- 6 NetIC GE - Altoclient - Installation and Administration Guide ..... 68
  - 6.1 Introduction ..... 68
  - 6.2 Prerequisites for Software and Hardware ..... 68
  - 6.3 NetIC - Altoclient Generic Enabler Instantiation Implementation ..... 68
    - 6.3.1 Installing the *Altoclient* ..... 68
    - 6.3.2 Compiling and Linking ..... 69
    - 6.3.3 Test Programs ..... 69
  - 6.4 Sanity Check Procedures ..... 69
    - 6.4.1 End to End testing ..... 69
    - 6.4.2 List of Running Processes ..... 71
    - 6.4.3 Network interfaces Up & Open ..... 72
    - 6.4.4 Databases ..... 72
  - 6.5 Diagnosis Procedures ..... 72
    - 6.5.1 Resource availability ..... 72
    - 6.5.2 Remote Service Access ..... 72
    - 6.5.3 Resource consumption ..... 72
    - 6.5.4 I/O flows ..... 72
  - 6.6 References ..... 72
- 7 NetIC GE - VNP - Installation and Administration Guide ..... 73
- 8 NetIC GE - VNEIC - Installation and Administration Guide ..... 74
  - 8.1 Introduction ..... 74

- 8.2 VNEIC ..... 74
  - 8.2.1 Software and Hardware environment..... 74
  - 8.2.2 Prerequisites..... 75
  - 8.2.3 Getting source code..... 75
  - 8.2.4 Compilation ..... 75
  - 8.2.5 Configuration..... 75
  - 8.2.6 Running..... 75
- 8.3 Sanity check procedures..... 76
  - 8.3.1 End to End testing..... 76
  - 8.3.2 List of Running Processes ..... 76
  - 8.3.3 Network interfaces Up & Open ..... 76
  - 8.3.4 Databases ..... 76
- 8.4 Diagnosis Procedures ..... 76
  - 8.4.1 Resource availability..... 76
  - 8.4.2 Remote Service Access ..... 77
  - 8.4.3 Resource consumption..... 77
  - 8.4.4 I/O flows ..... 77
- 9 S3C GE - EPC OTT API - Installation and Administration Guide ..... 78
  - 9.1 EPC-OTT API Enabler Installation..... 78
    - 9.1.1 Step 1: Installation of ubuntu 12.04 ..... 78
    - 9.1.2 Step 2: Installation of S3C EPC-OTT API Enabler ..... 78
    - 9.1.3 Step 3: Configuration..... 79
    - 9.1.4 Step 4: Running the EPC-OTT-API Application..... 81
    - 9.1.5 Step 5: Test Client..... 81
  - 9.2 Sanity check procedures..... 82
    - 9.2.1 End to End testing..... 82
    - 9.2.2 List of Running Processes ..... 82
    - 9.2.3 Network interfaces Up & Open ..... 82
    - 9.2.4 Databases ..... 82
  - 9.3 Diagnosis Procedures ..... 83
    - 9.3.1 Resource availability..... 83
    - 9.3.2 Remote Service Access ..... 83
    - 9.3.3 Resource consumption..... 83
    - 9.3.4 I/O flows ..... 83

- 10 S3C GE - Network Identity Management - Installation and Administration Guide..... 84
  - 10.1 S3C GE - Network Identity Management Installation ..... 84
    - 10.1.1 Important Note..... 84
    - 10.1.2 Step 1: Installation of operating system..... 84
    - 10.1.3 Step 2: Installation of NIM..... 84
  - 10.2 Sanity check procedures..... 87
    - 10.2.1 End to End testing..... 87
    - 10.2.2 List of Running Processes ..... 87
    - 10.2.3 Network interfaces Up & Open ..... 87
    - 10.2.4 Databases ..... 87
  - 10.3 Diagnosis Procedures ..... 87
    - 10.3.1 Resource availability..... 87
    - 10.3.2 Remote Service Access ..... 88
    - 10.3.3 Resource consumption..... 88
    - 10.3.4 I/O flows ..... 88
- 11 S3C GE - Network Positioning Enabler - Installation and Administration Guide..... 89
  - 11.1 S3C GE - Positioning Enabler - Installation and Administration Guide..... 89
    - 11.1.1 Important Note..... 89
    - 11.1.2 Installation of Ubuntu 12.04..... 89
    - 11.1.3 Installation of the Positioning Enabler ..... 89
  - 11.2 Sanity check procedures..... 95
    - 11.2.1 End to End testing..... 95
    - 11.2.2 Databases ..... 97
    - 11.2.3 List of Running Processes ..... 98
    - 11.2.4 Network interfaces Up & Open ..... 98
  - 11.3 Diagnosis Procedures ..... 99
    - 11.3.1 Resource availability..... 99
    - 11.3.2 Remote Service Access ..... 99
    - 11.3.3 Resource consumption..... 99
    - 11.3.4 I/O flows ..... 99
- 12 S3C GE - Seamless Network Connectivity - Installation and Administration Guide..... 100
  - 12.1 Introduction..... 100
  - 12.2 Requirements ..... 100
  - 12.3 SCC (Seamless Connectivity Client) installation and configuration..... 101

12.3.1	Installation .....	101
12.4	SCS (Seamless Connectivity Server) installation and configuration .....	101
12.4.1	Installation .....	101
12.5	Sanity check procedures.....	102
12.5.1	End to End testing.....	102
12.5.2	List of Running Processes .....	103
12.5.3	Network interfaces up & open .....	104
12.5.4	Databases .....	105
12.6	Diagnostic Procedures .....	105
12.6.1	Resource availability.....	105
12.6.2	Remote Service Access .....	105
12.6.3	Resource consumption.....	105
12.6.4	I/O flows .....	105
13	S3C GE - API Mediation - Installation and Administration Guide .....	106
13.1	API Mediation installation .....	106
13.1.1	List of components .....	106
13.1.2	Prerequisites.....	106
13.1.3	API mediation and admin GUI installation .....	107
13.2	Sanity check procedures.....	108
13.2.1	End to End testing.....	108
13.2.2	List of Running Processes .....	109
13.2.3	Network interfaces Up & Open .....	109
13.2.4	Databases .....	110
13.3	Diagnosis Procedures .....	110
13.3.1	Resource availability.....	110
13.3.2	Remote Service Access .....	110
13.3.3	Resource consumption.....	110
13.3.4	I/O flows .....	110
14	S3C GE - Telecom AS - Installation and Administration Guide .....	112
14.1	Telecom AS Installation .....	112
14.1.1	Step 1: Installation of ubuntu 12.04 .....	112
14.1.2	Step 2: Installation of mobicents.....	112
14.1.3	Step 3: Get and uncompress the OneAPIVoiceCallControlEnabler .....	112
14.1.4	Step 4 : Configure the files .....	112

- 14.1.5 Step 5: Launch JBoss..... 113
- 14.2 Sanity check procedures..... 113
  - 14.2.1 End to End testing..... 113
  - 14.2.2 List of Running Processes ..... 114
  - 14.2.3 Network interfaces Up & Open ..... 115
  - 14.2.4 Databases ..... 115
- 14.3 Diagnosis Procedures ..... 115
  - 14.3.1 Resource availability..... 115
  - 14.3.2 Remote Service Access ..... 115
  - 14.3.3 Resource consumption..... 115
  - 14.3.4 I/O flows ..... 115

## 2 Connected Device Interfacing - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 2.1 Goal of the document

Goal of this document is to provide a useful guide for the administrators of the Connected Device Interfacing (CDI) GE implementation, in the following called A-CDI. The functionality and interfaces of the CDI GE are described in the [CDI Open Specifications](#) document. The document starts describing basic software and hardware required to support A-CDI on top of a programmable device, such as a smartphone. The Webinos system will be introduced, being the base platform enabling CDI operations since the R2 release. Document follows with specific technical information that might help GEi administrators: running processes, diagnosis tests, network flows etc.

A-CDI provides a set of APIs by means of two independent subsystems: the “CDI-Webinos” platform and “Distributed Compute” framework. CDI-Webinos is specifically targeted for Android (in a client-only architecture), while the “Distributed Compute”, also called "Sisyphus", is based on javascripts and can be run in different systems, in client and server architectures.

### 2.2 Software and Hardware environment

A-CDI offers a collection of APIs, most of them being implemented on top of a CDI-Webinos platform, are available for Android devices. A new API, the Distributed Compute, has been introduced in release 3.2 of A-CDI, to enable easy communication between client and cloud application components. To provide this functionality, the Distributed Compute framework, also called Sisyphus for brevity, includes an element which should be run in a server. This document describe both the CDI-Webinos platform, running as a client in any Android environment (minimum Android release 3.x is required, while Android 4.x is suggested), and the Distribute Compute framework(alias Sisyphus), able to run in clients and servers with different operating systems (e.g. Android and Linux). Concerning the Distribute Compute framework, we will focus especially on the server requirements, because the client side only need standard javascript execution (available in any internet browser, already installed in any programmable device which have connectivity), while the server side may require further libraries to enable full javascript support.

#### 2.2.1 The CDI-Webinos platform

Webinos is a web based application platform that runs on top of multiple devices (smartphones, laptops, tablets, smartTV, invehicle infotainment etc.). Applications hosted within this platform are implemented using web standards (HTML5, CSS3, and JavaScript). Webinos creates an overlay network of Web applications based on APIs offered by local and remote native platforms. Local and remote API invocation is supported by means of JSON-RPC 2.0 calls. Webinos is based on the following concepts and architectural components:

- *Personal Zone (PZ)*: defines the set of devices and services owned by a particular user. The PZ appears to each Web application as a set of local APIs. The API set is built through service discovery routines.
- *Personal Zone Proxy (PZP)*: Every device runs a PZP, which exposes local APIs to local and remote web applications, by means of Remote Procedure Call (RPC) mechanisms. The PZP routes RPC calls generated by local applications in different ways:
  - To the native platform, when a local API is invoked by the web application.
  - To a peer PZP (direct communication) when the invoked API is available on a different device in the same PZ.
  - To the Personal Zone Hub (PZH) when the invoked API is available on a different device in a different PZ.

The PZP is able to relay JavaScript calls, generated by the web application, to the native device platform (both local and remote). PZP is mostly based on the NodeJS technology, and it is implemented as a NodeJS module, as well as its main structural components (RPC-Handler, Message-Handler etc.).

- *Web Runtime (WRT)*: This is the environment that hosts and executes the Web application. A WRT is provided by the web browser that hosts the Webapp. However a WRT is also provided by Webinos, so that Webapps can be directly deployed as Webinos Widgets. The final choice on the WRT to be used is up to the application developer.
- *Personal Zone Hub (PZH)*: Each PZ has one PZH that acts as a central node for various tasks: Policy Enforcement Point (PEP), certificate handler, federation of PZ's devices, shared context synchronization, mediation between local and external PZs etc. The PZH can also be placed in the cloud.

CDI-Webinos is a customization of the Webinos platform (mainly composed of a customized PZP and a WRT, together with APIs), to provide additional A-CDI functionalities, such as QoE, QoS and Mobility Management, specifically targeted to the Android environment.

#### 2.2.1.1 **CDI-Webinos installation**

CDI-Webinos is distributed as an Android Application Package file (APK), installable on an Android device (physical or emulated). The requirements for the installation are:

- A personal computer with any Windows or a Linux operative system, to download the binary packages and run the install scripts.
- An Android device, such as a smartphone or a tablet, or an Android emulator (available by installing the [Android SDK](#)), to install the A-CDI-Webinos Android build.

The installation packages of the CDI-Webinos platform are distributed in two separate parts; please consider both parts are strictly required to be able to use all the APIs of A-CDI GEi. The packages required can be downloaded from the following two repositories:

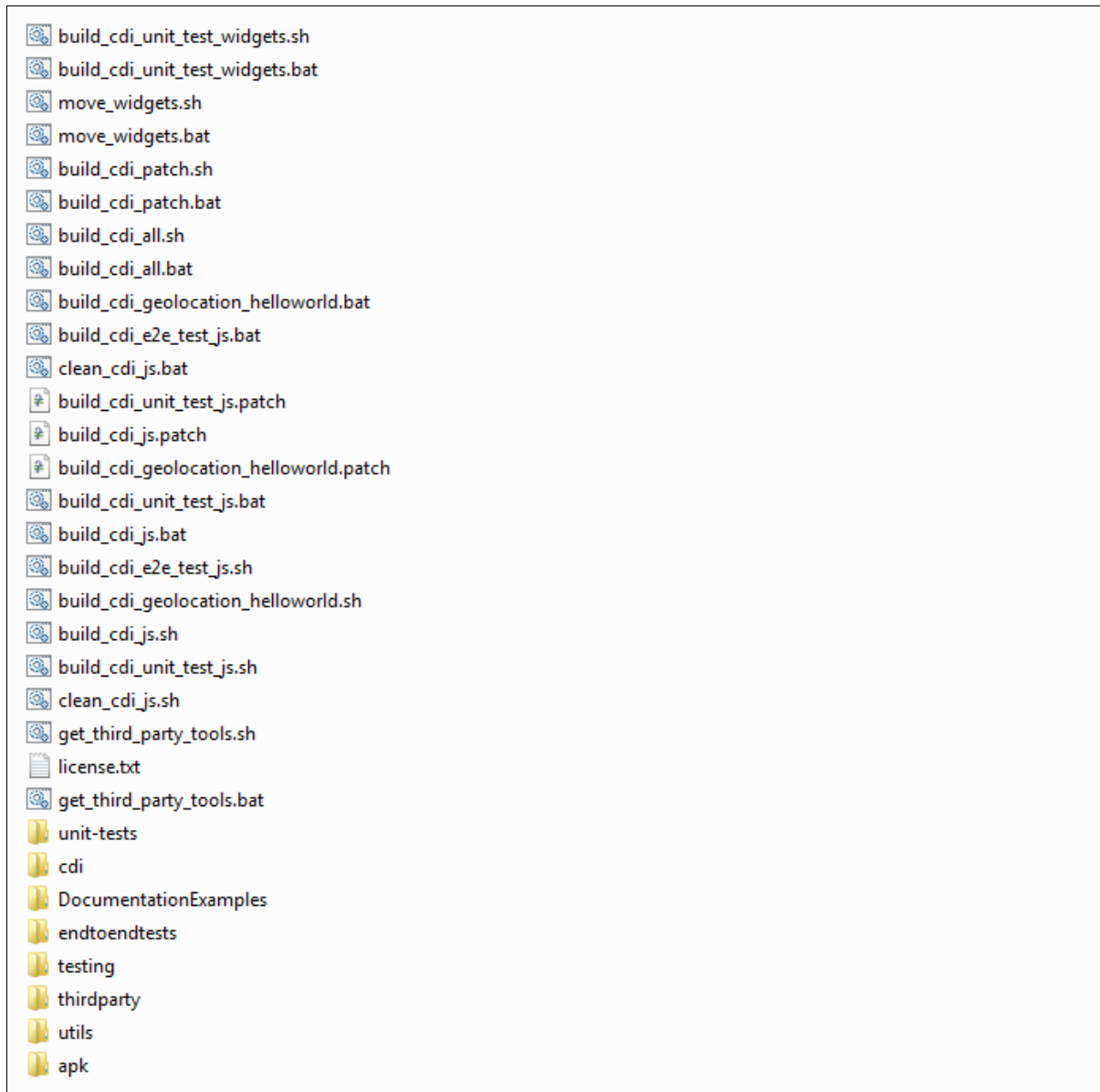
- [A-CDI Javascript APIs, Tests and Scripts \(A-CDI 1st part\)](#)
- [CDI-Webinos with QoE, Tests and Scripts \(A-CDI 2nd part\)](#)

To download the first part of A-CDI, there are two possible choices:

- By using a browser, open the [A-CDI Javascript APIs, Tests and Scripts \(A-CDI 1st part\)](#) web page and click on the *Browse Code Git repository* button, then click on the *Download Snapshot* folder icon in the right side of the top grey bar. Extract the ZIP file to any folder in your computer. Important: please consider that the ZIP file also contain a top level folder with the same name of the file; this top-level folder must be ignored in the next operations.
- Download a [Git Client](#), install it and run the command: `git clone git://git.code.sf.net/p/connecteddeviceinterface/code a-cdi`. Important: please consider that the `git clone` command also builds a top-level folder named `a-cdi`; this top-level folder must be ignored in the next operation.

Once you've downloaded and extracted the first part of A-CDI, you can download the ZIP file with the most recent [CDI-Webinos with QoE, Tests and Scripts \(A-CDI 2nd part\)](#). The next operation is the extraction of the ZIP file to the same folder used to extract the first part. Please, notice that you should consider for the extraction the previous inner folder (inside the top level folder, i.e. inside the *connecteddeviceinterface-code-xyz* or the *a-cdi* folder, depending on the extraction method previously used). You can verify that the extraction folder is correct, by checking that Warning messages appear asking you to merge the contents of some folders, while no file replacement should occur. If all operations have been completed successfully, you should find the following files and folders in the extraction folder:





### CDI folders

When you have obtained the above folder structure, you can run the scripts to build the test code and widgets. Depending on your operating system, you have to launch the following scripts:

- If you use a Linux OS: `source ./build_cdi_all.sh`
- If you use a Windows OS: `build_cdi_all.bat` (to build the end2end tests, the unit tests and few examples). You can run the scripts by clicking on them, or by opening a Command Prompt window and issuing all the commands there.

These scripts build the end2end tests, the unit tests and a few examples. All of these tests and examples will be available in two forms:

- as standalone HTML pages, that can be opened by any recent HTML5 compliant browser in the device, such as firefox, opera, chrome (e.g Firefox 26, Opera 19, Chrome 32);
- as widgets (files with extension WGT), that can be directly opened by the cdi-webinos application.

After the scripts complete successfully, you need to copy all the contents to the memory of an Android smartphone: connect the device to your computer, create a folder in the device storage memory and finally copy everything from the installation directory in your computer to the folder created in the device memory (to do this, you have to connect the device to your computer, then the internal memory will be visible from file explorer as a disk unit). The copied content contains the HTML pages to run all tests, examples and widgets (files with extension `.wgt`). To execute all the test as widgets, you only need to copy the *widgets* folder in the storage memory of the Android device.

The binary installable package for Android is contained in the *apk* folder. Before proceeding, please check that in your setting you have enabled *Unknown Application Sources* in the Security settings, and you have enabled the *Debug USB* from the Developer option settings of your Android device. The provided binary distribution package (apk file) supports Android versions ranging from 3 (API level 11) to 4.x.

You can install the package by copying it to the memory (SD card) of an Android smartphone and then opening it with any File Browser on your device. Alternatively, you can use facilities provided by the *adb* tool (coming with the Android SDK package) by running the following command in a Command Prompt in Windows (*cmd*), or a bash shell in linux:

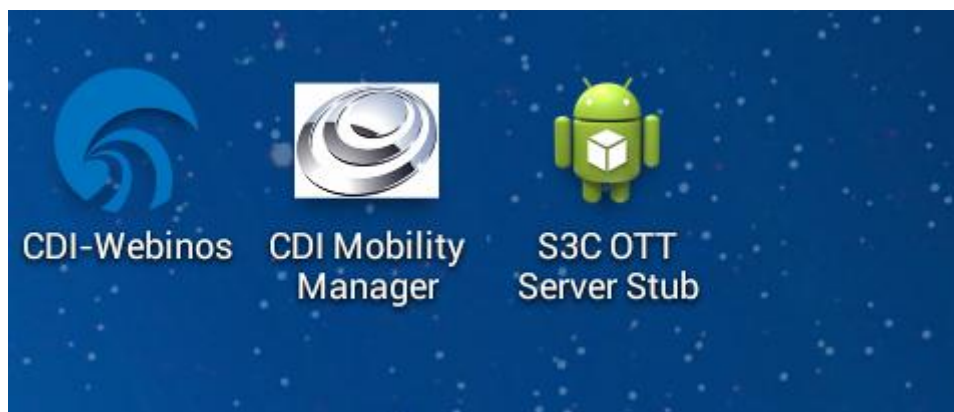
```
adb install cdi-webinos-android.apk
```

In the *apk* folder you'll also find other two APK, required to run Mobility Manager and some tests. It's also required to install them. They are:

```
MobilityManagerService.apk
```

```
AndroidQosRestServerStub.apk
```

The first APK installs a *CDI Mobility Manager* application, used to configure the communication between the Mobility manager APIs and the network. Description on how to use and configure this application is in the section [Mobility Management Setup](#). The *AndroidQosRestServerStub.apk* install an application called *S3C OTT Server Stub*, used to provide a stub service to run Unit Tests without the need to connect to the network. Please, refer to [Connected Device Interfacing - Unit Testing Plan](#) for usage and configuration of the *S3C OTT Server Stub* application. After having installed all the three APKs, three applications will be available as shown in the picture.



CDI apps

The CDI APIs have been developed as extensions of the Webinos platform, consequently they need to include a webinos definitions file (webinos.js), included in the binary distribution. To use this definitions file, a single instruction must be added at the beginning of the HTML code of the user:

```
<script type="text/javascript"  
src="http://localhost:8080/webinos.js"></script>
```

The above instruction enables the usage of the Webinos platform inside any webapp (HTML and Javascript code running in any HTML5-compliant browser). Applications that use Webinos (and CDI functions) may also be executed as Widgets. A Widget is a compressed archive containing the HTML and Javascript code, distributed as a single file, with extension .WGT (the widget is a ZIP archive renamed for convention with a .WGT extension). To enable the usage of the Webinos platform also for the Widgets, a different definition file must be included. That file is also contained inside the A-CDI distribution; to include the webinos definitions also for the widget, the following lines must be included at the beginning of the HTML code contained in the Widget or Webpage:

```
<script type="text/javascript" src="/static/webinos.js"></script>  
  
<script type="text/javascript"  
src="http://localhost:8080/webinos.js"></script>
```

The first line allows to run the code as a widget from the A-CDI Webinos application; the second line is the one needed to run the code as standalone HTML/JS page in any browser, as previously explained. Both lines can be added simultaneously in the code, paying attention to respect exactly the same order shown above (static webinos for widget mode is the first, followed by localhost webinos for the webpage mode).

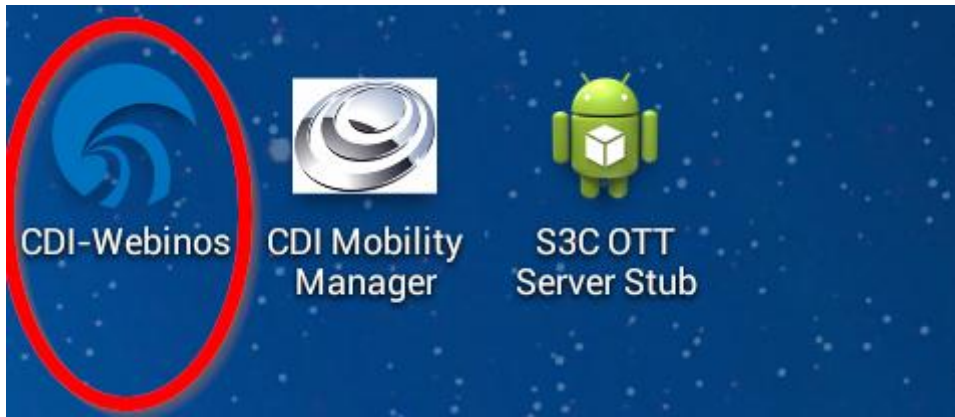
All CDI APIs require a unique front-end file. The idea is to include all the APIs description in a single JavaScript file, to let developers access any of the CDI functionalities (QoE, Sensors, Personal Data Services etc.). The CDI frontend file to include in any HTML code that uses the CDI APIs is called `cdi.js`. Specifically, the following line must be added to HTML code (for both Webapps or Widgets) after the the inclusion of `webinos.js` (explained before) to enable the usage of CDI functions inside an HTML page:

```
<script type="text/javascript" charset="utf-8" src="cdi.js"></script>
```

The `cdi.js` file is automatically built by the installation scripts. Copies of this file can be found in the installation folder (the same folder where are the installation scripts), and in all subfolders containing the tests and the examples.

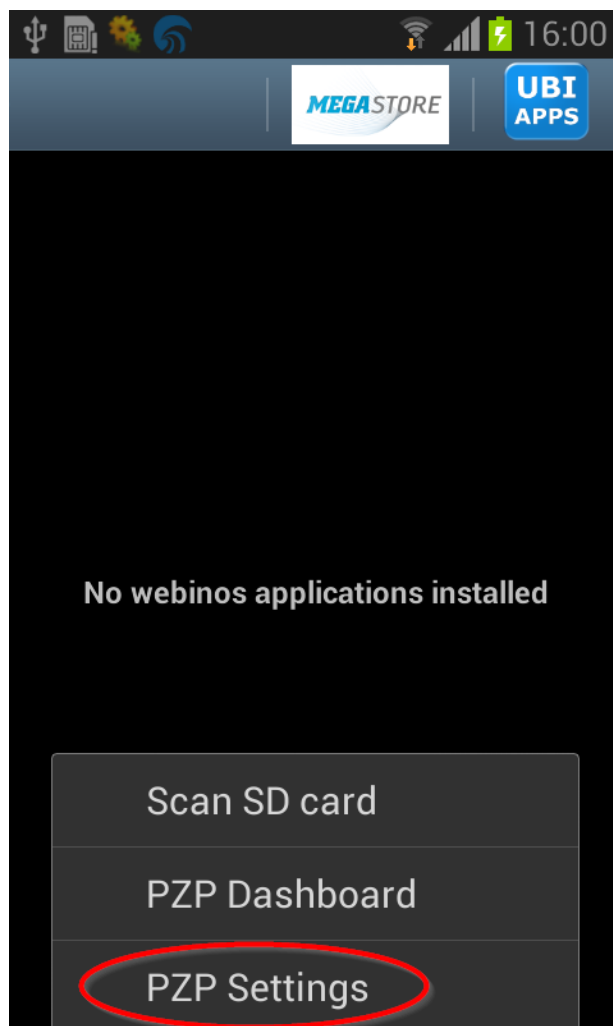
### 2.2.1.2 *CDI-Webinos startup*

Open the *CDI-Webinos* application icon that you can see in the picture below. The application allows to search and install widgets that use the API of the A-CDI GEi.



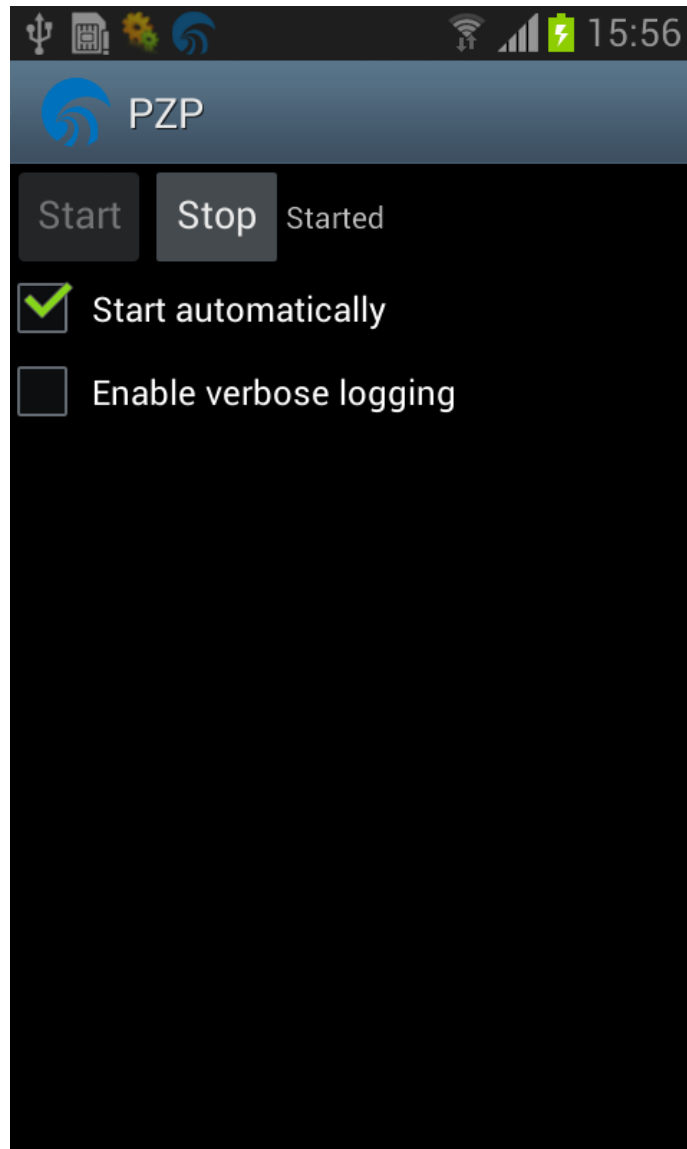
**CDI-Webinos app**

At the first start, a progress bar will appear; wait until the operations complete and the bar disappears. You should now have a black screen with a centered sentence saying *No webinos applications installed*. The CDI-Webinos application automatically initializes and starts the PZP activity and sets up the Widget Runtime (WRT). To check that the PZP activity is started and is correctly running, select *PZP Setting* from the context menu inside the CDI-Webinos Application.



**Settings**

A new PZP screen must appear, showing a *Start* and a *Stop* button, with the former grayed (active), and a text message saying *Started*.



**PZP Started**

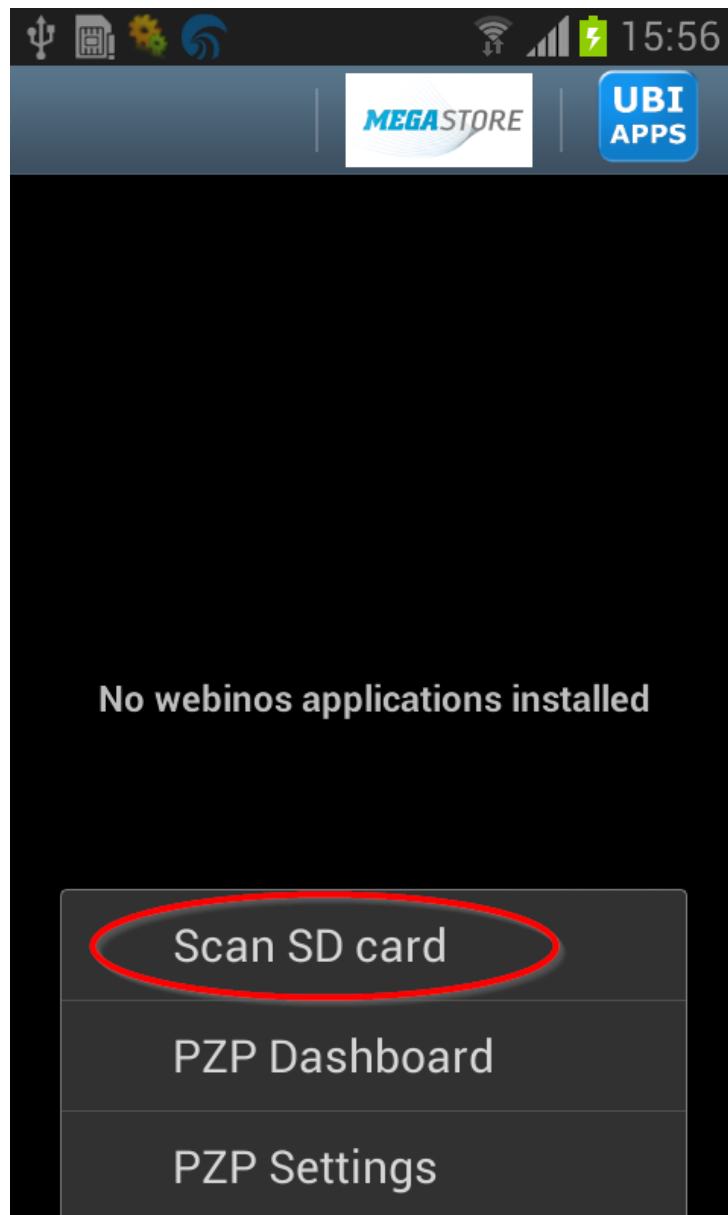
In the current A-CDI release the QoE functionality is embedded in PZP (since it requires also native implementation). In order to check if the native part of the QoE Handler [\[1\]](#) module has been correctly loaded inside the PZP, please check the logcat (running a command `adb logcat`) and look for a message like:

```
QoE-API-Handler: Loading QoE API Handler...  
node.js: QoE API Handler loaded
```

**Note:** The logcat utility can be downloaded with the Android SDK and provides tons of options and output filtering rules. Tutorials on the usage of the logcat tool are out of the scope of this document. Useful information can be found at [logcat man page](#).

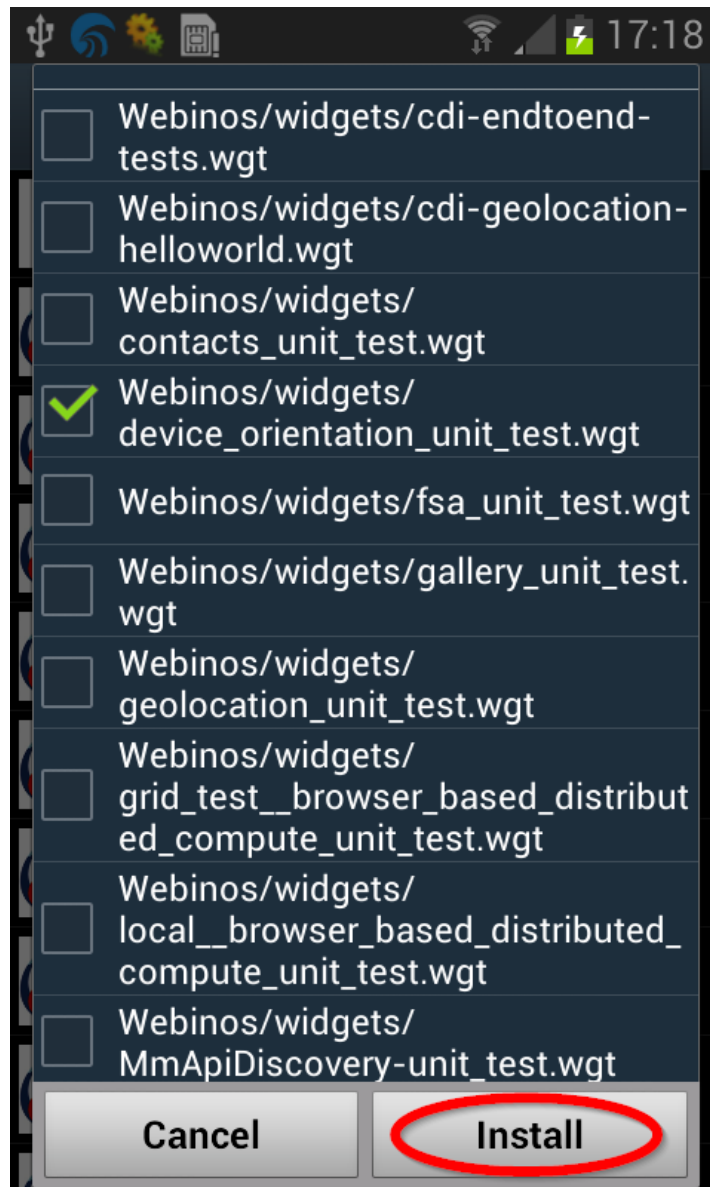
The A-CDI is now ready and it is possible to run the tests and the examples installed. To run them as webpages, open the `index.html` file (found in the `endtoend` or in the `unit-tests` folders) using a mobile

web-browser on the device (for instance, any recent version of Firefox, Opera, Chrome or any HTML5-compliant browser). To run test widgets, select the menu Scan SDCard after running the CDI-Webinos application in the Android device.



**Scan SDCard**

A popup window will appear showing all available widgets found; to install them, click on the checkboxes at the left side of their name and then on the *Install* button.



**Install CDI Widgets**

After this operation, the widgets will appear inside the main screen of the CDI-webinos application and can be launched by clicking on their name.



CDI Tests Widgets

## 2.2.2 Distributed Compute Framework (Sisyphus) Setup

The following provides instructions on how to install the required software environment for Distributed Compute framework, also called Sisyphus for brevity. These instructions should be run on an Ubuntu 12.04 LTS (32bit) based system. These steps are explained in detail below, and a set of shell scripts are also provided which help to automate the installation.

### 2.2.2.1 *Get Sisyphus on the Server*

The exact same code base which runs on a CDI based client device, runs on the server as a server node. The code for the server can be obtained by simply obtaining a copy of CDI from source forge using the following command:



```
git clone git://git.code.sf.net/p/connecteddeviceinterface/code
connecteddeviceinterface-code
```

### 2.2.2.2 *Installing Dependencies*

Sisyphus requires Node.js and Socket.io, Socket.io-client, Express and 'Q'. It is possible to install these by hand, and instructions are given below. Alternatively, the git repository contains a simple script file which will do all the hard work for you: `cdi/sis/install_sisyphys_server.sh`

#### 2.2.2.2.1 *Manual Install of node.js and dependencies*

```
sudo apt-get update
sudo apt-get install python-software-properties python g++ make git
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install nodejs
```

[From github.com...Installing-Node](#)

#### 2.2.2.2.2 *Manual Install express:*

```
sudo npm install express
```

#### 2.2.2.2.3 *Manual Install socket.io:*

```
sudo npm install socket.io
sudo npm install socket.io-client
```

#### 2.2.2.2.4 *Manual Install 'Q' for node.js*

```
sudo npm install q
```

### 2.2.2.3 *Install thirdparty software, jQuery and Q for clients*

Sisyphys relies on some third party software to be present for some of the test environments. The shell script `cdi/sis/get_thirdparty_tools.sh` will download the third party tools and setup the environment for you. Alternatively you can manually download the third party tools using the steps below:

#### 2.2.2.3.1 *Manual Install of Q, and jQuery*

The test application (only) requires jQuery. Specifically it requires jQuery, and the client side (browser) version of 'Q'. Both of these can be downloaded via `wget`:

```
wget https://s3-us-west-1.amazonaws.com/q-releases/q.min.js
wget http://code.jquery.com/jquery-1.10.2.min.js
```

Some of the unit and functional test supplied with CDI rely on these files being, renamed and placed in a `thirdparty` directory. The following diagram illustrates where the `thirdparty` directory should be located:

```

.
├── connecteddeviceinterface-code
│   ├── cdi
│   │   ├── pds
│   │   ├── sensors
│   │   └── sis
│   ├── DocumentationExamples
│   │   └── cdi-geolocation-helloworld
│   ├── endtoendtests
│   ├── testing
│   │   └── sisyphus_testing
│   └── unit-tests
└── thirdparty

```

The files should be renamed as follows: `* q.min.js` renamed to `q.js` `* jquery-1.10.2.min.js` renamed to `jquery.js`

#### 2.2.2.4 **Development Installation**

The shell script `cdi/sis/install_additional_development_tools.sh` will install some tools which are useful for developers working on Sisyphus, or creating applications which use Sisyphus. You can install the same packages manually using the following command:

```
sudo apt-get install chromium-browser meld vim-gtk vim kate
```

### 2.2.3 Mobility Management Setup

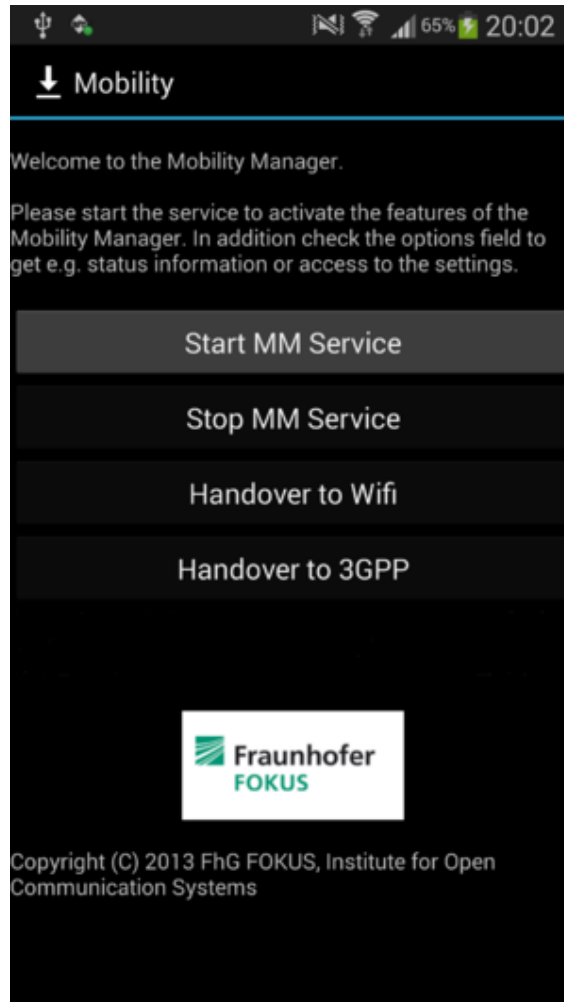
In order to use the Mobility Management API, an external Android application has to be installed. This application provides a service running in the background and a UI to configure the Mobility management component.

The Android application is provided as `MobilityManagerService.apk` file and is located in the `apk` folder of the A-CDI downloaded code. To install it on a device, copy the file in the device and open it with a file browser; if you have installed the Android SDK, you can directly run the following command in the `apk` subfolder from the folder where A-CDI has been placed in your computer:

```
adb install MobilityManagerService.apk
```

A new application named "Mobility" will appear.

### 2.2.3.1 Start Screen



The start screen of the Android application is mainly used to start and stop the MM. In addition, it can be used to trigger handover manually. The start screen contains four buttons:

- **Start MM Service**

Starts the MM as a background service. If the "Connect to ANDSF to retrieve policy information"-option in the settings is checked, the MM will try to contact the ANDSF component of the S3C GE (over the [S14 interface](#)) and fetch a policy and the ANDSF is able to push policies to the device.

- **Stop MM Service**

Stops the MM as a background service. This will also stop the connection to the ANDSF (if configured in the settings).

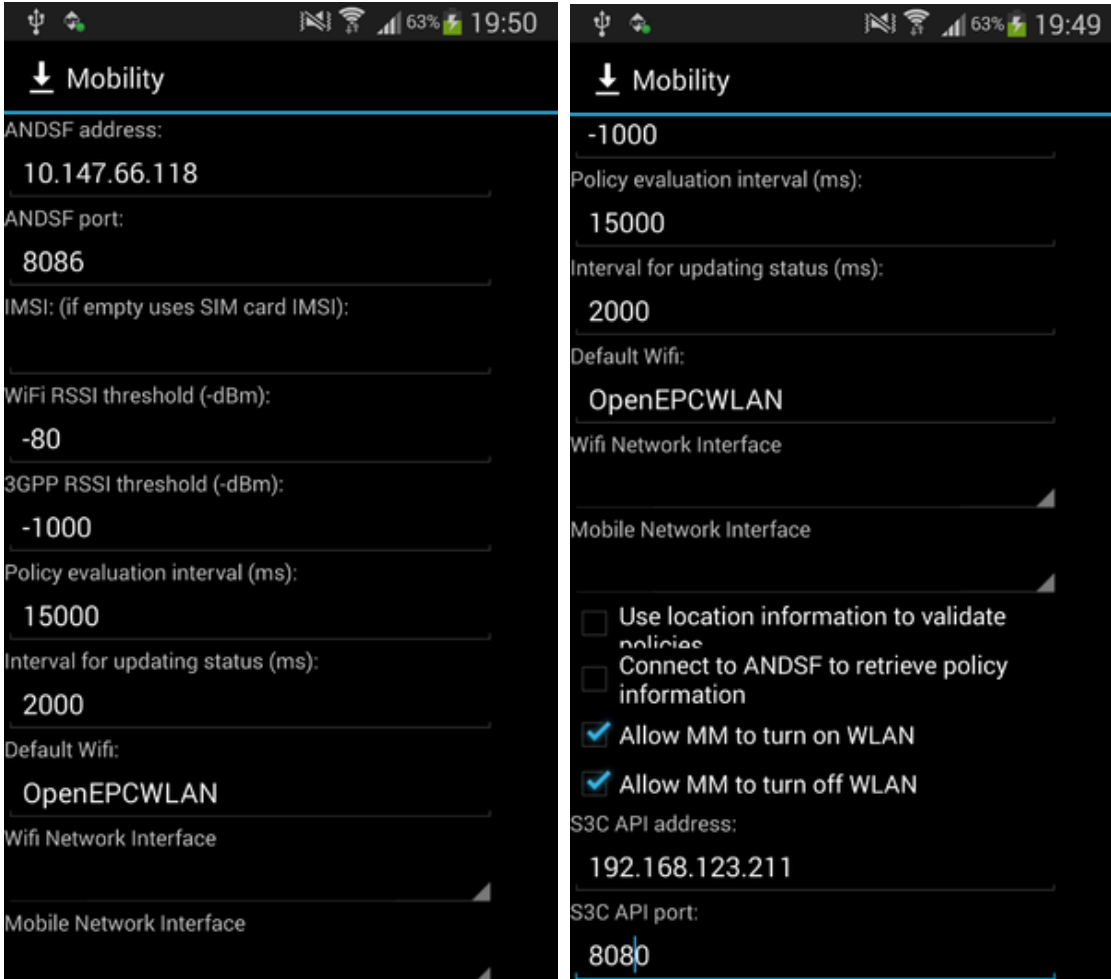
- **Handover to Wifi**

Triggers a handover to the Wifi network configured in the settings as "Default Wifi". This is enforced locally on the device.

- **Handover to 3GPP**

Triggers a handover from the Wifi network to the mobile network. This is enforced locally on the device.

### 2.2.3.2 Configuration



The configuration screen of the MM can be accessed by selecting the "settings" option of the start screen menu. The settings will be saved when leaving the settings screen by using the "back" button of the phone. The settings include the following options:

- **ANDSF address**

The IP address of the ANDSF that the client will use to fetch the policies over the [S3C Interface to CDI](#). If this value is not set, the Mobility Manager will not retrieve policies on access network selection.

- **ANDSF port**

The TCP port that the ANDSF is listening for HTTP POST requests on the [FIWARE.ArchitectureDescription.I2ND.S3C#Main\_interactions|S3C Interface to CDI]]. If this value is not set, the Mobility Manager will not retrieve policies on access network selection.

- **IMSI**

The IMSI is used to identify the MM to the ANDSF. It is stored on the SIM card of the phone. This field can be used to use a different IMSI than the SIM card of the phone contains or if no SIM card is used at all. If the field is left blank and a SIM card is used, the MM will insert here automatically the IMSI of the SIM card.

- **WiFi RSSI threshold**

This option configures the threshold to trigger handover based on the signal strength (RSSI) of the Wifi network. If the signal strength is less than the configured threshold, the MM will trigger a handover to the mobile network. Currently the interval in which the MM checks the signal strength of the current network is the same as the one the MM evaluates the received policies (this can be configured using the "Policy evaluation interval" option). The unit of the signal strength is measured in dBm. If there is a policy that tells the MM to use a specific Wifi network and the signal strength is lower than this threshold, the MM will not connect to this Wifi network. If the MM did a handover due to bad signal strength, it will check the signal strength of the former Wifi network continuously (in the same interval that is used to check the policies) and will reconnect as soon as the signal strength is higher than the threshold. This is enforced locally on the device without interaction to external components in the network.

- **3GPP RSSI threshold**

This option configures the threshold to trigger handover based on the signal strength (RSSI) of the mobile network. If the signal strength is less than the configured threshold, the MM will trigger a handover to the Wifi network configured in the "Default Wifi" option. Currently the interval in which the MM checks the signal strength of the current network is the same as the MM evaluates the received policies (this can be configured using the "Policy evaluation interval" option). The unit of the signal strength is measured in dBm. If there is a policy that tells the MM to use the mobile network and the signal strength is lower than this threshold, the MM will not connect to the mobile network. If the MM did a handover due to bad signal strength, it will check the signal strength continuously (in the same interval that is used to check the policies) and will reconnect as soon as the signal strength is higher than the threshold. This is enforced locally on the device without interaction to external components in the network.

- **Policy evaluation interval**

This parameter adjusts the time interval in which policies are being evaluated and also in which the signal strength of the current connection is checked. Once the policies are received, the evaluation happens locally on the device.

- **Interval for updating status**

This parameter is used by the status monitor screen. This parameter configures how often the values of the status monitor are refreshed.

- **Default Wifi**

This parameter names the SSID of the default Wifi. The MM will connect to the default Wifi in the following cases:

- 1) If a handover was triggered manually (with the buttons on the start screen or the status monitor)
  - 2) If the signal strength of the mobile network is below the threshold and there is no policy to be evaluated
- This parameter is only used if a handover to a wifi is enforced locally on the device. It is not used when policies have been received.

- **Wifi Network Interface**

This parameter is used to select the device's network interface for the Wifi access network. This parameter is used to set the routing tables during handover and to monitor the connection to the Wifi network.

- **Default Wifi Gateway**

Deprecated

- **Mobile Network Interface**

This parameter is used to select the device's network interface for the mobile access network. This parameter is used to set the routing tables during handover.

- **Default Mobile Gateway**

Deprecated

- **Use location information to validate policies**

This check box enables or disables the location awareness of the MM's policy engine: a policy may be only valid for a specific location. This option may be used to ignore such location conditions. The default value is disabled.

- **Connect to ANDSF to retrieve policy information**

This option enables or disables the connection to the ANDSF over the [S3C Interface to CDI](#) of the S3C GE. If this feature is enabled, then the MM will try to connect to the ANDSF when starting the MM service to get policies and the MM will open a socket to listen for further connection attempts from the ANDSF to push policies onto the device. By default this feature is enabled.

- **Allow MM to turn on WLAN**

If this checkbox is checked then the MM will automatically turn on the Wifi interface of the device during a handover to a Wifi network. The default setting is unchecked.

- **Allow MM to turn off WLAN**

If this checkbox is checked then the MM will automatically turn off the Wifi interface of the device during the handover to the mobile network. The default setting is unchecked.

- **S3C API address**

The IP address of the S3C EPC OTT API component of the S3C API is running. This is the IP address part of the [EPC-CDI API](#) interface. If this value is not set, the Mobility Manager is not able to request QoS.

- **S3C API Port**

The port number where the S3C EPC API GE is listening for REST requests. This is the IP port part of the [EPC-CDI API](#) interface. If this value is not set, the Mobility Manager is not able to request QoS.

### 2.2.3.3 **Limitations when not connecting to an S3C GE**

There are two different levels of interconnection with the S3C GE.

First level is the connection to S3C APIs. If the connection to the S3C API is not configured, then the CDI applications are not able to send QoS requests. If the connection to the ANDSF component of the S3C GE is not configured, then the CDI device will not receive network access policies from the S3C GE.

The second level is the access network where the CDI device is connected to or where the device hands over to. The S3C can (for obvious reasons) not manage the QoS of a foreign network. During a handover, if one of the access networks is not managed by the S3C GE, then this handover will not be seamless (meaning all IP connections will be interrupted).

## 2.3 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 2.3.1 End to End testing

#### 2.3.1.1 *CDI-Webinos (Android Client) Testing*

This is basically quick testing to check that everything is up and running. After you run CDI-webinos application installed on your Android device, please check that processes `org.webinos.android` are running with the command `adb shell ps`. A usage example is provided in the next section.

To validate that A-CDI is running as expected you can exercise a quick *End to End Test* which is supplied. The *End to End Test* is prepared by the installation procedure (the procedure is described above); the files to run the test are stored in the *endtoendtests* folder in the installation directory.

The end2end test may be run locally in a computer (Linux or Windows OS) connected to a smartphone (or an Android emulator), as well as directly in an Android device (smartphone, tablet). To run the test in a computer, the Android SDK is required (see ). First, extract the content in a folder, then run the command:

```
adb forward tcp:8080 tcp:8080
```

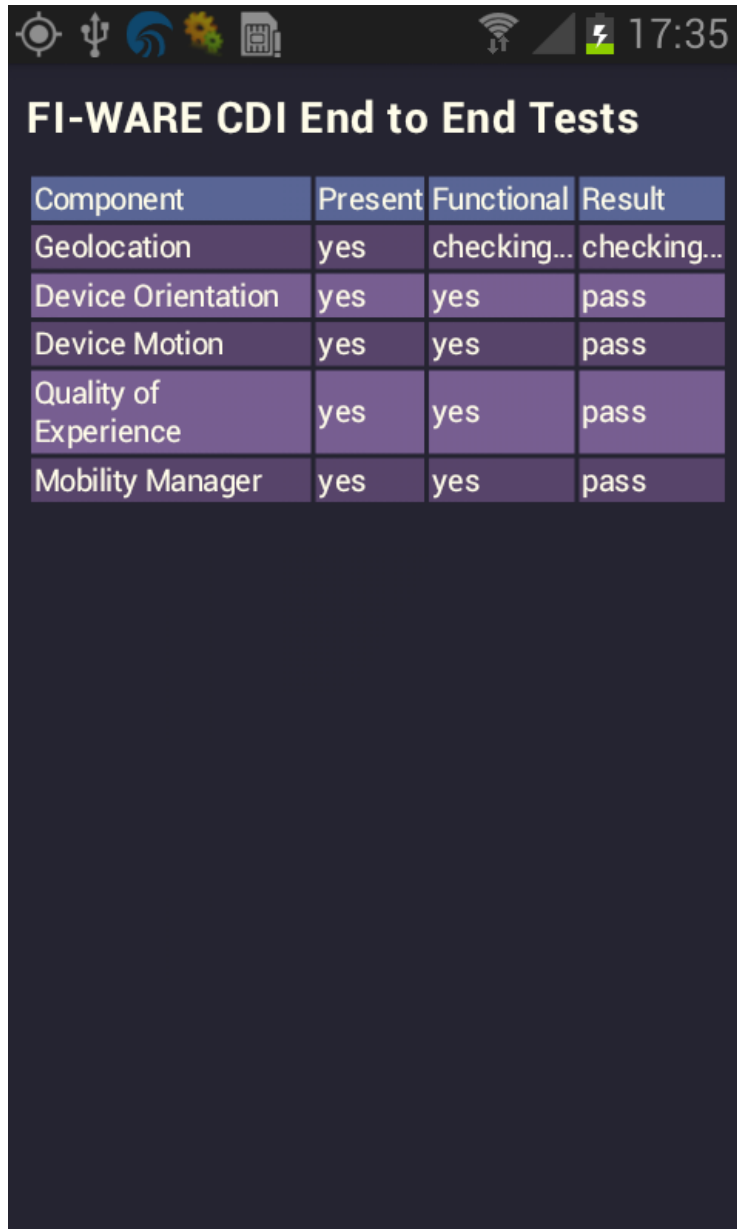
Search for the `index.html` file in the folder used for extraction and open it using an HTML5 compliant internet browser (for instance any recent release of Firefox, Opera, Chrome).

To run the end2end test directly in an Android device, copy the content to a folder in the sdcard of the device. Then run the CDI-Webinos application, open the `index.html` file in the folder used for extraction, using an HTML5 compliant mobile internet browser (for instance any recent release of Firefox Mobile, Opera Mobile, Chrome).

It is also possible to run the End To End widget in an Android device, by launching the CDI-Webinos application, selecting the *Scan SD Card* menu and selecting the widget file (`cdi-endtoend-test.wgt`). The widget will appear in a list in the main screen of the CDI-Webinos application and can be launched clicking on its name.

When the test runs you should see some of A-CDI's functionality tested and the test results displayed in the window; please, consider that the test may take some time to complete. If all of the tests fail, then A-CDI is not configured correctly. If one of the tests pass, then A-CDI is functioning. Errors with individual sub-systems can cause problems with some end to end tests, this can explain situations where some tests pass

and others fail; for instance, the Geolocation test is not Present/Functional when the GPS module is not present or it cannot lock to satellites signal (as shown in the picture below). The Device Orientation and Device Motion function may be not Present/Functional depending on the availability of Accelerometers functions in the device.



Component	Present	Functional	Result
Geolocation	yes	checking...	checking...
Device Orientation	yes	yes	pass
Device Motion	yes	yes	pass
Quality of Experience	yes	yes	pass
Mobility Manager	yes	yes	pass

**End2End Test Without GPS Locking**

**2.3.1.2 Distributed Compute (Sisyphus Server) Testing**

To run the Distributed Compute functional tests, open a terminal and navigate to the `testing/sisyphus_testing` directory. Then execute the following command:

```
sudo node ./testapp.js
```

**NB:** Note that this demo will base its self of the first IP address it can find on the system, excluding localhost. If no IP address is available it will select localhost, 127.0.0.1.



After Distributed Compute is started the console output will contain a line explaining what IP address has been selected:

```
my local ip is being used:192.168.15.12
```

Using your web browser visit the IP address listed by the Distributed Compute server, in the above example this is 192.168.15.12. This should display the Distributed Compute functional testing application.

### 2.3.2 List of Running Processes

The following sections describe how to show the list of running processes both on the CDI-Webinos (Android client) platform, and on the Distributed Compute (server) framework.

#### 2.3.2.1 CDI-Webinos (Android Client) Platform

In order to list running processes, connect your Android device to a Linux PC through USB cable and run `adb shell ps -t` This will return information about the core processes on which CDI-Webinos relies on. Running modules are enclosed in two processes, called `org.webinos.android` and `webinos.android`. These processes can exist in multiple instances, depending on the number of widget that have been launched from the CDI-Webinos application.

The provided command will show a table with the following columns; to filter only the webinos process, use the command chain `adb shell ps -t | grep -i webinos`, as in the picture below:

USER	PID	PPID	VSIZE	RSS	WCHAN	PC	NAME
------	-----	------	-------	-----	-------	----	------

```
adb shell ps -t | grep webinos
u0_a49 6596 1847 451148 81056 ffffffff 00000000 $ org.webinos.android
u0_a49 6619 6596 451148 81056 ffffffff 00000000 $ webinos.android
u0_a49 6620 6596 451148 81056 ffffffff 00000000 $ webinos.android
u0_a49 6621 6596 451148 81056 ffffffff 00000000 $ webinos.android
u0_a49 6665 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6667 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6668 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6669 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6670 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6671 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6672 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6673 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6674 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6675 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6676 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6677 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6697 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6698 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6702 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6707 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6708 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6714 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_a49 6715 6596 451148 81060 ffffffff 00000000 $ webinos.android
u0_i7 6678 1847 379828 57916 ffffffff 00000000 $ org.webinos.android:sandboxed_process0
u0_i8 6730 1847 379356 55036 ffffffff 00000000 $ org.webinos.android:sandboxed_process1
u0_i9 6764 1847 379644 58316 ffffffff 00000000 $ org.webinos.android:sandboxed_process2
```

Webinos Process

#### 2.3.2.2 Distributed Compute (Sisyphus Server) Framework

The distributed compute server framework is a JavaScript based script which is executed by node.js. When running a distributed compute server you will see the additional node.js process running.

Open a terminal session on the server running the Distributed Compute, then run `ps aux | grep 'node'` for the list of node processes executing on the server. The following shows what is returned:

```

USER      PID  %CPU  %MEM    VSZ   RSS TTY      STAT   START   TIME COMMAND
root      21613  0.0   0.1   58600   2052 pts/7    S+    14:50    0:00 sudo
node ./testapp.js
root      21614  0.2   1.0  662228  21584 pts/7    Sl+   14:50    0:00 node
./testapp.js
mcwoods   21631  0.0   0.0   11900    932 pts/1    R+    14:53    0:00 grep --
color=auto node
    
```

The example above the distribute compute testapp.js has been used as an example, you can see the name of the script executed in the output from the `ps` command.

### 2.3.3 Network interfaces Up & Open

The following described the network interfaces used on the CDI-Webinos (Android client) platform, and on the Distributed Compute (server) framework.

#### 2.3.3.1 CDI-Webinos (Android Client) Platform

- ANode activity running on the Android will open the following socket to enable communication with applications:

```

WebSocket port: TCP 8080 127.0.0.1 (localhost)
    
```

The WebSocket port is necessary for A-CDI applications to interact with lower level CDI-webinos components. A sanity check can be performed through the `adb shell netstat` command, to check that the webinos application has established connections on port 8080 at localhost address (127.0.0.1), after having launched any widget from CDI-Webinos or having opened any test page (for instance the `index.html` of the end2end test) in an HTML5 compatible internet browser.

```

adb shell netstat | grep 127.0.0.1:8080
tcp        0      0 127.0.0.1:8080      127.0.0.1:60152    ESTABLISHED
tcp        0      0 127.0.0.1:60158    127.0.0.1:8080     ESTABLISHED
tcp        0      0 127.0.0.1:8080     127.0.0.1:60157    ESTABLISHED
tcp        0      0 127.0.0.1:60154    127.0.0.1:8080     ESTABLISHED
tcp        0      0 127.0.0.1:60155    127.0.0.1:8080     ESTABLISHED
tcp        0      0 127.0.0.1:60156    127.0.0.1:8080     ESTABLISHED
tcp        0      0 127.0.0.1:60153    127.0.0.1:8080     ESTABLISHED
tcp        0      0 127.0.0.1:8080     127.0.0.1:60155    ESTABLISHED
tcp        0      0 127.0.0.1:8080     127.0.0.1:60156    ESTABLISHED
tcp        373    0 127.0.0.1:8080     127.0.0.1:60158    ESTABLISHED
tcp        0      0 127.0.0.1:60152    127.0.0.1:8080     ESTABLISHED
tcp        0      0 127.0.0.1:60157    127.0.0.1:8080     ESTABLISHED
tcp        0      0 127.0.0.1:8080     127.0.0.1:60154    ESTABLISHED
tcp        0      0 127.0.0.1:8080     127.0.0.1:60153    ESTABLISHED
    
```

Netstat command output

- Distributed Compute running on the client will establish a WebSocket connection to one or more specified Distributed Compute servers. It does not support incoming connections.

### 2.3.3.2 *Distributed Compute (Sisyphus Server) Framework*

The distributed compute server framework will open the following socket to enable communication with other distributed compute nodes, these include CDI-Webinos platform running on Android systems and other distributed compute nodes running inside the cloud:

```
WebSocket port: TCP 80 INADDR_ANY
```

The websocket connection is necessary to allow other A-CDI distributed compute instances to connect to the server. More information on the WebSocket protocol can be found in [RFC6455](#).

### 2.3.4 Databases

N/A for the current release.

## 2.4 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution.

Some insights about the way A-CDI integrates in CDI-webinos can help administrators to understand possible sources of errors. CDI exploits CDI-webinos in two ways:

- WebRT layer: Some A-CDI functionalities are natively supported by the native platform (i.e. sensors). In such a case A-CDI does not have to reimplement them but just needs to provide applications a JavaScript frontend, made of a set of `.js` files containing all the aforementioned functionalities. This layer of the A-CDI system doesn't have an own life, but it reacts upon API invocations performed by applications and dispatches API calls/replies between applications and the low level system. That means A-CDI runtime errors cannot happen at this stage (WebRT part of A-CDI doesn't have a runtime), however applications' wrong usage of the API and possible errors can be easily detected by means of common browser's debugging tools (i.e. Firefox or Chrome consoles) and countermeasures can be deployed accordingly.
- WebRT+CDI-Webinos embedded layers: In the other case there are functionalities (i.e. QoE, Mobility Manager) that require both a native implementation and a way for applications to access them from the WebRT. As explained in the previous point, the WebRT part of A-CDI is provided as a JavaScript frontend and same considerations made before keep holding here. On the other hand when native implementations are provided, more A-CDI components are integrated within the base CDI-webinos platform (at lower level). Here we have the A-CDI JavaScript backend (embedded server side API) acting as a glue to the native Java side. Being such components executed within an

Android process (the aNode activity), errors, exceptions and weird situations can be analyzed through the `adb logcat` tool. To this aim an administrator should connect via USB its Android device to a PC. Then fire up a terminal and run `adb logcat`. This will show every output message the GE administrator could have need of.

Please note that errors occurred at this stage often resolve with crash of the aNode activity, or at least with A-CDI (or CDI-webinos) modules shutting down (with relative messages on the logcat).

## 2.4.1 Resource availability

### 2.4.1.1 *CDI-Webinos (Android Client) Requirements*

All the processes run on Android devices and use an amount of memory compatible with most of them. Memory usage may depend on Android release. However in practice the maximum memory consumed by the A-CDI GEi is 64 Mbytes. With the memory management functionality provided by the Android OS the A-CDI GEi should launch and run fine, even when less than 64Mbytes of RAM is available. The Android OS will automatically terminate processes to free memory for the A-CDI GEi. However if errors occur, it is recommended that the user manually kill unwanted processes until at least 64 Mbytes of RAM is available.

There are no special requirements in terms of HW resources.

### 2.4.1.2 *Distributed Compute (Sisyphus Server) Requirements*

The cloud (server) requirements for a Sisyphus is as follows:

- RAM: 512 megs
- CPU: 1 core
- Hard Drive: 8 Gig (used 1.7G, including development tools not needed for a simply deployment)

## 2.4.2 Remote Service Access

The Mobility Manager component interacts with the S3C GEi in order to do REST requests to the [OTT API interface of S3C](#) and in order to receive access network policies from the ANDSF component of the S3C GEi.

When running on the Android platform, the A-CDI client may make a connection to the Distributed Compute server framework. To do so it establishes a websocket connection, as described above.

## 2.4.3 Resource consumption

The following describes the resource consumption on both the CDI-Webinos Platform (Android Client) and on the Distributed Compute Framework (Sisyphus Server).

**2.4.3.1 CDI-Webinos (Android Client)**

RAM: The maximum amount of RAM consumed by A-CDI, with a small running application should be 64Mbytes. Memory consumption significantly greater than this suggests a memory leak within the CDI-Webinos platform. In this case the CDI-Webinos platform should be terminated and restarted.

CPU Consumption: An accurate measure is not available, however existing usage suggests that anything greater than 20% CPU utilization for a sustained period of time (e.g. 3 minutes), would indicate an infinite loop and blocked / crashed process.

In case of either increased CPU consumption, or significantly greater than 64Mbytes or RAM consumed by A-CDI then the recovery step is to terminate the CDI-Webinos platform and restart it.

**2.4.3.2 Distributed Compute (Sisyphus Server)**

RAM: Under a small load the RAM consumed by A-CDI's distributed compute server component You can assess the amount of memory currently allocated to the distributed compute element by using the [smem](#) command; `sudo smem | grep 'node'` produces the following output:

PID	User	Command	Swap	USS	PSS
21613	root	sudo node ./testapp.js	0	548	718
21614	root	node ./testapp.js	0	42288	42333

Converting RSS to bytes is possible by multiplying the RSS by the system's page size. In the system above the page size is 4096 bytes. This value is obtained by using the `getconf PAGESIZE` command. This gives a RAM load at idle of 178372608 bytes, or 170 Mbytes.

CPU Consumption: Under a similar idle bases CPU usage will be less than 1%. If the system appears to be idle, but CPU load is over 5% then the distributed compute server component should be killed and restarted.

The following command can be used to monitor the CPU utilization of the distributed compute element:  
`while [ 1 ]; do ps -C node -L -o command,pcpu|sort -k 2 -n; echo; sleep 1; done.`  
 This will produce output similar to the following:

COMMAND	%CPU
node ./testapp.js	0.0
node ./testapp.js	0.0
node ./testapp.js	0.0

```
node ./testapp.js      0.0
node ./testapp.js      0.0
node ./testapp.js      0.9
```

Further testing of the distribute compute server is required to obtain similar data for load characterization.

#### 2.4.4 I/O flows

I/O, or Network traffic is not generated by default by A.-CDI. However when some services and APIs are requested then network traffic will ensue. Examples of this include the Geo-Location API. When requests for the devices current location are made, network traffic is generated by the Android Operating system.

##### 2.4.4.1 **Mobility Manager**

If the Mobility Manager component of A-CDI is configured to receive access network policies from the S3C GEi, upon starting the Mobility Management service for Android (via JavaScript API or provided external application) the service will send an HTTP request to the ANDSF component of the S3C GEi and will keep a socket connection open in order to be reachable by the ANDSF. Every time the phone changes the access network, the Mobility Management service will send an HTTP POST request to the ANDSF in order to notify the ANDSF about the new network location.

If an application uses the network resource allocation capabilities of the Mobility Management component, the device will send REST requests to the [OTT API interface of the S3C GEi](#).

##### 2.4.4.2 **CDI-Webinos Platform**

The CDI-Webinos platform will remain operational, with reduced functionality (e.g. Geo-Location) when network connectivity and an internet connection is not available.

##### 2.4.4.3 **Distributed Compute Framework**

When the Distributed Compute Framework is used the main IO Flows are an outgoing Web Socket connection from A-CDI on the Android device to a Distributed Compute server on Port 80.

## 3 Cloud Edge - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 3.1 Installation

This document describes the installation, setup and administration of the Cloud Proxy box.

It must be noticed that prior to release 1, the Cloud Proxy was distributed as a software module to be installed on a regular PC over a fresh Ubuntu 12.04 LTS installation. This old version (preview) is not anymore maintained. The Cloud Proxy is now released as a hardware box with the embedded software already installed.

Q2/2013 version is to be considered as a beta release, Q4/2013 is the finalized release.

The Cloud Proxy box is based on an existing commercial device (Dti716 aka "CuboBox mini"). The Dti716 is currently used by Telecom Italia on its national market. It is therefore compliant with the various standards in use inside the European Union and is "CE marked". This means it complies with electrical safety, electromagnetic compatibility and other requirements. It can be deployed in normal customer premises during experimentation without specific constraints.

**The Cloud Proxy is designed only for home / small offices environments and does not comply with industrial or outdoor usages**

Hardware architecture

The box is based on a STB System On Chip that provides:

- A 1.2GHz ATOM-class x86 CPU supporting multi-threading and a 512KB L2 cache
- up to 8GB of flash storage (storing of the virtual machines, parameters, databases etc ...)
- up to 2GB of RAM
- one USB 2.0 port (can be used to add external storage (hard disk, flash dongle ...) or other devices (zigbee dongle, webcam ...))
- one ethernet (100TX) RJ45 port
- one internal WiFi 802.11b/g/n module

The box is also providing STB-specific I/Os (HDMI, video, sound, antenna in/out) **that are NOT CURRENTLY SUPPORTED by the FIWARE distribution**. These dedicated I/Os could be activated depending on the PPP FI use-case projects requests (to be discussed).

The box is provided with a power adapter (standard continental EU power plugs only. For usage in UK or Ireland, please use a user-provided power plug adapter).

### 3.1.1 STEP1: Install the Cloud Proxy Software on the box

A simple mechanism is provided by the bootstrap in order to allow the Cloud Proxy SW to be easily replaced by a new version even if no network connectivity is available (first setup, recovery after catastrophic failure etc... ). This requires the current SW to be installed on a USB flash dongle.

STB software updates can be released as files named cloudedge-rootfs-update-MAJOR\_VERSION.MINOR\_VERSION

In order to update the STB software :

```
- put all available updates on a USB key
- plug the USB key on the STB
- switch on the STB
  -> required updates are applied and the STB automatically reboot
when it's finished
- remove the USB key of the STB
```

### 3.1.2 STEP2: Environment setup

This part of the Installation guide explains how to setup a development environment on a regular linux PC.

A PC based on a Linux system with a FTP server is required to install and use the STB SDK release The CloudEdge Virtual Machine images are stored on the FTP server

Install the toolchain on the PC :

```
$ mkdir -p /opt/IntelCE/IntelCE-25/i686-linux-elf
$ tar xjf toolchain_bin.tar.bz2 -C /opt/IntelCE/IntelCE-25/i686-linux-elf
```

And add the toolchain cross-compiler in the PATH :

```
$ export PATH=/opt/IntelCE/IntelCE-25/i686-linux-elf/bin:$PATH
```

This toolchain optimized for the 32-bit STB target must be used for software building (cross-compilation)

STB with <STB IP address> configured by DHCP and PC with <PC IP address> must be on the same network

Copy files of the CloudEdgeVMStore directory in the FTP server directory in order to install CloudEdge Virtual Machines on STB



## 3.2 Sanity check

Note1: The current release of the Cloud Proxy is considered as a beta, a following version will include more on-board testing facilities. Note2: the Cloud Edge is an embedded device, most of the following paragraphs have been thought for Cloud based applications and are not relevant.

### 3.2.1 End to end testing

**test1: test that the Cloud Proxy has an IP address** The Cloud Proxy is intended to be connected to a LAN that sports both a linux PC for entering networking commands and a DHCP server (the linux PC can do both).

When the Cloud Proxy is connected to this LAN and switched on, it'll do some internal tests, boot up procedure and than ask for an IP address.

Having a look to the dhcp deamon logs will give this address (the MAC address of the Cloud Proxy is printed on a small label sticked on it).

another procedure can be:

```
ping -b x.y.z.255  
arp -a
```

(x.y.z.255 is expected to be the broadcast address for the LAN)

Thar apr -a command will list the MAC / IP@ couples.

#### **test2: get the Cloud Proxy resources**

expected: the Cloud Proxy address is 10.12.56.106, this have to be replaced by the address you found in the previous step.

Command on host PC:

```
curl --user smithj:secret http://10.12.56.106:8080/cloudedge/platform
```

Result: :

```
{  
  "platform": {  
    "name": "DTI716 IntelCE Cloud Proxy",  
    "version": {  
      "major": 1,  
      "minor": 1  
    },  
  },  
}
```

```
"ip": "10.12.56.106",
"resources": {
  "cpu": "Intel(R) Atom(TM) CPU CE4253 @ 1.20GHz (siblings = 2)",
  "ram": 603,
  "disk": 2162
},
"metrics": {
  "cpuUsage": 28,
  "ramUsage": 79,
  "diskUsage": 195,
  "networkIn": 64889,
  "networkOut": 10455
}
}
```

**testx: further testings**

If the simple sanity tests described below are not enough, it is advised to have a look to the testing procedure document [Cloud Edge - Unit Testing Plan](#)

### 3.2.2 List of running process

The Cloud Edge is a stand alone embedded device, running processes are not relevant.

### 3.2.3 Network interfaces up & open

The Cloud Edge has one ethernet port (physical).

### 3.2.4 Databases

There are no externally available databases

## 3.3 Diagnosis Procedures

The various commands described in [Cloud Edge - Unit Testing Plan](#) can be used for diagnostic.

It must be noticed that some commands are used to update the embedded software. These commands are not strictly API functions but "maintenance" features that have been developed for the experimenters.

They allow the Cloud Proxy to be locally updated even if the internal software has been destroyed and/or there is no Internet access.

These update commands are:

- Test STB update: Add File and update existing file
- Test STB update: remove file
- Test STB update: Apply multiple update successively
- Check than version return by cloudedge is updated

### 3.3.1 Resource availability

The Cloud Proxy is provided as an HW box with SW already installed. There is no resource compatibility to check.

### 3.3.2 Remote service access

The Cloud Proxy is a stand alone HW box. It answers to services requests as described in the API documentation.

A rapid test to check remote service access is the following one:

Check the version returned by cloudedge

Command on host PC:

```
curl --user smithj:secret http://10.12.56.106:8080/cloudedge/platform
```

Result: :

```
{
  "platform": {
    "name": "DTI716 IntelCE Cloud Proxy",
    "version": {
      "major": 2,
      "minor": 7
    },
    "ip": "10.12.56.106",
    "resources": {
      "cpu": "Intel(R) Atom(TM) CPU CE4253 @ 1.20GHz (siblings = 2)",
```

```
    "ram": 603,  
    "disk": 2162  
  },  
  "metrics": {  
    "cpuUsage": 2,  
    "ramUsage": 79,  
    "diskUsage": 180,  
    "networkIn": 8106,  
    "networkOut": 3782  
  }  
}  
}
```

### 3.3.3 Resources consumption

Not relevant because there are no resources that have to be or that can be shared with other devices. The Cloud Edge is a stand alone device.

### 3.3.4 I/O flows

I/O flows are HTTP REST based. As the Cloud Edge is a stand alone device, there are no specific reservations to be done (bandwidth ...).

## 4 Cloud Edge OSGi - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 4.1 Introduction

This manual provides instructions to install and administer an OSGi Equinox framework, and its SDK, on a [Cloud Edge Generic Enabler](#) implementation, whose Open Specifications are available at [this page](#). The two operations are described in the main sections of the manual.

Moreover, a final section of the manual describes as a sort of example application, the activation of a component in the OSGi framework. The component is another FI-WARE GEi, that is the [IoT Gateway ZigBee Protocol Adapter](#).

### 4.2 OSGi Equinox framework Installation

This section describes the installation and setup of the OSGi Equinox framework on a dedicated container of the Cloud Proxy box. The framework is based on the Oracle Java Development Kit 7 for x86 Linux, while the OSGi 3.5.2 Equinox implementation is profiled as requested by the HGI Residential Profile specification with some optional services specified by Telecom Italia for its new High End Residential Router.

The installation procedure here detailed needs a preliminary knowledge of the documentation [Cloud Edge - Installation and Administration Guide](#) and [Cloud Edge - User and Programmers Guide](#). The STB HW and SW platform needs to be activated and tested before going on with the operation here described (see [Cloud Edge – Unit Testing Plan](#)).

However it is also possible to activate the OSGi container simply following the later instructions.

#### 4.2.1.1 **Step 1: Activation of the Cloud Edge STB and FTP server on the Local LAN**

The Cloud Edge STB needs to be activated on a LAN and its IP address should be identified as described in the [Installation and Administration Guide](#). This IP address is later identified as <STB\_IP\_address>. On the LAN a FTP server should also be activated. Its IP address is later identified as <PC\_IP\_address>. The two files "rootfs-final-1.1.tar.gz" and "manifest.xml.tar", available on the [FI-WARE Forge file repository](#), under release 3.2. at the "I2ND-CloudEdge" package, should be loaded on the FTP download directory. We assume that the same PC which is hosting the FTP server will be used to access the STB HTTP REST API (APIs used to control the CE) in order to instruct the Cloud Edge to activate the OSGi Equinox Framework container. We also assume that the PC runs a Linux O.S.

#### 4.2.1.2 **Step 2: Activate the OSGi Equinox framework on a dedicated container**

This section describes how to activate the OSGi Equinox framework on the STB. The command through the HTTP REST interface instructs the Cloud Edge to activate a new container by downloading the root file system indicated in the "EquinoxFramework.manifest.xml.tar" file. The "login" and "pass" are parameters of the FTP server.

To preliminary check that the CE is in the right state (after the Power-on), before downloading the OSGi image, use the following commands:

```
curl --user smithj:secret
http://<STB\_IP\_address>:8080/cloudedge/platform -d '{"reset" : null}'
```

and

```
curl --user smithj:secret http://192.168.1.2:8080/cloudedge/images
```

If the CE is replying that no 'Image' are available go on with the image download. In case of problem refer to the CE Proxy documentation.

```
curl --user smithj:secret http://<STB\_IP\_address>:8080/cloudedge/images
-d '{"image" : {"imageRef" :
"ftp://login:pass@<PC\_IP\_address>/manifest.xml.tar"}}'
```

If the above command is successful, the STB should reply with the following output (assuming that the image id = 1 and the <STB\_IP\_address> is 192.168.1.34)

```
"image": {
  "id": 1,
  "name": "tilab-container",
  "link": "http://192.168.1.34:8080/cloudedge/images/1"
}
```

The next step is the activation of the OSGi Equinox container. To do that it is requested to submit the following command to the Cloud Edge.

```
curl --user smithj:secret http://192.168.1.34:8080/cloudedge/instances
-d '{"instance" : {"imageRef" : 1}}'
```

The output provided by the Cloud Edge will be

```
"instance": {
  "id": 1,
  "name": "tilab-container-instance-1",
  "link": "http://192.168.1.34:8080/cloudedge/instances/1"
}
```

Now we can move the OSGi container in the running state. This can be requested with the following command

```
curl --user smithj:secret
http://192.168.1.34:8080/cloudedge/instances/1 -d '{"start" : null}'
```

#### 4.2.1.3 Step 3: List the OSGi Container parameters

Last step is detecting if the OSGi-Container instance is correctly running and discovering the LAN IP address assigned to this new instance. The command is:

```
curl --user smithj:secret http://192.168.1.34:8080/cloudedge/instances/1
```

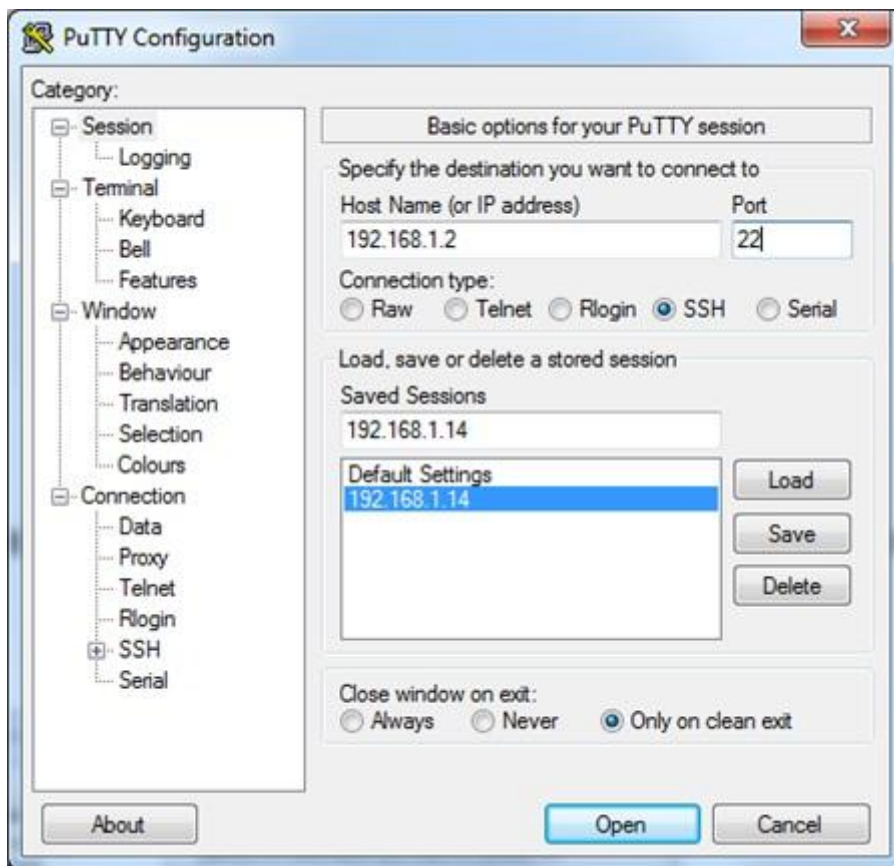
Then check for the following output

```
"instance": {
  "id": 1,
  "name": "tilab-container-instance-1",
  "status": "RUNNING",
  "ip": "192.168.1.2",
  "metrics": {
    "cpuUsage": 0,
    "ramUsage": 832,
    "diskUsage": 149,
    "networkIn": 57587837,
    "networkOut": 1359043
  },
  "image": {
    "id": 1,
    "name": "tilab-container",
    "link": "http://192.168.1.34:8080/cloudedge/images/1"
  },
  "created_by": "smithj",
  "created_at": "2013-11-15T13:59:23+00:00",
  "link": "http://192.168.1.34:8080/cloudedge/instances/1"
}
```

## 4.2.2 Sanity check

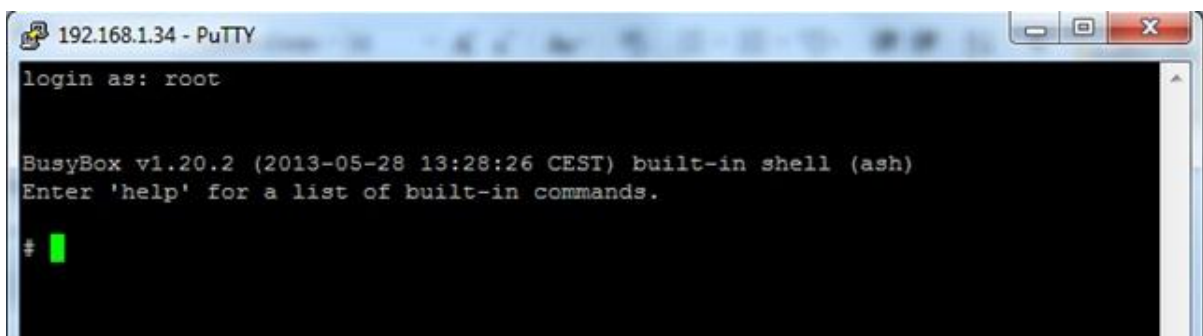
### 4.2.2.1 End to End testing

As above described, the new OSGi-Container is getting a new 'OSGi-Container\_IP\_Address' on LAN (see the previous step 3 at the IP TAG).



**PuTTY Configuration Window**

On the PC (the same hosting the FTP server) activate a Putty SSH client (<http://www.putty.org/>), specify in the 'Host Name' the IP of the Instance (<OSGi-Container\_IP\_Address>) and connect to the OSGi-Container. Logon as a 'root' user without specifying any password.



**PuTTY Command Window**

With the command



```
ps axw | grep java
```

it's possible to check if the OSGi framework is correctly running. The following output should be provided

```
/opt/jre/bin/java -Xdebug -
Xrunjdpw:transport=dt_socket,server=y,suspend=n,address=8000 -Xmx32m -
XX:MaxPermSize=32m -XX:ReservedCodeCacheSize=16m -
Dosgi.compatibility.bootdelegation=true

-jar org.eclipse.osgi_3.5.2.R35x_v20100126.jar -configuration
../configuration
```

The OSGi-Container also includes a Web Server service . To check if it works, simply run a browser specifying the '[http://<OSGi-Container\\_IP\\_Address>/](http://<OSGi-Container_IP_Address>/)' URL. The Jetty Web server will reply



**Error 404 accessing the OSGi Web server**

#### 4.2.2.2 **List of Running Processes**

The Cloud Edge OSGi is running on the Cloud Edge stand alone embedded device, running processes are not relevant.

#### 4.2.2.3 **Network Interfaces Up & Open**

The Cloud Edge OSGi is running on the Cloud Edge which has only one ethernet port (physical).

#### 4.2.2.4 **Databases**

There are no externally available databases

### 4.2.3 **Diagnosis Procedure**

The various commands described in [Cloud Edge – Unit Testing Plan](#) can be used for diagnostic.

#### 4.2.3.1 **Resource availability**

The Cloud Edge OSGi resources can be checked as detailed in "Step 3" of the "OSGi Equinox framework Installation" section

#### 4.2.3.2 **Remote service access**

The Cloud Edge OSGi is a framework. The Remote Services are provided by the third party application bundles. Refer to the documentation of the ZPA component developed by the WP5 on the OSGi framework

#### 4.2.3.3 **Resources consumption**

The Cloud Edge OSGi framework is running in a Cloud Edge container. The resources are allocated to each image by the container Refer to the "<http://10.12.56.106:8080/cloudedge/image>" command provided by the Cloud Edge

#### 4.2.3.4 **I/O flows**

The interfaces are provided by the third party OSGi application. Cloud Edge OSGi framework interface is only for control/managing and has not requirements for bandwidth, response time, etc.

## 4.3 OSGi SDK Installation

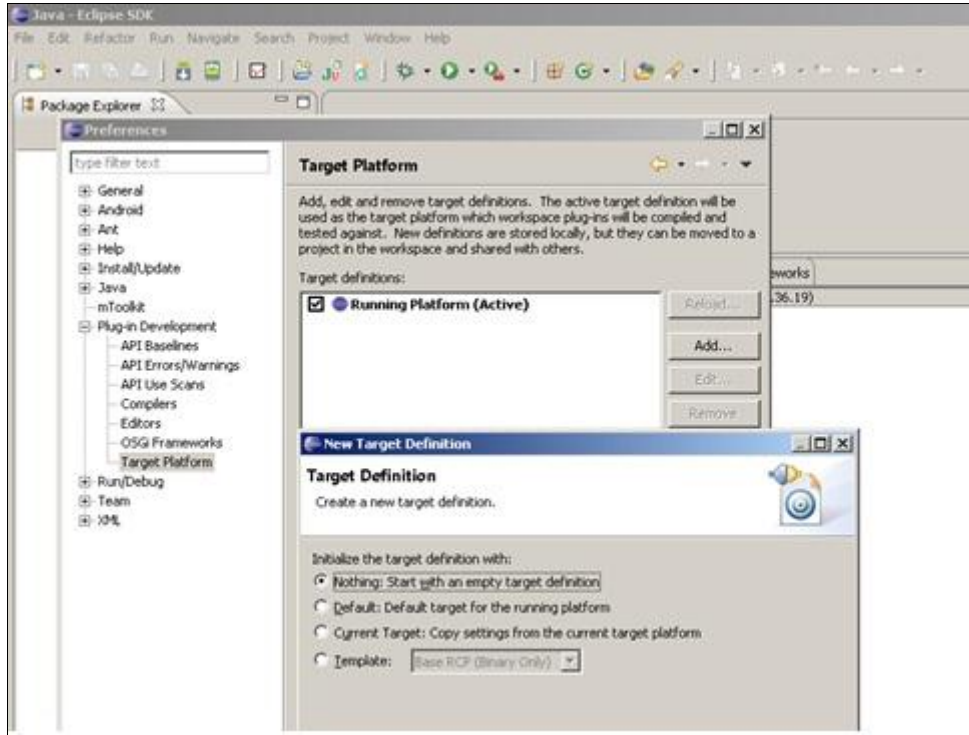
### 4.3.1 Installation

This section describes the activation of the OSGi SDK based on the Eclipse framework. The Eclipse is an open source project providing a full IDE (Integrated Development Environment) for Java and other languages. Eclipse can be downloaded from the (<http://www.eclipse.org/>) and can be activated on Linux and Windows platform with the Java runtime installed. The Eclipse version suggested for the OSGi SDK is the 4.3.1 identified as 'Kepler'

### 4.3.2 Configuration of the PDE target platform

To develop new OSGi Bundles for the Cloud Edge OSGi Equinox Framework some preliminary configurations are requested for the Eclipse IDE. The steps are:

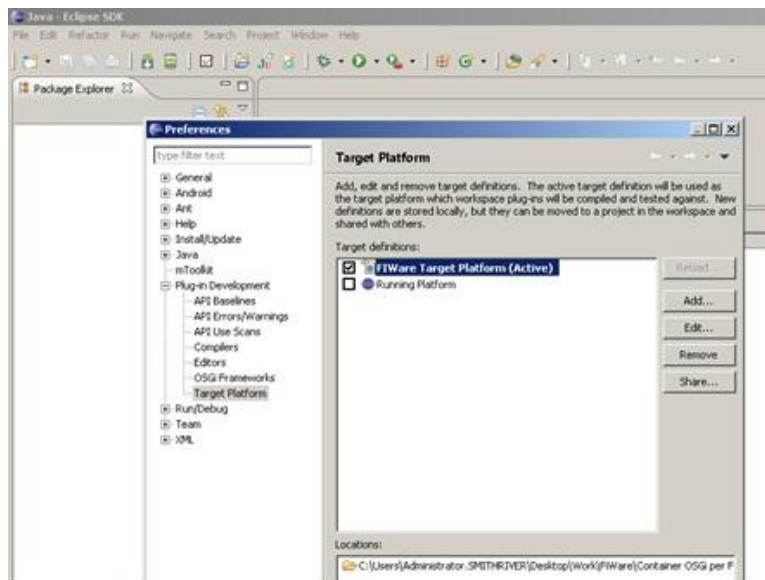
1. Select the "Plug-in Development" perspective, Windows > Open perspective > Other > Plug-In Development
2. Import the "CloudEdge-OSGI-Equinox-Framework" target platform into Eclipse following the next instructions: from the tab "Window > Preferences" select the option "Plug-In-Development>Target Platform" , click on the 'Add' button to open the dialog box "Target Definition".



**Target Platform configuration on Eclipse**

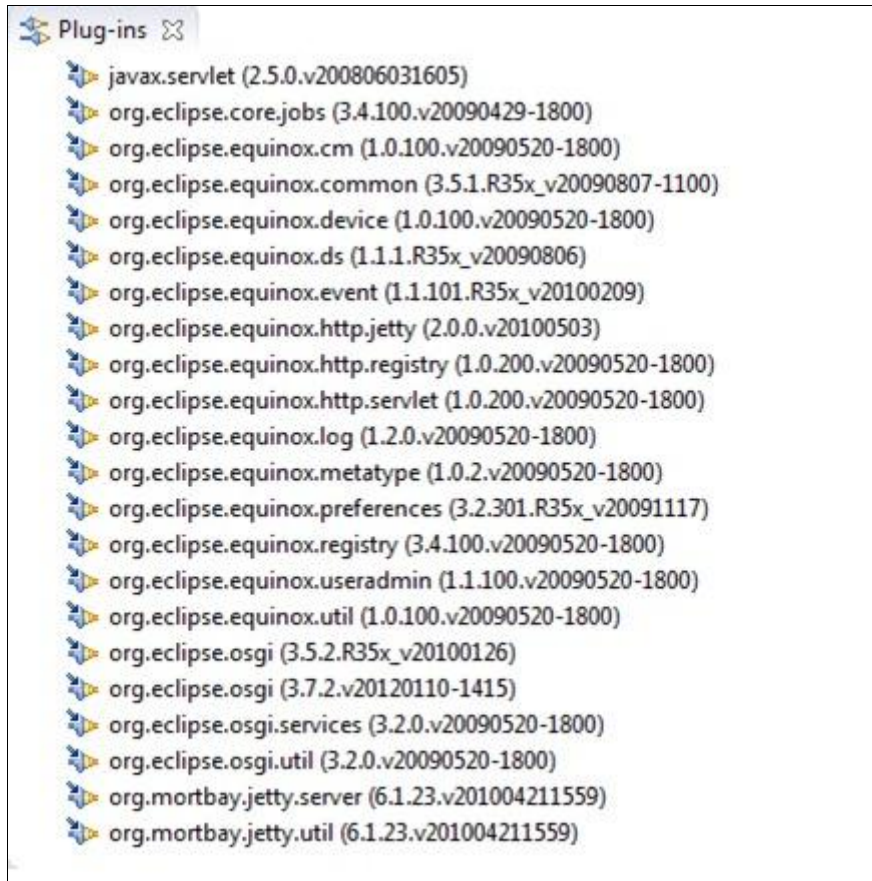
3. Select “Nothing: ...” and click on “Next”. Assign a name to the new target platform (i.e. FI-WARE target Platform) and click on ‘Add’. Select ‘Directory’ and click on ‘Next’, select the ‘Target Platform’ directory you have unzipped from the file ‘Target Platform.tar’ downloaded from the FI-WARE Forge file repository at the “I2ND-CloudEdge” group. Click on ‘Finish’

4. Next step is the activation of the FI-WARE target platform selecting the relative check box and clicking ‘Ok’



**Target Platform activation on Eclipse**

6. Select the Tab “Windows > Show View > Plug-Ins” to have the list of the Plug-ins included in the ‘FI-WARE target Platform’. A list of 22 bundles will be displayed.



**Target Platform Bundle list**

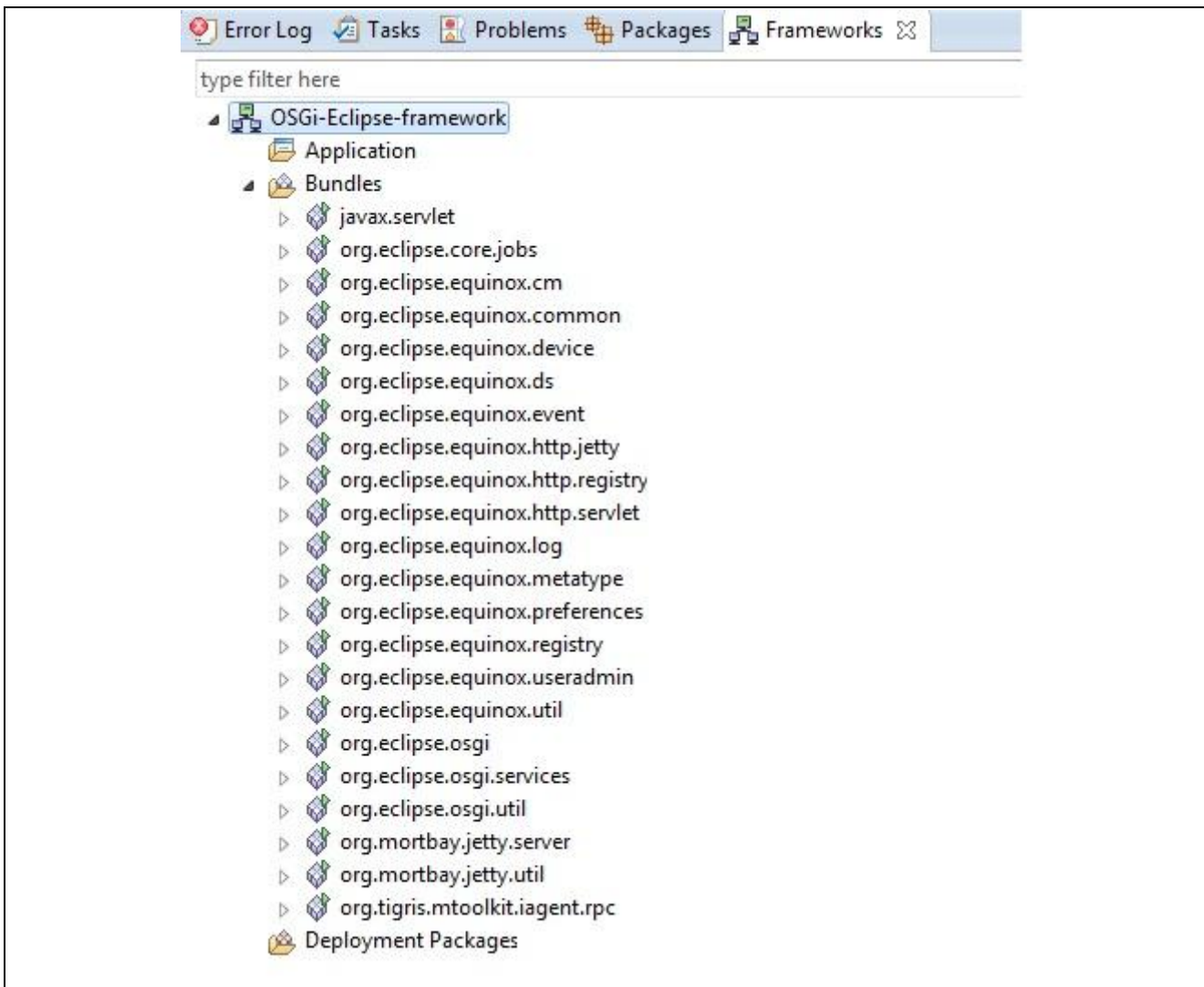
#### 4.3.3 OSGi SDK –The mToolkit

The mToolkit project equips the Eclipse with the necessary facilities for the development and deployment of the OSGi components and applications. The main mToolkit feature is the remote OSGi management which conforms to ‘Deployment Admin Service’ specifications. To support the mToolkit in Eclipse, the ‘mToolkit Instrumentation Agent RPC Bundle’ is included in the i‘Cloud Edge – OSGi Eclipse framework’ To activate mToolkit in Eclipse it is requested to download the .zip file from the [‘http://mtoolkit.tigris.org/’](http://mtoolkit.tigris.org/) and unzip it in the ‘dropins’ directory of the eclipse installation. Eclipse also needs to be restarted from ‘Menu File’ > ‘Restart’. To open the mToolkit in Eclipse select ‘Windows’ > ‘Show view’ > ‘Others’ > ‘mToolkit’ > ‘Frameworks’. In the new ‘Frameworks’ windows click on the ‘ Add Framework’ tab and specify the ‘OSGi-Container\_IP\_Address’ as described in the “OSGi Equinox framework Installation and Administration Guide” (see picture below).



**mToolkit on Eclipse**

Finally double click on 'Bundles' folder to list the installed bundles. On the Eclipse 'Frameworks' windows click on the 'OSGi-Eclipse-framework' and select 'Bundles'. The list of the active bundles on OSGI-Equinox-Framework I2ND.CE platform will be displayed with the possibility to stop, start and update them.



#### list the Bundles on the Cloud Edge using mToolkit

To deploy a new bundle developed with the OSGi-SDK simply click (right button) on 'Bundles' and select 'Install Bundles', a new popup will allow you to select the .jar file to be installed.

### 4.3.4 Sanity check

#### 4.3.4.1 End to End Testing

To check if the OSGi-SDK is properly working, the 'Hello World' bundle is provided. In Eclipse select the 'OSGi-Eclipse-Framework' from the 'Framework' windows, click on 'Bundles' (right button) and select 'Install Bundle'. A popup windows will allow you to select the 'it.telecomitalia.osgi.fiware.helloworld\_1.0.0.201311051546.jar' file you have previously downloaded from the FI-WARE Forge file repository. From Eclipse 'Framework' windows check if the new bundle is installed and in the running state. Finally check if the bundle is properly working simply running a browser and specifying the '[http://<OSGi-Container\\_IP\\_Address>/index.html](http://<OSGi-Container_IP_Address>/index.html)' URL. The 'Hello World!' message will be displayed.





**Check the correct activation of the Hello World bundle**

In Eclipse select again the 'it.telecomitalia.osgi.fiware.helloworld' and stop it. Refresh the browser page and now an 'HTTP ERROR 404' will be reported.

#### 4.3.4.2 **List of Running Processes**

The OSGI SDK is a SW toolkit. List of running processes is not significant

#### 4.3.4.3 **Network Interfaces Up & Open**

The OSGI SDK is a SW toolkit. List of the Network Interfaces is not applicable

#### 4.3.4.4 **Databases**

There are no externaly available databases

### 4.3.5 **Diagnosis Procedure**

For the Eclipse framework refer to the manual ([www.eclipse.com](http://www.eclipse.com)). About the OSGi plug-in it is possible to check the mToolkit as described in the documentation of the "OSGi SDK –The mToolkit"

#### 4.3.5.1 **Resource availability**

The OSGi SDK is running on a Win/linux PC that is not part of the FI-WARE platform. Resource availability is not significant

#### 4.3.5.2 **Remote service access**

The OSGi SDK is SW toolkit. Remote service access is not applicable

#### 4.3.5.3 **Resources consumption**

The OSGi SDK is running on a Win/linux PC that is not part of the Fi-Ware platform. Resource consumption is not significant

#### 4.3.5.4 I/O flows

I/O flows is not applicable for an SDK

#### 4.3.6 Activation of the IoT Gateway protocol Adapter - ZPA in the Cloud Edge OSGi Equinox framework

This section describes the integration and activation of the Gateway Protocol Adapter-ZPA GEi provided by the IoT chapter of FI-WARE (see the [Gateway Protocol Adapter - ZPA User and Programmers Guide](#) for details) within the OSGi Equinox framework of the Cloud Edge.

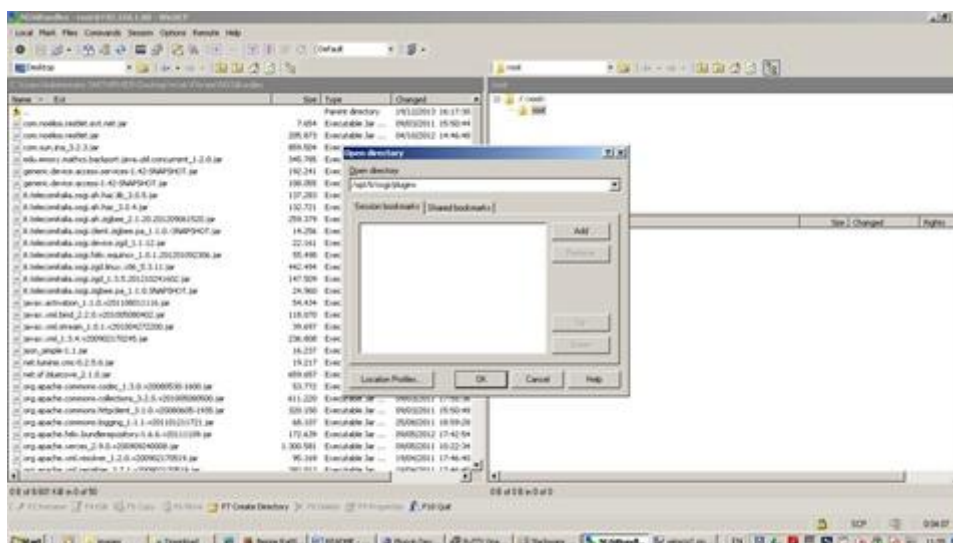
The reader should be familiar with the ZPA functionality and the contents of the related documentation and manuals before proceeding with this section.

The API bundles are provided by the IoT chapter.

In order to fast downloading all the requested bundles on the CE Proxy a client SFTP (winscp.net/) is used instead of mToolkit above mentioned.



Run WinSCP from a PC, a configuration windows is displayed and is requested to insert in the 'Host name' field the 'OSGi-Container\_IP\_Address' and specify both the 'port number' and the 'User name' as indicated in the picture above. Please check that the SFTP protocol is selected





The SCP client presents both the local (PC on the left side) and remote (CE on the right side) file systems. On the right window select the '/opt/ti/osgi/plugins' directory of the CE OSGi Equinox framework container. On the left windows select the PC local directory where you have unzipped the 'NGSI-bundle.zip' file, available on the FI-WARE Forge file repository at the "[I2ND-CloudEdge](#)" group, under release 3.2.

Copy the 50 files from the PC to the '/opt/ti/osgi/plugins' CE OSGi Equinox container directory.

Moreover copy from the PC to the CE the '/opt/ti/osgi/configuration/' target directory the 'config.ini' file. Also this file is available on the the FI-WARE Forge file repository. The old 'config.ini' on the CE will be overwritten.

To activate the new IoT component first you need to connect the ZigBee USB Key (please see the IoT documentation) to the USB port of the CE then stop & start the container with the two commands on the <PC\_IP\_address>:

```
curl --user smithj:secret http://IP\_CONTAINER:8080/cloudedge/instances/1
-d '{"stop" : null}'

curl --user smithj:secret http://IP\_CONTAINER:8080/cloudedge/instances/1
-d '{"start" : null}'
```

To check if the "Gateway Protocol Adapter-ZPA" is properly running simply open a Browser to the URL: '[http://IP\\_CONTAINER:8020/zpa/](http://IP_CONTAINER:8020/zpa/)'. The 'The ZigBee Protocol Adapter is running' message will be displayed. More complete tests on the "Gateway Protocol Adapter-ZPA" are available at the IoT WP documentation wiki pages



## 5 NetIC GE - OFNIC - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 5.1 Goal of the document

The OFNIC is an implementation of the [NetIC Generic Enabler Open Specifications](#). This GEi is in charge of providing a common programmable interface to an Openflow Network, by collecting information and statistics regarding the managed Openflow Network. This interface is based on the NetIC Open API RESTful specifications.

The OFNIC GEi is an extension of the open-source NOX Controller [1]. It relies on the Openflow protocol to retrieve network information about the managed network. In order to collect more accurate statistics, the SNMP protocol [2] has been utilized. For this purpose an SNMP Sub-Agent has been implemented based on Agent-X protocol [3] and it is integrated in all the network nodes.

The OFNIC GEi queries all nodes both for Openflow and SNMP information. The OFNIC GEi provides also a Graphical User Interface (GUI) based on web technologies. Basically this is a web page with javascript code that communicates with the RESTful interface of the GEi. Goal of this document is to provide a useful guide for the installation of OFNIC GEi, together with its GUI, but also the SNMP Sub-Agent that runs on the network nodes. OFNIC GEi collects more accurate statistics if the SNMP Sub-Agent is present in the network nodes. Otherwise, if it is not possible (for administrative reasons) to install the SNMP Sub-Agent in the network nodes, the OFNIC GEi will provide the network statistics relying only on the Openflow protocol.

The document starts describing basic software and hardware required to support the OFNIC GEi on top of a device. It then follows with specific technical information that might help users and administrators: running processes, diagnosis tests, network flows etc.

### 5.2 OFNIC GEi

#### 5.2.1 Software and Hardware environment

The Hardware requirements are a PC or Laptop with at least 1GB RAM, while there are not recommended values for processor speed but it is the biggest performance impact factor.

The OFNIC GEi can theoretically run in any Linux-based operating system. The installation has been tested, thus is recommended to choose, one the following Linux distributions:

- Ubuntu 10
- Debian 6

The operating system listed above are chosen, because it has the following library version constraints satisfied:

- Python version at least 2.6 up to 2.7.3

- GCC version up to 4.4.5
- Boost library version at least 1.40 up to 1.43
- Swig version at least 1.3.0
- Autoconf (GNU Autoconf) version up to 2.67

## 5.2.2 Prerequisites

Root privileges are required to install additional software in the hosting machine. The following packages are required for the correct compilation and execution of the OFNIC GEi:

```
sudo apt-get install build-essential
sudo apt-get install git autoconf libtool libboost-all-dev swig libssl-
dev
sudo apt-get install python-simplejson
sudo apt-get install python-setuptools
sudo apt-get install python-twisted-web
sudo apt-get install mysql-server mysql-client
sudo apt-get install php5 php5-mysql libapache2-mod-php5
sudo apt-get install python-tz
sudo easy_install pysnmp
sudo apt-get install python-mysqldb
```

The prerequisites packages and the software dependencies can be resolved by running the following script from the FI-WARE forge:

```
https://forge.fi-ware.org/frs/download.php/1407/prerequisites.sh
```

## 5.2.3 Getting source code

The source code of the OFNIC GEi is located in FI-WARE forge, and can be downloaded at the following link:

```
https://forge.fi-ware.org/frs/download.php/1197/ofnic\_R3.3-master.zip
```

Another way of getting the source code is to download it from the public repository of Github: ofnic\_R3.3. One can clone (download) the source code with the following terminal command:

```
git clone git@github.com:sanandrea/ofnic_R3.3.git ofnic
```

## 5.2.4 Compilation

Using a system shell locate in the OFNIC source code folder. Run the commands step by step:

```
make configure
make compile
```

You will be prompted for a privileged mysql user and the relative password in order to create a new mysql user that is going to manage the OFNIC mysql database.

## 5.2.5 Configuration

Most of the configuration parameters can be passed as arguments in the execution command needed to launch the main process. OFNIC GEi is developed in a modular way, so the administrator can choose which component to load at each launch of the instance. Components are declared in meta.json files under each folder in the source code. For example in

```
ofnic/src/nox/netapps/
```

there are several folders that contain one or more components, each one declared in a meta.json file (which is unique in each sub-folder). Arguments of configuration can be passed to each component that is loaded (if the component expects any). The arguments are delimited with the = and , characters. For example to configure the tcp port and ssl port of the message streams in the **jsonmessenger** component the following command might be used:

```
./nox_core -i ptcp:6633 jsonmessenger=tcpport=11222,sslport=0
```

An important configuration parameter, is the TCP port of the secure connection with network switches. It should be set to the same value in both the OFNIC GEi and in the network switches. The default value is:

```
ptcp:6633
```

It should be noted that OFNIC GEi contains several components each of them with a specific functionality, some of them not intrinsically related with OFNIC GEi. The user can explore all the arguments needed for each component in its header file documentation. The configuration needed for the components used in this guide are explained in the section *Running*.

## 5.2.6 Running

Once the Prerequisites, Getting source code and Compilation instructions have been completed, OFNIC GEi is ready to start running. With a terminal locate in the folder

```
cd path/to/ofnic
```

The configuration in this case consists of running the following components: *discoveryws*, *snmp\_stats\_ws*, *bod\_routing\_ws*, "db\_manager", and "acl". All of these components do not need additional arguments to be configured. The command reported below starts the OFNIC controller:

```
sudo make start
```

In order to stop the execution of OFNIC, the following command shall be used:

```
sudo make stop
```

## 5.3 OFNIC GUI

The OFNIC GEi provides a Graphical User Interface (GUI) based on web technologies, which is a web page with javascript code that communicates with the RESTful interface of the GEi. Next sections explain how to install the OFNIC GUI.

### 5.3.1 Software and Hardware environment

The hardware requirements of the OFNIC GUI match the hardware requirements for the installation of the OFNIC GEi abovementioned, please refer to that section of this document for details. The OFNIC is a web based GUI so a web server is needed for its deployment. The developers have chosen Apache HTTP Server running in a Linux-based machine, to deploy the GUI. Any other combination of Operating System - web server would be fine.

### 5.3.2 Prerequisites

Root privileges are required to install additional software in the hosting machine. The following packages are required for the the deployment of the web server and gui in the hosting machine.

```
sudo apt-get install apache2
sudo apt-get install php5-curl
sudo apt-get install curl
```

It is suggested to run the following command:

```
sudo service apache2 restart
```

### 5.3.3 Getting source code

If a Linux machine is being used, locate with a terminal under the folder:

```
cd /var/www
```

The source code is located in public repository of Github: ofnic\_gui\_R3.3 One can clone (download) the source code with the following terminal command:

```
git clone git@github.com:sanandrea/ofnic_gui_R3.3.git gui
```

Another way of getting the source code is to download it from the FI-WARE forge at the following link:

[https://forge.fi-ware.org/frs/download.php/1408/ofnic\\_gui\\_R3.3-master.zip](https://forge.fi-ware.org/frs/download.php/1408/ofnic_gui_R3.3-master.zip)

### 5.3.4 Running

A web browser is needed. In the location area (bar), type in the local address:

```
http://localhost/gui
```

and then press Enter from the keyboard.

## 5.4 SNMP Sub-Agent

### 5.4.1 Software and Hardware environment

The hardware requirements are a very loose and match those of the lightweight SNMP daemon. It has been tested to run in embedded devices with 64 MB of RAM and processor speed of 480 MHz. The operating system should be any Linux distribution with kernel version at least 2.6.24.

### 5.4.2 Prerequisites

Root privileges are required to install additional software in the hosting machine. The following packages are required for the correct compilation and execution of the SNMP Sub-Agent:

```
sudo apt-get install libsnmp-dev
```

### 5.4.3 Getting source code

The source code is located in FI-WARE forge, and can be downloaded at the following link:

[https://forge.fi-ware.org/frs/download.php/889/SNMP\\_Sub-Agent.tar.gz](https://forge.fi-ware.org/frs/download.php/889/SNMP_Sub-Agent.tar.gz)

### 5.4.4 Compilation

Using a system shell locate in the SNMP Sub-Agent source code folder. Run the commands step by step:

```
cd snmp_subagent
./autogen.sh
./configure
make
```

### 5.4.5 Configuration

SNMP Sub-Agent needs the SNMP Master agent in order to run properly. To configure the SNMPd acting as Master Agent and SNMP Sub-Agent the following configuration steps are required:

- If SNMPd is not installed install it with the following command:

```
sudo apt-get install snmpd
```

- Edit the contents of the text file (root privileges are required):

```
/etc/snmp/snmpd.conf
```

The following lines should be added:

```
rocommunity public
syslocation "Switch_Number"
com2sec readonly default public
master agentx
```

- Restart SNMPd with the following command:

```
sudo /etc/init.d/snmpd restart
```

## 5.4.6 Running

Once the Prerequisites, Getting source code, Configuration and Compilation instructions have been completed, SNMP Sub-Agent is ready to start running. With a terminal locate in the `snmp_subagent` folder. Run the following commands step-by-step:

```
cd src
sudo ./ethtool-snmpd -d
```

## 5.5 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 5.5.1 End to End testing

This is basically quick testing to check that everything is up and running.

1. Launch ofnic with the command:

```
sudo make start
```

2. To verify that the OFNIC GEi is loaded correctly it should display `bootstrap complete` in the terminal on which it was launched.

```
00093|openflow|DBG:Passive tcp interface bound to port 6633
00094|nox|INFO:nox bootstrap complete
```

3. Network nodes side, give the following command to verify that the SNMP Sub-Agent is loaded successfully:

```
sudo ./ethtool-snmpd -d
```

it should display the following log message:

```
Initializing Net-SNMP subagent
Initializing ethtoolStatTable
NET-SNMP version 5.4.3 AgentX subagent connected
Initialization complete, SNMP Sub-Agent ready...
```

4. After this two checks have been done, the GEi should be up and ready. To test that is actually running a simple check can be done from the browser. Going with a normal internet browser application to the following address:

```
https://localhost/netic.v1/doc
```

should display the list of all the loaded components of OFNIC GEi. If there are problems in displaying the documentation page some installing step has not been carried out correctly.

### 5.5.2 List of Running Processes

In order to list running processes on a Linux distribution one can use `ps aux` command. In order to get more filtered results one can use this more articulated command:

```
ps aux | grep "Name_of_process"
```

In the machine that hosts the OFNIC GEi the `nox_core` process is required to be active.

In the network nodes:

- `snmpd` is the SNMP Master daemon which is being polled by the OFNIC GEi. SNMPd can delegate the collection of certain information of MIB to Sub-Agents following the Agentx protocol.
- `ethtool-snmpd` is the SNMP Sub-Agent which is responsible for the branch of the MIB interesting network statistics.

So in the network nodes



USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START
TIME	COMMAND							

```
snmp 30957 0.0 0.2 9968 3772 ? S 14:10 0:00 /usr/sbin/snmpd
```

```
root 1903 0.2 0.1 4484 1492 pts/0 S+ 14:43 0:00 sudo ./ethtool-snmpd -d
```

### 5.5.3 Network interfaces Up & Open

The OFNIC GEi listens to the ports 6633, 8080 and 443, with the following command you can verify it:

```
netstat -lnptu | grep nox
```

Command output:

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
PID/Program name					
tcp	0	0	0.0.0.0:6633	0.0.0.0:*	
LISTEN	1941/lt-nox_core				
tcp	0	0	0.0.0.0:8080	0.0.0.0:*	
LISTEN	1941/lt-nox_core				
tcp	0	0	0.0.0.0:443	0.0.0.0:*	
LISTEN	1941/lt-nox_core				
tcp	0	0	127.0.0.1:443	127.0.0.1:46946	
ESTABLISHED	1941/lt-nox_core				

All incoming connections to TCP port 6633 are used for the Openflow Protocol messages. Local host TCP port 443 is used by the Web Server and the Web Services exposed by OFNIC GEi.

Network nodes side the Master SNMP daemon listen on UDP port 161:

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
PID/Program name					
udp	0	0	0.0.0.0:161	0.0.0.0:*	
30957/snmpd					

Also in the active domain Unix socket should be present the following entries:

Proto	RefCnt	Flags	Type	State	I-Node	PID/Program
name	Path					

unix	3	[ ]	STREAM	CONNECTED	83287	30957/snmpd
/var/agentx/master						
unix	3	[ ]	STREAM	CONNECTED	83286	2134/ethtool-
snmpd						

#### 5.5.4 Databases

OFNIC manages the access control list of the REST interface in a mysql database. Verify (e.g. with phpmyadmin) that `openflow_users` is present in the mysql server.

## 5.6 Diagnosis Procedures

OFNIC GEi logs to the stdout on the terminal on which it was launched. Some critical modules log also on file located in `.../ofnic/build/src/OFNIC.log`

### 5.6.1 Resource availability

- The required RAM depends on many factors such as network topology, number of flows in the network, frequency of the statistics updates, frequency of web service requests, etc. Generally RAM size varies from 20 MB to 100 MB.
- Usually the disk size required during run time is negligible.
- More processor cycles are required when analysing SNMP statistics from the network nodes.

### 5.6.2 Remote Service Access

User can verify the correct execution of the OFNIC GEi, by directing the browser (all types are supported) to the following page:

<https://localhost/netic.v1/doc>

which should display a list of commands that can be sent to the interface. Note that if the browser app is not on the same machine of the GE, the remote IP address of the GEi can be used.

### 5.6.3 Resource consumption

The resource consumption is highly dependent on the number of network events processed. The minimum amount of RAM is nearly 10 MB so eventually in any lower amount of RAM means that the application did not load properly. Under normal working conditions RAM size reaches the order of 100 MB so values of greater orders mean that there is some malfunctioning. CPU percentage ranges and is highly dependent on the processor speed. However it should be noted that at idle state the OFNIC GEi processor consumption can be even lower than 0.1%.

#### 5.6.4 I/O flows

Port 443 for webservices and port 6633 for the communication with the network nodes.

### 5.7 References

[1]	Nox Openflow Controller: <a href="http://www.noxrepo.org/">http://www.noxrepo.org/</a>
[2]	Simple Network Management Protocol (SNMP): <a href="http://www.ietf.org/rfc/rfc1157.txt">http://www.ietf.org/rfc/rfc1157.txt</a>
[3]	AgentX protocol: <a href="http://en.wikipedia.org/wiki/Agent_Extensibility_Protocol">http://en.wikipedia.org/wiki/Agent_Extensibility_Protocol</a>

## 6 NetIC GE - Altoclient - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 6.1 Introduction

The *Altoclient* instantiation of the 'Network Information and Control' Generic Enabler is intended to allow easy retrieval of selected information about a network from an ALTO server attached to this network. The *Altoclient* instantiation implements the 'Topology Information Module' described in more detail in the [NetIC Open Specification](#). *Altoclient* provides easy-to-use access to network information at the level of detail being useful to applications (e. g., for application-level load balancing).

*Altoclient* is delivered as a C function library (source code). Offering the *Altoclient* as function library facilitates the integration of *Altoclient* functionality in applications. The function library offers a set of specific function calls solving specific network information needs while hiding most of the complexity of the underlying ALTO protocol, which is used to retrieve the requested information from the network of interest. The initial implementation of the library is developed for Linux/Unix-based applications.

*Altoclient* library and related test programs can be downloaded [here](#).

When using *Altoclient* to obtain information from a real network, *Altoclient* requires a working ALTO server (IETF ALTO protocol draft version 13) being able to provide information about that specific network.

### 6.2 Prerequisites for Software and Hardware

*Altoclient* is designed for use in Linux operating systems. Up to now, it was tested with Ubuntu 12.04 LTS. For proper operation, the *Altoclient* must have access to a specified ALTO server. Especially, if a port other than 80 is specified for the ALTO server, this port must not be blocked by firewalls. If the ALTO server is not specified by its IP address, access to a DNS server must be assured. The *Altoclient* directly relies on the [Jansson](#) library (version 2.4 or newer) and the [Curl](#) library (version 7.22.0 or newer). Both libraries may depend on further libraries. During operation the *Altoclient* allocates memory for storing the answers of the ALTO servers. The size of these buffers can be specified by the HTTP\_BUF\_SIZE option. However, a minimum of 400kB of RAM is required for proper operation.

### 6.3 NetIC - Altoclient Generic Enabler Instantiation Implementation

#### 6.3.1 Installing the *Altoclient*

The install procedure relies on the autotool set. To install the *Altoclient* choose an appropriate directory then run:

```
tar -xvzf libalto-2.0.0.tar.gz
cd libalto-2.0.0
```

```
./configure  
make  
make install
```

The last command installs the library on the target system and makes the basic test program 'alto\_test\_easy' available from command line. The command 'alto\_test\_easy -h' provides further information.

### 6.3.2 Compiling and Linking

The main declarations can be found in `alto.h`, further utilities are declared in `alto_util.h`. So the source files of programs calling *altolib* functions must contain

```
#include <alto.h>  
  
#include <alto_util.h>
```

Linking must be performed using the option `-lalto`.

All symbols defined in the *altolib* have the prefix 'alto\_', all defines start with 'ALTO\_'. To avoid naming interferences, users should avoid using names and defines starting with these prefixes as this may result in conflicts during compiling and linking.

### 6.3.3 Test Programs

In the directory *contrib* there are five test programs which can be used for verification of the user's software and hardware environment and as a starting point for the user's own developments. More details on usage can be found in the readme file in the same directory.

- `alto_test_easy`: basic test program to verify the ALTO operation
- `alto_test_advanced`: test program with increased access to control parameters
- `alto_test_expert`: test program demonstrating all available features including error reporting
- `alto_epps_easy`: basic test program to verify the ALTO endpoint property service
- `alto_epps_advanced`: test program to show all features of the client handling of the ALTO endpoint property service

## 6.4 Sanity Check Procedures

### 6.4.1 End to End testing

- During installation the test program `alto_test_easy` (see section [Test Programs](#) above) is installed and is available from the command line. `alto_test_easy -h` provides further information. If no argument is given, the public ALTO server <http://alto.alcatel-lucent.com:8000/directory> is used.

- When starting the program with no arguments, you should get the following result:

```
-----  
Start of Alto Client Test  
See file Debug_TestAltoHeader.txt for further information.  
Debug_HTTPMessages.txt contains messages from the HTTP Protocol  
-----  
Entering alto_init  
This Alto directory is used:  
http://alto.alcatel-lucent.com:8000/directory  
-----  
-----  
The following proxy parameters are used:  
  
Proxy:      (null)  
NoProxy:   (null)  
  
-----  
This timeout is used 10  
-----  
Alto Client successfully initialized  
-----  
-----  
The following proxy parameters are used:  
  
Proxy:      (null)  
NoProxy:   (null)
```

```
-----
This timeout is used 10
-----
-----
```

```
Resulting Cost Matrix
```

No	Node	Cost
0	135.0.0.0	8
1	123.45.24.2	5.1
2	15.0.0.0	15

```
-----
This endpoint was selected: 123.45.24.2
-----
```

- If the above sanity check fails, a system operator should check for the reachability of the ALTO server.
  - This can be done by

```
echo -n "Get <Path to ALTO directory> HTTP/1.1" | nc -X connect -x
<proxy>:<proxy port>
<ALTO server URL> <port>
```

- For example type the following to access the Alcatel-Lucent ALTO test server (assuming that there is no proxy in the connection)

```
echo -n "GET /directory HTTP/1.1" | nc alto.alcatel-lucent.com 8000
```

- The server will return the directory in JSON format.

## 6.4.2 List of Running Processes

The alto client does not use multi processing, so only the application process, which uses the library, will be visible with the linux command *ps*.

### 6.4.3 Network interfaces Up & Open

The open network ports depend on the specified ALTO server; the ALTO protocol as such works with only port 80 being open.

### 6.4.4 Databases

No database used.

## 6.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GEi. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

### 6.5.1 Resource availability

*Altoclient* needs about 400kB RAM. The size of the client test program is 270 kB.

### 6.5.2 Remote Service Access

No remote service access provided.

### 6.5.3 Resource consumption

RAM Consumption min. 400 kB, maximum depends on configuration File system sizes and storage requirements 270kB, size of easy test program

### 6.5.4 I/O flows

- For proper operation, *Altoclient* must have access to a specified ALTO server (HTTP port 80). Especially, if a port other than 80 is specified for the ALTO server, this port must not be blocked by firewalls. If the ALTO server is not specified by its IP address, access to a DNS server must be assured.
- Interface to application by C function calls.

## 6.6 References

- [NetIC Open Specification](#)
- [NetIC Library API Specification](#)
- more information about ALTO can be found [here](#)



## 7 NetIC GE - VNP - Installation and Administration Guide

The NetIC-VNP Generic Enabler will be offered as a service by the GE owner. Besides, the software release of the GE and the accompanying Installation and Administration guide are of PP dissemination level<sup>[1]</sup>.

In accordance with the privacy restrictions requested by GE owner, both the software (binaries) and the Installation and Administration Guides were not reviewed by the FI-WARE coordinator or the Work Package leader and the responsibility for the quality and appropriateness of this part of the deliverables remain solely with the partner who is the GE owner.

The software (binaries) and the Installation and Administration Guides are accessible to the EC for reviewing purposes on a remote server located at 93.189.118.151 via ssh with user: EC and password to be disclosed upon explicit request to the project coordinator. The partner who is the GE owner will ultimately provide this access.

## 8 NetIC GE - VNEIC - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 8.1 Introduction

The Virtual Network Element Information and Control (VNEIC) is a [NetIC generic enabler](#) implementation to virtualise an ALU-1850TSS network element by its native management interface based on both TL1 and SNMP protocols.

The VNEIC may act as a proxy for an ALU-1850TSS allowing offline standing alone analysis and interactions; in case further study is requiring to remotely access an actual ALU-1850TSS, please refer to the contacts in the [FI-WARE Catalogue page](#) of this GEi.

Listed below more VNEIC references you may also like to follow

- [NetIC Open Specification](#)
- [Network Information and Control - User and Programmers Guide](#)
- [Network Information and Control - Unit Testing Plan](#)
- [A Common Open Interface to Programmatically Control and Supervise Open Networks in the Future Internet](#)
- [1850 Transport Service Switch](#)
- [Ruby on Rails Guide \(v3.2.13\)](#)

### 8.2 VNEIC

#### 8.2.1 Software and Hardware environment

The reference operating environment for the VNEIC application is the Debian Linux distribution but in principle it can be run on any other environment capable to support ruby on rails. Details of the reference operating environment are reported below

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:      Debian
Description:        Debian GNU/Linux 7.3 (wheezy)
Release:             7.3
Codename:            wheezy
$ uname -a
```

```
Linux debian 3.2.0-4-486 #1 Debian 3.2.51-1 i686 GNU/Linux
```

The operating environment described above is enough for running this GEi as a standing alone application acting as a proxy for a network of real ALU-1850TSS as for example in the scenario of its integration in the FI-WARE ecosystem. However to have it fully operating a back-end of actual ALU-1850TSS to be accessed via telnet or ssh sessions is also required and a test-bed of both real and emulated ALU 1850TSS would be available at need: please refer to the contacts in the [FI-WARE Catalogue page](#) of this GEi if this is the case.

## 8.2.2 Prerequisites

Install your personal ruby on rails framework

```
$ curl -sSL https://get.rvm.io | bash -s stable --rails
```

and set-up the environment to make it available on the shell where the GEi will run

```
$ source $HOME/.rvm/scripts/rvm
```

## 8.2.3 Getting source code

SW releases are distributed as compressed archives: please refer to Network Information and Control - VNEIC [downloads](#) area of FI-WARE Catalogue for available releases.

Having downloaded the archive, corresponding to a VNEIC SW release of your choice, unpack it with

```
$ tar xzf I2ND-VNEIC-3.3.1.tgz
```

## 8.2.4 Compilation

VNEIC application consists of a set of scripts and do not require to get compiled.

## 8.2.5 Configuration

VNEIC application does not require to be configured; however if any customization is required please refer to the contacts in the [FI-WARE Catalogue page](#)

## 8.2.6 Running

Enter the directory created by unpacking the VNEIC application tarball as described above

```
$ cd VNEICApp
```

the application have to be reset and provided with a clean database by the following commands

```
$ rm db/development.sqlite3  
$ rake db:migrate
```

or alternatively if you like to perform a database migration from a past release just overwrite the new DB image with the old one

```
$ cp <path to my previous vneic>/db/development.sqlite3  
db/development.sqlite3  
  
$ rake db:migrate
```

and start the application with the command

```
$ rails server
```

## 8.3 Sanity check procedures

### 8.3.1 End to End testing

The smoke test for the VNEIC application makes sure that you have your software configured correctly enough to serve a web page. To see your application in action, open a browser window and navigate to <http://localhost:3000>. You should see VNEIC' default welcome page.

### 8.3.2 List of Running Processes

The VNEIC application consists of the single process run as described above.

### 8.3.3 Network interfaces Up & Open

The VNEIC application includes an HTTP server in listening state at TCP port 3000.

### 8.3.4 Databases

The VNEIC application persistency is based on the SQLite database in the file

```
db/development.sqlite3
```

## 8.4 Diagnosis Procedures

Diagnostic about the VNEIC application is provided on both standard output and error of the terminal used to start the application itself. More detailed log can also be found in directory

```
log
```

### 8.4.1 Resource availability

VNEIC application resource requirements should be negligible with respect to a smoothly running reference operating environment described above; if that is not the case consider at least 10MB of free disk space for installing the sources and 20MB of free memory to allow it running.

#### 8.4.2 Remote Service Access

A fully operating VNEIC application is providing a WEB service whose welcome page can be accessed at

```
http://localhost:3000
```

#### 8.4.3 Resource consumption

Both memory and cpu consumption have to be considered negligible with respect to the reference operating environment described above; disk usage too but for the SQLite database

```
db/development.sqlite3
```

whose size depends on the actual network resources configuration.

#### 8.4.4 I/O flows

VNEIC application provides its [NETIC API](#) interface by an HTTP server on port 3000.

## 9 S3C GE - EPC OTT API - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 9.1 EPC-OTT API Enabler Installation

#### Important Note

The EPC-OTT API Enabler is requiring connection to an operator network instance, such as a testbed based on the Fraunhofer FOKUS OpenEPC. In order to be able to test and use the benefits of the EPC-OTT API Enabler at least one device has to be provisioned in the network and it has to be connected to the operator network.

#### 9.1.1 Step 1: Installation of ubuntu 12.04

We suppose that we have a x86 PC of a WM with an installed and running ubuntu LTS 12.04 version. For installing ubuntu 12.04, please refer to <http://www.ubuntu.com>. Choose any kind of machine name and install at least one user account. We call the PC “PCtest” and the user account “user” in the command lines we give in the next chapter.

#### 9.1.2 Step 2: Installation of S3C EPC-OTT API Enabler

##### 9.1.2.1 *Step 2.1: Installation of Jetty*

The Jetty server has to be installed prior to the EPC-OTT-API component, as it functions as a web server on top.

- The Jetty distribution should be downloaded from <http://download.eclipse.org/jetty/stable-9/dist/>
- The Jetty server should be installed as described in <http://degreesofzero.com/article/19>

##### 9.1.2.2 *Step 2.2: Installation of Maven*

Maven can be installed using the package manager:

```
sudo apt-get install maven2
```

Another option is to visit the web site of the maven project <http://maven.apache.org>, download the binary packages and follow the installation instructions at the documentation pages of the project.

### 9.1.2.3 **Step 2.3: Installation of EPC-OTT API Enabler**

1. Download the I2ND-S3C-EPC-OTT-API.zip file from [FI-WARE repository forge](#) into a directory of your choice.
2. Uncompress the I2ND-S3C-EPC-OTT-API.zip file unzip I2ND-S3C-EPC-OTT-API.zip
3. Copy the file "I2ND-S3C-EPC-OTT-API/ngsi.applicationDrivenQoS/target/ngsi.applicationDrivenQoS.war" to the /opt/jetty/webapps folder
4. Copy the rxconfig.xml and the s14.properties to the /opt/jetty/config/
5. Edit the two configuration files according to the underneath IP configuration

## 9.1.3 Step 3: Configuration

### 9.1.3.1 **Step 3.1: Web Server Configuration**

The web server is listening on all the interfaces of the machine on port 8080. The EPC-OTT API requests can be transmitted to any of the addresses.

### 9.1.3.2 **Step 3.2: QoS Connectivity Configuration**

For being able to exchange QoS related information between the operator network and the EPC-OTT API component, a network connection has to be established via Diameter protocol between the EPC-OTT API component and the network PCRF component. For this both ends of the connection have to be established.

#### 9.1.3.2.1 **Core Network PCRF Configuration**

1. The PCRF component located in the operator core network has to be reachable from the EPC-OTT API component. For this an IP address has to be configured to the PCRF, here named <<PCRF\_IP>>.
2. The PCRF Diameter peer has to be configured to listen on the IP reachable from the EPC-OTT API component. For example, in case of Fraunhofer FOKUS OpenEPC PCRF, a new XML tag (mark in bold) has to be added in the *pcrf.xml* configuration file such as:

```
<Acceptor port="3868" bind="PCRF_IP"/>
```

3. The PCRF has to be configured with a new Diameter peer, For example, in case of Fraunhofer FOKUS OpenEPC PCRF, the following line (mark in bold) has to be added in the *pcrf.xml* configuration file such as:

```
<Peer FQDN="EPC-OTT-API_IP" Realm="epc.mnc001.mcc001.3gppnetwork.org"
port="3868" />' '
```

### 9.1.3.2.2 EPC-OTT-API Component Configuration

1. The EPC-OTT-API component has to be reachable from the EPC PCRF component. For this an IP address has to be configured to the EPC-OTT-API component, here named <<EPC-OTT-API\_IP>>.
2. The EPC-OTT-API Component Diameter peer has to be configured to listen on the IP reachable from the EPC PCRF component. The file `/opt/jetty/config/rxconfig.xml` has to contain the following information related to the Diameter peer and the acceptor interface:

```
<Peer FQDN="PCRF_IP" Realm="epc.mnc001.mcc001.3gppnetwork.org"
port="3868" />
<DiameterPeer FQDN="EPC-OTT-API_IP"
Realm="epc.mnc001.mcc001.3gppnetwork.org" ..../>
<Acceptor port="3868" bind="IP_vm"/>
```

### 9.1.3.3 Step 3.3: Access Network Selection Configuration

For being able to exchange Access Network Selection related information between the operator network and the EPC-OTT API component, a network connection has to be established between the EPC-OTT API component and the network ANDSF component. For this both ends of the connection have to be established.

#### 9.1.3.3.1 Core Network ANDSF Configuration

1. The ANDSF component located in the operator core network has to be reachable from the EPC-OTT API component. For this an IP address has to be configured to the ANDSF, here named <<ANDSF\_IP>>.
2. The ANDSF has to be configured to listen on the IP reachable from the EPC-OTT API component. For example, in case of Fraunhofer FOKUS OpenEPC ANDSF, a new XML tag (mark in bold) has to be added in the `andsf.xml` configuration file such as:

```
<Acceptor type="tcp" port="10000" bind="ANDSF_IP"/>
```

#### 9.1.3.3.2 EPC-OTT-API Component Configuration

1. The EPC-OTT-API component has to be reachable from the EPC ANDSF component. For this an IP address has to be configured to the EPC-OTT-API component, here named <<EPC-OTT-API\_IP>>.
2. To reach the ANDSF, the EPC-OTT-API Component has to be configured with the following information in the `/opt/jetty/config/s14.properties`

```
"ANDSF_URL", "ANDSF_IP"
```



```
"ANDSF_PORT", "10000"  
"LTE", "4"  
"WiFi", "3"  
"POLICYID_UMTS", "6"
```

#### 9.1.4 Step 4: Running the EPC-OTT-API Application

In order to start the application the following commands should be executed:

```
cd /opt/jetty  
/opt/java/jre/bin/java -jar start.jar
```

#### 9.1.5 Step 5: Test Client

The Test Client application is a small size application which enable the testing of the basic functionality of the EPC-OTT-API component, as well as of the functionality supported by the core network.

##### 9.1.5.1 **Step 5.1: Installing the Test Client**

1. Pre-requisites: the Eclipse has to be installed <http://www.eclipse.org/>
2. From the "I2ND-S3C-EPC-OTT-API.zip" file, the projects "ngsi.applicationDrivenQoS" and "ngsi.testClient" have to be imported in Eclipse.
3. The maven dependencies have to be installed. From the "I2ND-S3C-EPC-OTT-API.zip" file, from the interior "mvn\_repository\_s3c\_epc\_ott\_api.zip" file, the "repository" directory has to be copied to the local maven repository. The default path is "/home/<user>/.m2/repository/"
4. In the project "ngsi.testClient", in the file "Util.java", the server address of the EPC-OTT-API component has to be configured. In the class Util, the following parameters have to be configured:

```
private String brokerhost = "EPC-OTT-API IP address as String";  
private String port = "8080";
```

##### 9.1.5.2 **Step 5.2: Running the Test Client**

1. The ngsi.testClient (Main.java) should be run as Java Application from Eclipse.
2. A console will appear giving the possibility to test the different functions:

- 2 - requesting specific QoS rules to be installed on the data path  
(should be the first command tested)
- 3 - deleting the specific QoS rules installed
- 4 - modifying the specific QoS rules installed
- 5 - printing the QoS status for a specific subscriber
- 6 - indicating that an access network selection to LTE is required
- 7 - indicating that an access network selection to WiFi is required
- 8 - retrieving the access networks available in the device vicinity

## 9.2 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 9.2.1 End to End testing

1. To see if the jetty server is running, you have to open in a browser of your choice the following web page "<http://localhost:8080/>"
2. An operator network should be attached to the EPC-OTT API component, including and EPC PCRF and an EPC ANDSF functions. Additionally, at least one device should be connected to the operator network.
3. To see if the REST interface is up, the "Test Client" has to be run, as described in the Step 7 of this section. The success of this procedure, allows the testing of the availability of the Diameter interface to the PCRF, as well as of the interface to the ANDSF.

### 9.2.2 List of Running Processes

The EPC-OTT-API jetty process has to be running. It can be visible while running the `ps -ef | grep "start.jar"`

### 9.2.3 Network interfaces Up & Open

With the command "`netstat -tulpen`" the Diameter and the ANDSF communication ports should be visible. Additionally, the 8080 port of the webserver should be opened on all the available IP addresses.

### 9.2.4 Databases

N/A

## 9.3 Diagnosis Procedures

For Diagnosis of the EPC-OTT-API component, a test client was developed enabling a minimal sanity check of the overall functionality.

### 9.3.1 Resource availability

- For the specific EPC-OTT API component, it is recommended to have at least 500MB RAM allocated and around 30MB hard-disk space for the installation
- Additional resources are required for the Jetty server and the Apache Maven summing up to 1GB RAM and 1GB hard-disk space for the installation.

### 9.3.2 Remote Service Access

The EPC-OTT-API component is accessible at all its IP addresses on port 8080.

Additionally, for a correct functioning it requires connectivity to an EPC operator core network as described in the configuration part of this document.

### 9.3.3 Resource consumption

The resource consumption is mainly dependent on the usage of the GE. In idle state, the resource consumption is determined by the resource consumption of the Jetty server, which is typically around 25 MB of RAM and 0.1% CPU load (of course depending on the power of the CPU).

### 9.3.4 I/O flows

For the communication between applications and the EPC-OTT-API component http data traffic is expected on port 8080.

For the communication with the network, TCP connections will be established with the PCRF and the ANDSF. The connection to the PCRF is containing Diameter protocol messages.

## 10 S3C GE - Network Identity Management - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 10.1 S3C GE - Network Identity Management Installation

#### 10.1.1 Important Note

The Network Identity Management (NIM) is requiring connection to an operator network instance, such as a testbed based on the Fraunhofer FOKUS OpenIMS Core or another IMS core network which are able to run Application Servers over the ISC interface. In order to be able to test and use the benefits of the NIM at least one device has to be provisioned in the network and it has to be connected to the operator network. Next to this, a service profile for the NIM is needed to be configured in your Home Subscriber Server as well as allowing Services to SIP based SUBSCRIBE and receive NOTIFY messages with the reg-event context. Also a user profile which allows the application server (AS) also be handled as a subscriber is needed to be configured in the HSS. The name of this subscriber can be configured in the nim.preferences file.

#### 10.1.2 Step 1: Installation of operating system

The NIM was tested under Ubuntu Linux 12.04 and therefore the suggested Linux distribution for the enablement of S3C.NIM.

You can download the Ubuntu Operating System via the Website <http://www.ubuntu.com/>.

#### 10.1.3 Step 2: Installation of NIM

##### 10.1.3.1 *Pre-required Package*

The NIM is written in Java programming language and needs a runtime environment. The NIM was tested with OpenJDK as well as Oracle Java. Other pre-required packages will be provided via the downloadable archive.

##### 10.1.3.2 *Installation*

The [I2ND S3C Network Identity Management R.2.3](#) archive can be downloaded from the FI-WARE [download area](#). It includes an installation shell script. This script is needed to run as *root*:

```
$ cd /path/to/extracted/nim/  
$ sh install.sh
```

Make sure that your working directory where the install script is located.

All required libraries are copied to

```
/opt/nim/
```

Next to this, an initial script for automatically restarting will be copied to

```
/etc/init.d/
```

This allows automatically starting after reboot etc.

### 10.1.3.3 **Configuration**

Under following folder

```
/opt/nim/config/
```

you will find the nim.properties file, which is needed to be configured

```
# nim properties #

# sip host ip address
http_host=192.168.7.10

# http RESTful Interface Port
http_port=8080

# sip host ip address
sip_host=192.168.7.10

# SIP Port
sip_port=9060

#the transport protocol, usually udp
sip_transport=udp

#the contact name in the via fields etc.
ims_contact_name=idm
```

```
#the domain in which the identity manager is running
ims_domain=ims.testbed.test

#authentication against IMS Network of the "AS"
auth_user=as
auth_domain=ims.testbed.test
auth_password=as_password

#proxy settings
domain_proxy=scscf.ims.testbed.test
domain_proxy_port=6060
domain_proxy_transport=udp

#subscriber task, time value in seconds
subscriber_task_delay=0
subscriber_task_period=30
subscribe_expires=30
send_options_request=false;

#one api resources link
oneapi_resources=oneapi
```

#### 10.1.3.4 *Final steps*

After finishing the configuration, an initial start of the NIM can be performed via

```
/etc/init.d/nim.sh start
```

Other parameters are "restart" and "stop".

## 10.2 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 10.2.1 End to End testing

To check if the web component is running, a simple HTTP GET request to

```
http://<ipaddress>:<port>/<oneapiresource>/status
```

will give short information about the currently running instance.

### 10.2.2 List of Running Processes

Since the NIM is an entity which is java based, it should be one running process within the machine. A simple check with

```
ps aux | grep java
```

should show the running networkidentitymanagement.jar process. You can access the console via

```
screen -r nim
```

and exit safely via CTRL + A + D.

### 10.2.3 Network interfaces Up & Open

The SIP communication is UDP based and needs the configured ports opened. The HTTP communication is TCP based and is also needed to be open.

### 10.2.4 Databases

N/A

## 10.3 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

### 10.3.1 Resource availability

The function is set to work on a

## Future Internet Core Platform

- $\geq$  1GB RAM VM (for OS and application),
- $\geq$  2 GHz CPU, and
- 10GB hard disc drive(for OS and application).

### 10.3.2 Remote Service Access

The NIM component is accessible at its IP addresses on port 8080. Additionally, for a correct functioning it requires connectivity to an operator core network as described in the configuration part of this document.

### 10.3.3 Resource consumption

The resource consumption is mainly dependent on the usage of the GE. In idle state, the resource consumption is determined by the resource consumption of the SIP Application instance and the integrated HTTP/REST Service, which is typically around 256 MB of RAM and less than 5% CPU load (of course depending on the power of the CPU). However, if the usage is increasing, the CPU and memory consumption is also increasing; but no more hard-disk space is required.

### 10.3.4 I/O flows

For the communication between applications and the NIM component http data traffic is expected on port 8080, yet not secured. For the communication with the network, UDP datagrams will be transmitted with the SCSCF of the IMS network. The transmitted protocol is SIP.



# 11 S3C GE - Network Positioning Enabler - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

## 11.1 S3C GE - Positioning Enabler - Installation and Administration Guide

### 11.1.1 Important Note

The Positioning Enabler (PE) platform does only work with a mobile operator core network, e.g. the Evolved Packet Core (EPC), and hence requires an interface to the EPC. Within FI-WARE, the PE maintains an interface to the OpenEPC core network testbed provided by Fraunhofer FOKUS. For illustrating the functionality of the PE it is required that the OpenEPC has at least two radio access networks up and running. Furthermore, at least one user equipment (UE) has to be provisioned in the network as well as connected to the core network via a 3GPP or non-3GPP access network supported by the OpenEPC.

### 11.1.2 Installation of Ubuntu 12.04

We suppose that we have a x86 PC with an installed and running Ubuntu LTS 12.04 version. For installing Ubuntu 12.04, please refer to <http://www.ubuntu.com>. Choose any kind of machine name and install at least one user account. We call the PC “PCtest” and the user account “user” in the commands described in the following sections.

### 11.1.3 Installation of the Positioning Enabler

The PE is written in Java and it needs a runtime environment. The modules used in this project are J2EE, Spring 3.2.2 and Hibernate 3.6. The module was tested with Oracle Java 1.7.

#### 11.1.3.1 *Before you start*

- Ensure that there is an active user equipment connected to an access network of the openEPC core network. (For the provided client app to work properly this UE should have the IMSI “001011234567890”)
- Download the “Network\_Positioning\_Enabler.zip” from the FI-WARE repository forge at [section I2ND-S3C](#) (Release 3.2).
- Ensure that you have permissions to use sudo on the machine you want to install the software upon.

#### 11.1.3.2 *Installation of OpenVPN*

As stated earlier, the PE does only work in conjunction with the OpenEPC core network and hence a connection needs to be maintained between the PE and the core network. This connection is realized via

VPN. You need a running OpenVPN as well as a VPN certificate in order to access the OpenEPC. In order to build this VPN connection, permission certificates are required. For VPN certificates to the OpenEPC testbed, please contact the GEi owner listed in the [S3C catalogue](#).

For installation of the openVPN under Ubuntu, use the following command:

```
sudo apt-get install openvpn
```

For access to the OpenEPC copy the files from the directory “openvpn\_etc\_directory” from the previously downloaded zip file to “/etc/openvpn” for the configuration of the access to the OpenEPC. Furthermore, please insert the file called “openvpn” to “/etc/init.d” for the automatic starting of the vpn.

Start the openVPN with the command

```
sudo service openvpn start
```

### 11.1.3.3 *Installation of Tomcat*

The Tomcat server has to be installed prior to the Positioning Enabler component, as it functions as a web server on top. This version of the PE was tested with Tomcat7.

- The Tomcat distribution should be downloaded from <http://tomcat.apache.org/download-70.cgi>
- The Tomcat server should be installed as described in <http://tomcat.apache.org/tomcat-7.0-doc/setup.html>
- Or it can be installed via the default packaging manager (like “sudo apt-get install tomcat7”).
- Adapt <tomcatroot>/conf/tomcat-users.xml so you can log into the manager app.

### 11.1.3.4 *Installation of MySQL*

MySQL has to be installed prior to the Positioning Enabler component. This version of the PE was tested with MySQL 5.5 and 5.6.

- Install MySQL via the default packaging manager (like “sudo apt-get install mysql-server”)
- Create a database for the PE (name it like “pe\_db”).
- Create a user with all necessary privileges for creating, updating and deleting tables and data.
- Log in as MySQL root user and increment the maximal allowed connections with the following query: “SET GLOBAL max\_connections = 350;”
- After that edit your /etc/my.cnf to uncomment and change “max\_connections” to “350”.

### 11.1.3.5 *Installation of the Positioning Enabler module*

- Open the PositioningEnabler.war file from the previously downloaded S3C-GE-PositioningEnabler.zip, locate the “hibernate.cfg.xml” file under “/WEB-INF/classes/” and fill in \$DATABASEURL, \$DATABASENAME, \$USERNAME and \$PASSWORD.
- Do the same thing in the “config.properties” file under “src/main/java/de/tuberlin/snet/pe/util/netpos/config.properties” and fill in \$DATABASEURL, \$DATABASENAME, \$USERNAME and \$PASSWORD as well as \$ANDSF\_PASSWORD.
- In a browser, open the manager app of the Tomcat under “localhost:8080/manager”. Under the point “WAR file to deploy” click on “Choose file”, find and select the “PositioningEnabler.war” file and click on “Deploy”.
- Now the PE is running and waiting for input.

### 11.1.3.6 *Installation of the Android client*

The Android Client can be found within the “Network\_Positioning\_Enabler.zip”. Download the file at the [FI-WARE repository forge](#) (at section I2ND-S3C - Release 3.3).

#### 11.1.3.6.1 *Pre-requisites*

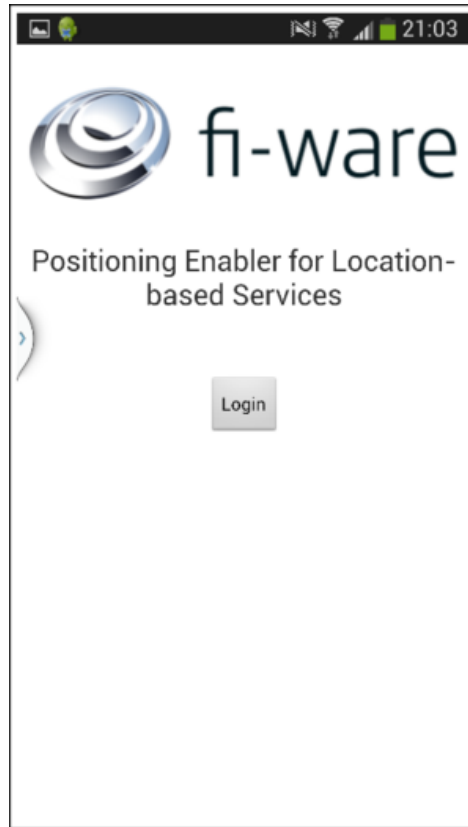
- Eclipse has to be installed (<http://www.eclipse.org/>)
- A Google Mail account
- Android smartphone (at least Android 2.3.3, API v10).

#### 11.1.3.6.2 *Installation*

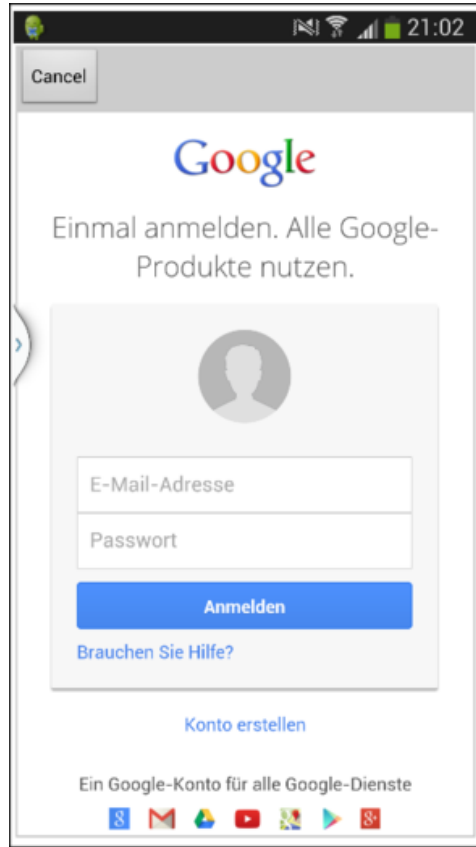
- From the “Network\_Positioning\_Enabler.zip” file, the project “AndroidClient” has to be imported in Eclipse.
- Locate the “strings.xml” in the “res/values/” directory and change the IP of the hosts to your server IP. The “url\_test\_host” is for testing with an emulator on the same machine as the one the Positioning Enabler is installed on. Optional: If you want the Android client to subscribe for a UE with another IMSI (not “001011234567890”) you have to change the userId “userid\_alice” to point to a new userId (this new userId needs to be published to the Positioning Enabler. See section “End to end testing” for more details of publishing a new user/IMSI). The reason lies in the fact that for testing purposes the Positioning Enabler always relates the userId “1337” to the fixed IMSI “001011234567890”.
- Change the string SERVICE\_URI in the class “de.tuberlin.snet.nlocation.client.NetworkPositioningClient” so that it points to your server.
- Build the application and install it on your smartphone.

#### 11.1.3.6.3 *Sample screenshots*

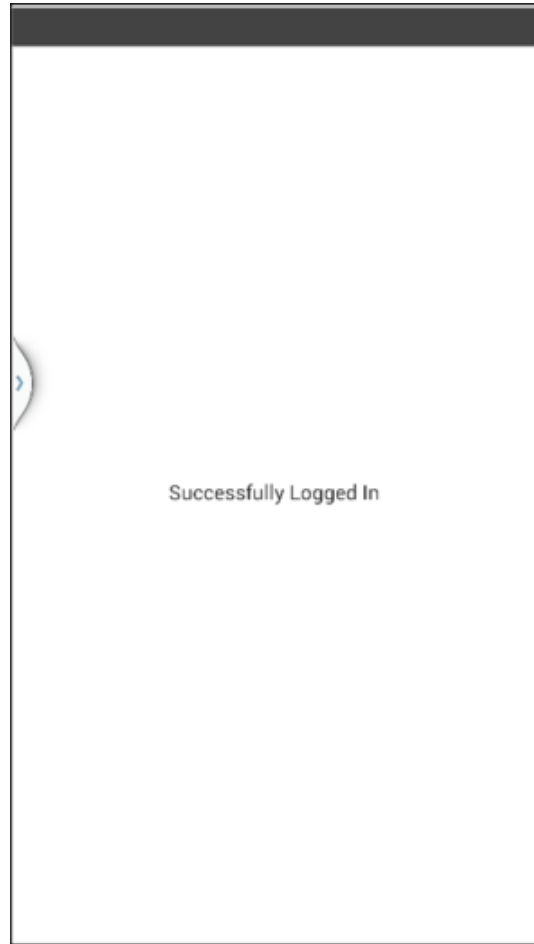
Login:



Resource Description: Start screen of the FI-WARE test app

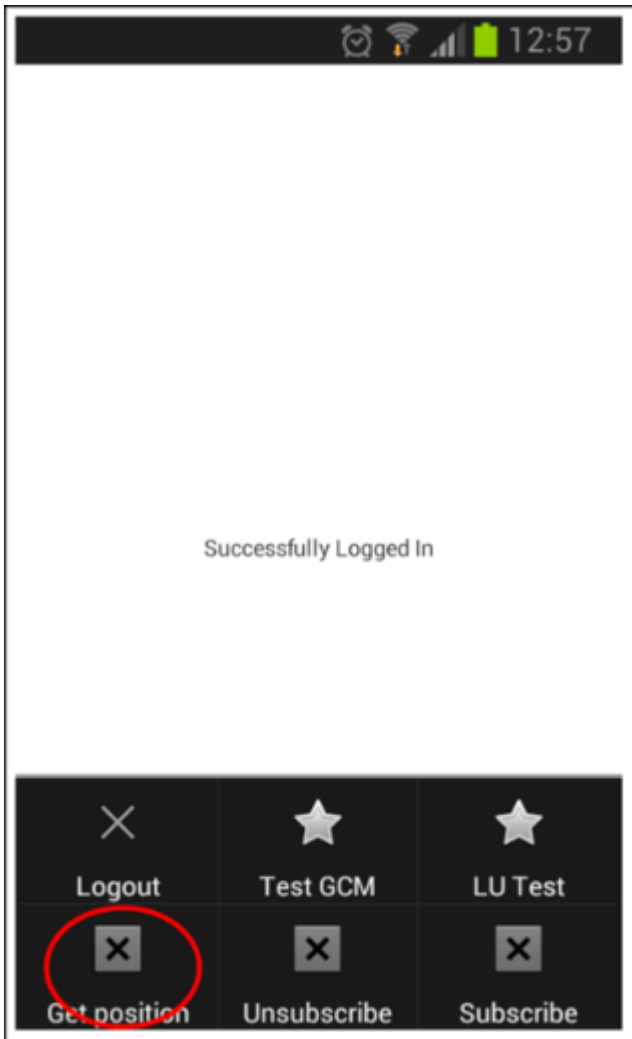


Resource Description: Login screen of the the FI-WARE test app



Resource Description: Home screen of the FI-WAR  
E test app

Position request:



**Resource Description:** Click on 'Get position' in the Menu of the FI-WARE test app

**Resource Description:** The position of the UE the user subscribed to is shown

## 11.2 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 11.2.1 End to End testing

The first thing you should test is the connection to the OpenEPC as well as if there is a UE connected to an access network in the openEPC. The ANDSF is available at the address 192.168.254.31, so you can try to ping the address. You can as well try to build a ssh connection to the OpenEPC (ideal way), with the user "root":

```
user@PCtest:~$ ssh root@192.168.254.31
```

Please contact (Baumgart, Matthias <Matthias.Baumgart@telekom.de>) for getting the password for accessing the EPC. You can also directly test the availability of the ANDSF through a tcp connection; the ANDSF listens at the port 10000:

```
user@PCtest:~$ nc 192.168.254.31 10000
```

To test the connectivity in the DB, do the following:

- Connect to the openepc database with (please contact (Baumgart, Matthias <Matthias.Baumgart@telekom.de>) for getting the password for the ANDSF component of the EPC):

```
user@PCtest:~$ mysql -h 192.168.254.31 -u andsf -p
```

- Select the andsf database:

```
mysql> USE andsf_db
```

- Try to get the UE's location:

```
SELECT location_xml FROM s14_clients WHERE subscription_id_data =  
001011234567890;
```

To see if the PE is running on the Tomcat server, you have to open in a browser of your choice the following web page "<http://localhost:8080/PositioningEnabler>" Under the URL "<http://localhost:8080/PositioningEnabler/service/fiware/notification>" you can create a new Notification. You will get a unique id. You can see the notification in the FiwareNotification table in MySQL using the query:

```
SELECT * FROM fiwarenotification WHERE id = '<notification id>'
```

For testing you can register a user equipment (UE) with use of a RestClient (like Postman for Chrome) sending a POST message with a user id to this URL:

<http://{serverip}:8080/PositioningEnabler/api/user/001011234567890/tracking>





**Resource Description: Home screen of the FI-WARE test app with options shown**

Optional: If you want to test using an UE with another IMSI (not "001011234567890") you have to publish the new IMSI to the Positioning Enabler. This is done by 1. Inserting a new user (UE) with an arbitrary userId except "1337" and 2. Giving the new user (UE) a fixed IMSI. These are the corresponding two SQL statements that need to be done to publish a new user (UE):

```
INSERT INTO `userProfile` (`userId`, `authToken`, `openId`, `status`)
VALUES ('<new_userId>', NULL, 'foo', b'1')

INSERT INTO `subscription` (`userId`, `serviceName`, `imsi`,
`mobilePushId`) VALUES ('<new_userId>', 'fiware', '<new_IMSI>', 'alice')
```

If the new user is successfully published to the Positioning Enabler make sure the Android client is able to subscribe for the new user (see "Installation" of Android client for more details).

### 11.2.2 Databases

To see if the MySQL database is setup correctly query it with "SHOW TABLES;". The following tables have to exist (there will be other tables as well but they are not in the scope of this document):

- fiwareNotification
- service
- subscription

- userLocation
- userProfile

### 11.2.3 List of Running Processes

A list of running processes related to the PE environment (users can vary by installation procedure):

postgres	/usr/lib/postgresql/9.1/bin/postgres -D /var/lib/postgresql/9.1/main -c config_file=/etc/postgresql/9.1/main/postgresql.conf
postgres	postgres: writer process
postgres	postgres: wal writer process
postgres	postgres: autovacuum launcher process
postgres	postgres: stats collector process
mysql	/opt/mysql/server-5.6/bin/mysqld
root	/bin/sh /opt/mysql/server-5.6/bin/mysqld_safe
tomcat	/usr/lib/jvm/default-java/bin/java
www-data	/usr/sbin/apache2

### 11.2.4 Network interfaces Up & Open

Open ports (all TCP):

- 8080
- 5228
- 5229
- 5230

## 11.3 Diagnosis Procedures

For Diagnosis of the Positioning Enabler module you can mainly use the catalina.out file under the following path: “<tomcat install directory>/logs/”

### 11.3.1 Resource availability

For the specific components, it is recommended to have at least 1GB RAM allocated and around 30MB hard-disk space for the installation. Additional resources are required for the Tomcat server summing up to 1.5GB RAM and 1GB hard-disk space for the installation.

### 11.3.2 Remote Service Access

The components are accessible on port 8080. Additionally, for a correct functioning PE requires connectivity to an EPC operator core network connected to at least one access network as described in the configuration part of this document.

### 11.3.3 Resource consumption

Memory consumption is high within the environment of the PE. This is due to a running MySQL server as well as Tomcat and Apache and, of course, the PE itself. CPU and Disk space usage are low.

### 11.3.4 I/O flows

For the communication between applications and the Positioning Enabler and Android app http data traffic is expected on port 8080 as well as 5228, 5229, and 5230 (for Google Cloud Messaging service). For the communication with the network, database connections will be established with the ANDSF.



## 12.3 SCC (Seamless Connectivity Client) installation and configuration

### 12.3.1 Installation

1. download the SeamlessNetworkConnectivity-r2.3.zip file from [FI-WARE Forge site](#)
2. choose a directory you wish to install the SCC
3. untar the file in this directory
4. open the variables.sh file with an editor (please edit the following variables)
  1. REMOTE\_IP\_SCS (this is the physical IP address of the SCS)
  2. LOCAL\_IP\_SCC\_1 (this is the physical IP address of the first access network)
  3. LOCAL\_IP\_SCC\_2 (this is the physical IP address of the second access network)
  4. UPPER\_TUNNEL (name of the tunnel interface, first tunnel)
  5. LOWER\_TUNNEL (name of the tunnel interface, second tunnel)
  6. SCC\_DEFAULT\_ROUTE\_ADDRESS\_1 (remote address of access network 1)
  7. SCC\_DEFAULT\_ROUTE\_ADDRESS\_2 (remote address of access network 2)
  8. BRIDGE\_INTERFACE (name of the interface showing up in the interfaces dialog, you don't have to change this)
5. Before you start the installation script, please make sure that you configured the same IP addresses as configured in the variable.sh script
6. start the install.sh script and follow the instructions (during the set up the script will ask if it will be installed on the server or client. For the SCC use 'c' for client)
7. configure the <Bridge Interface> IP address persistently

The script will install some Debian packages, also provided in this downloaded file. After that it generates a script (/etc/rc.local.snc) which contains all necessary setup which has to be done at boot time e.g. setting IP addresses, adding additional routes and adding flows. This script will be triggered by the /etc/init.d/rc.local boot script.

## 12.4 SCS (Seamless Connectivity Server) installation and configuration

### 12.4.1 Installation

1. download the SeamlessNetworkConnectivity-r2.3.zip file from [FI-WARE Forge site](#)
2. choose a directory you wish to install the SCS

3. untar the file in this directory
4. open the variables.sh file with an editor (please edit the following variables)
  1. REMOTE\_IP\_SCC\_1 (this is the physical IP address of the first access network of the SCC)
  2. REMOTE\_IP\_SCC\_2 (this is the physical IP address of the second access network of the SCC)
  3. LOCAL\_IP\_SCS (this is the physical IP address of the SCS)
  4. UPPER\_TUNNEL (name of the tunnel interface, first tunnel)
  5. LOWER\_TUNNEL (name of the tunnel interface, second tunnel)
  6. BRIDGE\_INTERFACE (name of the interface showing up in the interfaces dialog, you don't have to change this)
5. Before you start the installation script, please make sure that you configured the same IP addresses as configured in the variable.sh script
6. start the install.sh script and follow the instructions (during the set up the script will ask if it will be installed on the server or client. For the SCS use 's' for server)
7. configure the <Bridge Interface> IP address persistently

The script will install some Debian packages, also provided in this downloaded file. After that it generates a script (/etc/rc.local.snc) which contains all necessary setup which has to be done at boot time e.g. setting IP addresses, adding additional routes and adding flows. This script will be triggered by the /etc/init.d/rc.local boot script.

## 12.5 Sanity check procedures

1. Check IP addresses of the interfaces with ip address list they should comply with the addresses configured in the variables.sh file.
2. Check the routes on the machines
  1. ip route list (there should be a default route via the bridge interface (SCC))
  2. **SCC only:** ip route list table <Lower Tunnel> and ip route list table <Upper Tunnel> (there should be an entry for the ip address of the SCS)
3. **SCC only:** Check ip rule list (here should be two entries, one pointing to table <Lower Tunnel> and one pointing to table <Upper Tunnel> with the matching local addresses)
4. Check if the bridge interface contains the rules specified in the script.

### 12.5.1 End to End testing

This test will show if the network settings are made properly, which is necessary to have a working enabler.

The seamless network connectivity sub enabler is an enabler which offers enhanced network connectivity, it is therefore an OSI layer three "application". That means that the connection can be checked by a simple ping packet.

For the ping use the IP address configured on the <Bridge Interface> at the SCC and SCS.

The ping packet will be transmitted over the tunnel interface. The based on the automatically installed example rule set the packet will use port 1 (the UPPER\_TUNNEL) of the bridge interface of the SCC. The response will be treated similarly at the SCS bridge interface.

The successful transmitted and received ping indicates that the network settings were correctly made and additional tests can be performed.

For further tests see the [Unit Testing Plan](#) page.

## 12.5.2 List of Running Processes

Normally there are running the following processes on both machines:

- ovs-controller
- ovssdb-server
- ovs-vswitchd

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	2315	0.0	0.2	22288	964	?	Ss	14:01	0:00	/usr/bin/ovs-controller --detach --pidfile=/var/run/openvswitch/ovs-controller.pid
										psstl: --private-key=/etc/openvswitch-controller/privkey.pem
										-- certificate=/etc/openvswitch-controller/cert.pem
										--ca- cert=/etc/openvswitch-controller/cacert.pem
root	2752	0.0	0.0	0	0	?	S	14:01	0:00	[ovs_workq]
root	2776	0.2	0.1	22236	576	?	S<s	14:01	0:01	ovssdb-server: monitoring pid 2777 (healthy)
root	2777	0.0	0.5	22376	2104	?	S<	14:01	0:00	ovssdb-server /etc/openvswitch/conf.db -vconsole:emer -vsyslog:err -vfile:info
										-- remote=punix:/var/run/openvswitch/db.sock --private-key=db:Open_vSwitch,SSL,private_key

```

--
certificate=db:Open_vSwitch,SSL,certificate
--
bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert --no-chdir
--log-
file=/var/log/openvswitch/ovsdb-server.log
--
pidfile=/var/run/openvswitch/ovsdb-server.pid --detach --monitor
root      2787  0.2  0.1  22664   592 ?          S<s  14:01   0:01 ovs-
vswitchd: monitoring pid 2788 (healthy)
root      2788  0.0  1.6  23184  6304 ?          S<L  14:01   0:00 ovs-
vswitchd unix:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -
vfile:info
--
mlockall --no-chdir --log-file=/var/log/openvswitch/ovs-vswitchd.log
--
pidfile=/var/run/openvswitch/ovs-vswitchd.pid --detach --monitor
root      2789  0.0  0.3  22668  1176 ?          S<   14:01   0:00 ovs-
vswitchd: worker process for pid 2788

```

*This list is reformatted.*

### 12.5.3 Network interfaces up & open

The configured bridge interface (Standard name: snc0) should be up. With the command `ovs-vsctl show` you can check if the switch is configured correctly and contains both tunnels. `ovs-ofctl show <Bridge Interface>` shows the assigned ports for the Tunnels. Port 1 should be the upper tunnel and port 2 the lower tunnel. If it is not the case follow the following guideline:

- `ovs-vsctl del-port <Bridge Interface> <Upper Tunnel>`
- `ovs-vsctl del-port <Bridge Interface> <Lower Tunnel>`
- `/etc/init.d/ovs-vswitchd stop`
- `/etc/init.d/ovs-vswitchd start`
- For SCC: `ovs-vsctl add-port <Bridge Interface> <Upper Tunnel> -- set interface <Upper Tunnel> type=gre options:remote_ip=<REMOTE_IP_SCS> options:local_ip=<LOCAL_IP_SCC_1> options:key=<TUNNEL_KEY_1>`
- For SCC: do the same with the other tunnel, of course with different ip addresses (look at the `create_interfaces_client.sh` script)



- For SCS: look at the create\_interfaces\_server.sh script and do the same thing like above.
- ovs-ofctl show <Bridge Interface> should now show the right ports

#### 12.5.4 Databases

N/A

## 12.6 Diagnostic Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

#### 12.6.1 Resource availability

- at least 1GB of RAM
- 8 GB hard disk for the operating system, the SNC GE doesn't consume much storage

#### 12.6.2 Remote Service Access

N/A

#### 12.6.3 Resource consumption

In high load situation there is about 30 – 40% CPU load caused by the ovs-vswitchd process. But even if it is a higher load that doesn't mean that it could be a fault.

#### 12.6.4 I/O flows

At the external interfaces should be GRE encapsulated packets during traffic transmission.

## 13 S3C GE - API Mediation - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 13.1 API Mediation installation

#### 13.1.1 List of components

- OSA:
  - Apache module: additional apache module that manages the mediation controls for OSA. This module protects, with some other standard modules, the access to the back end APIs.
  - appliance add-ons: set of web api transformers managing more mediations for some APIs (conditional routing, or adding SOAP headers to propagate identity...)
  - Web application that allow to subscribe to APIs in a marketplace

#### 13.1.2 Prerequisites

##### 13.1.2.1 *Installation of ubuntu 12.04*

We suppose that we have a x86 PC of a WM with an installed and running ubuntu LTS 12.04 version. For installing ubuntu 12.04, please refer to <http://www.ubuntu.com>. Choose any kind of machine name and install at least one user account. We call the PC “PCtest” and the user account “user” in the command lines we give in the next chapter.

##### 13.1.2.2 *General*

**This prerequisite is only useful is you want to install the components with a mysql database located in another server.**

In this case, and only in this case, during the install/config, the *root* mysql user must be able from the remote machine:

- create/remove users
- create/remove databases
- create/remove tables
- administrate grants

For this, the simplest way is to connect mysql root locally on the DB server and to enter:

```
GRANT all on *.* to 'root'@'%' identified by 'password' WITH GRANT OPTION;
```

After the installation, for safety you can run:

```
DELETE from mysql.user WHERE user='root' AND host='%';  
flush privileges;
```

### 13.1.2.3 **OSA**

Apache module: apache2

```
apt-get install apache2
```

Appliance Manager: apache2 php5 php5-mysql php5-curl openssl curl zip mysql-server

```
apt-get install apache2 php5 php5-mysql php5-curl curl zip mysql-server
```

### 13.1.2.4 **Appliance add-ons**

OSA apache2 php5 php5-curl

```
apt-get install apache2 php5 php5-curl
```

### 13.1.2.5 **Selfcare**

OSA tomcat6 curl zip mysql-server

```
apt-get install tomcat6 curl zip mysql-server
```

### 13.1.2.6 **All**

```
apt-get install mysql-server tomcat6 apache2 php5 php5-mysql php5-curl  
openssl curl zip
```

## 13.1.3 API mediation and admin GUI installation

### 13.1.3.1 **Open Services Access**

Download the file libapache2-mod-osa\_1.0.0.1271-nursery1\_i386.deb from the [API Mediation package](#), and execute it on your linux environment (debian 12.04 32 bits).

### 13.1.3.2 **Appliance Manager (admin GUI)**

Download the file ApplianceManager\_1.0.1409.tgz from the [API Mediation package](#) in a temp folder, and uncompress it:

```
tar xvf ApplianceManager_1.0.1409.tgz
```

Copy ApplianceManager.php and RunTimeApplianceManager to your install dir (called further \$INSTALL\_DIR)

- Note: Application is pre-configured to be installed in /usr/local/nursery/ApplianceManager

run install.sh with \$INSTALL\_DIR as argument

```
./install.sh /usr/local/nursery/ApplianceManager
```

Go to \$INSTALL\_DIR/RunTimeAppliance/shell Edit envvars.sh file and set configuration variables according to your system. By default just the following are required:

- APACHE\_ADMIN\_MAIL: system administrator mail (for information, no email send)
- ROOT\_MYSQL\_PW: password for mysql "root" user
- APPLIANCE\_MYSQL\_PW: wished password for mysql user created to allow application to connect mysql on application database
- APPLIANCE\_ADMIN\_PW: "admin" application user wished password
- USE\_HTTP: publish (or not) service with HTTP
- USE\_HTTPS: publish (or not) service with HTTPS

and (end of file)

- INSTALL\_DIR: application location
- LOG\_DIR: application logs location

run ./configure-osa.sh

```
./configure-osa.sh
```

If, at the end of execution the message "OSA Configuration done, exiting..." appears, OSA is correctly installed!

## 13.2 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 13.2.1 End to End testing

After install, connect <http://localhost:82/ApplianceManager> or <https://ServerName:6443/ApplianceManagerAdmin> with admin as user and \$APPLIANCE\_ADMIN\_PW to manage services publishing (port used can be changed via envvars.sh file) If HTTPS publication is activated services are available at:

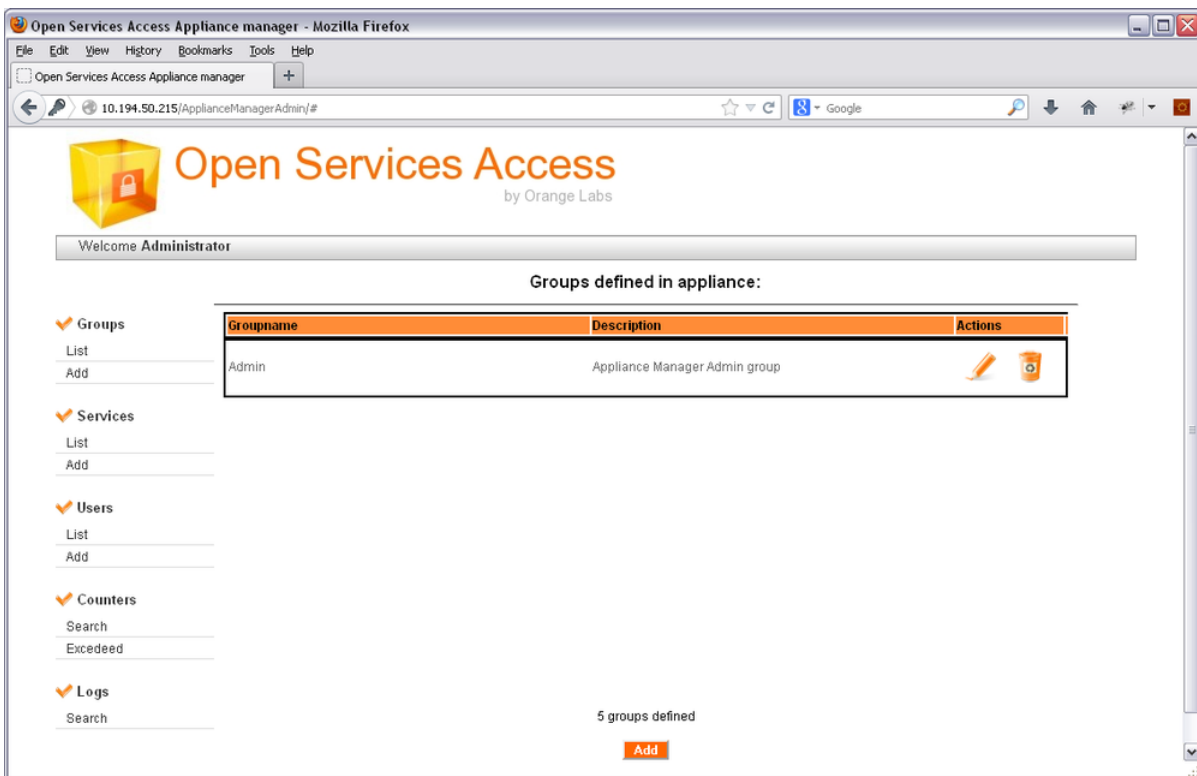
<https://ServerName:8443/your-service> (port used can be changed via envvars.sh file)

If HTTP publication is activated services are available at:

<http://ServerName:81/your-service> (port used can be changed via envvars.sh file)

- NOTE: If you re run configure-osa.sh all the configuration is re-build, including drop and re-create of database if already existing. To preserve existing DB ad -keep-db parameter to configure-osa.sh

You should see such a screen:



### 13.2.2 List of Running Processes

/usr/sbin/apache2 -k start

/usr/sbin/mysqld

### 13.2.3 Network interfaces Up & Open

Several interfaces have to be open and up:

https 6443 for administration. This value is configured in envvar.sh (HTTPS\_ADMIN\_VHOST\_PORT)

http port 81 and https port 8443 are used by default for the mediation. These values are configured in envvar.sh (values HTTP\_VHOST\_PORT HTTPS\_VHOST\_PORT)

http port 82 is used locally for administration. This value is configured in `envar.sh` (`PRIVATE_VHOST_PORT`).

mysql port 3306.

### 13.2.4 Databases

Mysql database with schema **appliance** has to be up. By default, a user `appliance` is created, and a password has to be entered in the `envvars.sh` file (`APPLIANCE_MYSQL_PW`). To test it enter the command:

```
mysql -uappliance -p
```

with the password configured. Then in the mysql command:

```
use appliance
select count(*) from users
```

A single user `Admin` should appear in the table.

## 13.3 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

### 13.3.1 Resource availability

The API mediation is set to work on a 1GB RAM VM, mono processor, and a 8GB hard drive.

### 13.3.2 Remote Service Access

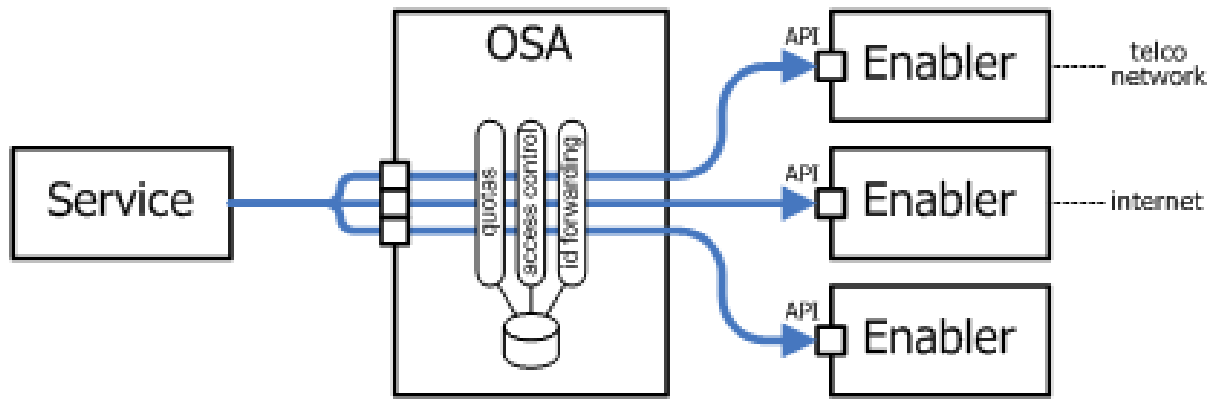
N/A

### 13.3.3 Resource consumption

Resources needed for the complete installation of the enabler is about 50MB on the hard drive. Free memory of 1GB is needed to ensure a high rate response of the mediation.

### 13.3.4 I/O flows

As stated in the picture below, the API mediation platform is used like a proxy. On the left side, the consumer of the services, on the right side, the producer. The consumers have to use the `http` and `https` ports configured during the installation. The producer may be very heterogeneous, and the flows depend on the configuration of the service.



## 14 S3C GE - Telecom AS - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

### 14.1 Telecom AS Installation

Telecom AS rely on a Mobicent SIP platform that should be installed.

#### 14.1.1 Step 1: Installation of ubuntu 12.04

We suppose that we have a x86 PC of a WM with an installed and running ubuntu LTS 12.04 version. For installing ubuntu 12.04, please refer to <http://www.ubuntu.com>. Choose any kind of machine name and install at least one user account. We call the PC "PCtest" and the user account "user" in the command lines we give in the next chapter.

#### 14.1.2 Step 2: Installation of mobicents

The Mobicents Communication Platform is a server that allows to create, deploy and manage services and applications integrating voice, video and data across a range of IP and legacy communications networks. Follow the steps described here: [JBoss/Mobicents installation](#)

Download the JSR 289 sip-servlets-ized version of Jboss AS 5.0.1.GA [nightly snapshot](#) in a temp directory. Unzip it in a jboss directory (for example `/opt/servers/mss-jboss-5.0.1.GA`).

Set the JBOSS\_HOME env variable to your freshly unzipped root directory of jboss such as  
`JBOSS_HOME=/opt/servers/mss-jboss-5.0.1.GA`

The best place to set this environment variable is to put the command in the `.bashrc` file, if JBoss is started in command line mode.

#### 14.1.3 Step 3: Get and uncompress the OneAPIVoiceCallControlEnabler

First operation: get the file `oneapi-VoiceCallControl.tar` from the [Telecom AS package](#). Uncompress it in the `=/opt/servers/mss-jboss-5.0.1.GA/server/default/deploy`

```
tar xvf oneapi-VoiceCallControl.tar -C /opt/servers/mss-jboss-5.0.1.GA/server/default/
```

(As the deploy directory is already in the archive, it should not be included un the path of the command).

#### 14.1.4 Step 4 : Configure the files

**Important remark:** The Voice Call Control API provided is based on an Orange service call IKEA. Ensure that this service is enabled and accessible. For this reason, the machine should access the internet.

Configure the url into properties file deliver as sample. This url is the endpoint of the Orange IKEA service.



### 14.1.5 Step 5: Launch JBoss

From the bin directory (*/opt/servers/mss-jboss-5.0.1.GA/bin*), launch JBoss with *./run.sh* command.

## 14.2 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 14.2.1 End to End testing

We propose a simple test to conclude the installation process. You have to use a REST Console plugin in Google Chrome or REST Client in Firefox to test the REST functionalities. To do this, simply enter the following parameters:

```
POST http://ServerName/vccV1/thirdpartycall/callSessions HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json

{"callSessionInformation": {
  "participant": [
    { "participantAddress": "tel:+33ABPQMCDU",
      "participantName": "Mister Bean"},
    { "participantAddress": "tel:+33ABPQMCDU",
      "participantName": "John Smith"
    }
  ]
}}
```

The server should answer:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: tp://ServerName/vccV1/thirdpartycall/callSessions/cs001
Content-Length: nnnn
```

Date: Mon, 27 Jun 2013 18:15:21 GMT

```
{ "callSessionInformation": {
  "participant": [
    { "participantAddress": "tel:+33ABPQMCDU",
      "participantName": "Mister Bean",
      "participantStatus": "CallParticipantConnected",
      "resourceURL":
"http://ServerName/vccV1/1/thirdpartycall/callSessions/cs001/participants
/pt001",
      "startTime": "2010-06-28T17:50:51"
    },
    { "participantAddress": "tel:+33ABPQMCDU",
      "participantName": "John Smith",
      "participantStatus": "CallParticipantInitial",
      "resourceURL":
"http://ServerName/vccV1/1/thirdpartycall/callSessions/cs001/participants
/pt002"
    }
  ],
  "resourceURL":
"http://ServerName/vccV1/1/thirdpartycall/callSessions/cs001",
  "terminated": "false"
}}
```

**Important remark:** Only pre declared tel number can be used for test (declare tel number to Orange before test). Request has to be done to the contacts mentioned in the [FI-WARE S3C Catalogue page](#).

#### 14.2.2 List of Running Processes

The only process that have to be checked is the JBoss process

```
/opt/servers/mss-jboss-5.0.1.GA/bin/run.sh
```

### 14.2.3 Network interfaces Up & Open

The JBoss ports 80 for http and 443 for https have to be opened and in use. Also, 5080 port on UDP should be opened for SIP traffic.

### 14.2.4 Databases

N/A - No database is used in this implementation of the enabler.

## 14.3 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE.

Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

### 14.3.1 Resource availability

The enabler is set to work on a 1GB RAM VM, mono processor, and a 8GB hard drive.

### 14.3.2 Remote Service Access

The enabler is delivered with a connector top the IKEA Orange platform. If you intend to use the Voice Call Control enabler, you have to check the URL that is configured in the configuration file. Use the url, and proceed to use it in a web browser or with a wget command line. If it works you should be prompted to a user/password.

If these tests are not successful, then you should check your network settings, or availability of services.

### 14.3.3 Resource consumption

Resource needed for the complete installation of the enabler is about 50Mo on the hard drive. Free memory of 500Mo is needed.

### 14.3.4 I/O flows

The Voice Call Control API is implemented in the form of a generic enabler (V.C.C.E) interacting with other enablers via a SOAP interface. The V.C.C.E is managed by a service logic via REST API specified by GSMA.

