SEVENTH FRAMEWORK PROGRAMME

Specific Targeted Research Project

Call Identifier:	FP7–ICT–2011–7
Project Number:	287305
Project Acronym:	OpenIoT
Project Title:	Open source blueprint for large scale self-organizing cloud environments for IoT applications

D6.2.1 Report on Users, Contributors and Developers Support a

Document Id:	OpenIoT-D621-131223-Draft
File Name:	OpenIoT-D621-131223-Draft.pdf
Document reference:	Deliverable 6.2.1
Version:	Draft
Editor(s):	Arkady Zaslavsky
Organisation:	CSIRO
Date:	2013 / 12 / 23
Document type:	Deliverable
Security:	PU (Public)

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Arkady Zaslavsky	CSIRO	2013/11/04	ToC & contributors
V02	Arkady Zaslavsky	CSIRO	2013/11/12	ToC & contributors updated
V03	Nikos Kefalakis	AIT	2013/11/27	New ToC proposal
V04	Johan E. Bengtsson	AL	2013/11/29	Some contents updated
V05	Arkady. Zaslavsky	CSIRO	2013/12/03	ToC and content updated
V06	Nikos Kefalakis	AIT	2013/12/09	Added Sections 3, 4, 6 and definitions of roles, users, contributors in section 2
V07	Reinhard Herzog	IOSB	2013/12/10	Added IOSB related use case
V08	Arkady Zaslavsky	CSIRO	2013/12/10	Edited/added content to various sections and finished the document
	Johan E. Bengtsson	AL	2013/12/16	Contributed to Chapter 5 (added 5.3). Many small suggested improvements.
V09	Panos Dimitropoulos	SENSAP	2013/12/14	Technical Review
V10	Kresimir Pripuzic	UNIZG-FER	2013/12/22	Quality Review
V11	Arkady Zaslavsky	CSIRO	2013/12/22	Final editing
V12	Martin Serrano	DERI	2013/12/22	Circulated for Approval
V13	Martin Serrano	DERI	2013/12/23	Approved
Draft	Martin Serrano	DERI	2013/12/23	EC Submitted

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ACRONYMS

Term	Meaning
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
AAL	Ambient Assisted Living
ARM	Architecture Reference Model
BPM	Business Process Language
BPMN	Business Process Modelling Notation
BPWME	Business Process Workflow Management Editor
CoAP	Constrained Application Protocol
CPI	CSIRO Plant Industry
CRUD	CReate, Update, Delete
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DoW	Description-of-Work
DSO	Decision Support Ontology
EPC	Electronic Product Code
EPC-ALE	Electronic Product Code Application Level Events
EPC-IS	Electronic Product Code Information Sharing
ERP	Enterprise Resource Planning
GPL	General Public Licence
GPS	Global Positioning System
GSN	Global Sensor Networks
GTIN	Global Trade Item Number
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSF	Java Server Faces
ICO	Internet-Connected Objects
ICT	Information and Communication Technologies
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IERC	Research Cluster for the Internet of Things
IoT	Internet of Things
LGPL	Lesser General Public License

MRP	Manufacturing Resource Planning
OGC	Open Geospatial Consortium
OMG	Object Management Group
ONS	Object Naming Service
OSS	Open Source Software
PDA	Personal Digital Assistant
PET	Privacy Enhancing Technologies
QoS	Quality of Service
QR-Code	Quick Response Code
RDF	Resource Description Format
REST	Representational State Transfer
RFID	Radio Frequency Identification
SGTIN	Serialized Global Identification Number
SLA	Service Level Agreement
SME	Small Medium Enterprise
SOA	Service Oriented Architecture
SOS	Sensor Observation Service
SPS	Sensor Planning Service
SSN	Semantic Sensor Networks
UML	Unified Modelling Language
WSN	Wireless Sensor Networks
XML	eXtensible Markup Language

GLOSSARY AND TERMINOLOGY

Term	Meaning
(OpenIoT) Architecture	The set of software and middleware components of the OpenIoT platform, along with the main structuring principles and inter-relationships driving their integration in an IoT/cloud platform.
(OpenIoT) Middleware	System level software (compliant to the OpenIoT architecture), which facilitates the integration of on-demand cloud-based IoT services.
(OpenIoT) Platform	A set of middleware libraries and tools, which enable the development and deployment of (OpenIoT compliant) cloud-based IoT services
(OpenIoT) Use Case	A domain-specific application serving needs of end users, which is implemented based on the OpenIoT platform.
(OpenIoT) Service	An IoT service deployed over the OpenIoT platform.
(OpenIoT) Scenario	A specific set of interactions between OpenIoT components serving the needs of an application.
(OpenIoT) Cloud	A set of computing resources enabling the delivery of IoT services over the network and based on the use of OpenIoT platform.
Global Scheduler	A software component that regulates how IoT services access the different resources managed by the OpenIoT platform.
Local Scheduler	A software component that regulates how IoT services access the local resources managed by an instance of the sensor middleware (and more specifically the GSN middleware).
Utility Metrics	A set of quantities that are used for the metering of IoT services.
(OpenIoT) Service Delivery	The process of deploying and offering an OpenIoT service, after selecting the resources that are involved in the service.
(OpenIoT) Request Presentation	The software components that visualize the outcomes of an OpenIoT service based on the use of appropriate mashups and mashup libraries
Sensor Selection	The process of selecting sensors that can contribute information to a particular service.

Virtual Sensor	All the physical or virtual items (i.e. services, persons, sensors, GSN nodes) which provide their information through a GSN endpoint.
Sensor Discovery	The process of detecting physical and virtual sensors, as well as of the services offered by them.
Resource Discovery	The process of detecting an IoT resource (such as a sensor, a service or a database).
Utility Manager	A software component (part of the OpenIoT platform), which performs metering based on the tracking and combination of utility metrics.
Sensor Directory	A software service that stores organizes and provides access to information about (physical and virtual) sensors.
(OpenIoT) Sensor Middleware	The part of the OpenIoT middleware platforms that facilitate access to, collection and filtering of OpenIoT data streams.
Global Sensor Networks (GSN)	An open source sensor middleware platform enabling the development and deployment of sensor services with almost zero-programming effort.
Data Streams	A stream of digital information stemming from a physical or virtual sensor.
Data Stream Engine	A software component enabling the processing of data streams, as well as the management of the process of publishing and subscribing to data stream.
Linked Sensor Data	A set of (Semantic Web) technologies for exposing, sharing, and connecting sensor data, information, and knowledge.

1 INTRODUCTION

1.1 Scope

This deliverable is a living document describing Report on Users, Contributors and Developers Support (a), in-line with the intentions and exploitation modalities listed in the description of work. It includes information on who is using and/or contributing to OpenIoT and for which purpose. It is the first of two releases planned for November, 2013 and the second release is planned for November 2014.

1.2 Audience

The audience of this document is primarily the European Commission, the OpenIoT consortium, user and developer communities which are currently being established and planned for growth. This document also acts as a reference for OpenIoT end-users, software developers and anyone interested in the software developments of the OpenIoT project for development of applications and services based on OpenIoT open source software platform as well as for dissemination purposes.

It is important to accurately define roles of people and entities that will be using OpenIoT, contributing to its development and taking it further to SMEs, industry, business with the purpose of developing novel applications, services and IoT-based systems. The following definitions are intended to define roles:

- [1]A **user** is an agent, either a human agent (**end-user**) or software agent, who uses a computer or network service. A user often has a **user account** and is identified by a **username** (also **user name**). Users are also widely characterized as the class of people that use a system without complete technical expertise required to understand the system fully. In the context of the OpenIoT platform, a user is the entity (natural person, legal entity, or software service), which uses IoT services deployed based on the OpenIoT platform and within the OpenIoT cloud environment.
- [2]The end user is the individual who uses the product after it has been fully developed and marketed. The term is useful because it distinguishes two classes of users, users who require a bug -free and finished product (end users), and users who may use the same product for development purposes. The term end user usually implies an individual with a relatively low level of computer expertise. Unless a person is a programmer or engineer, s/he is almost certainly an end user. End-user in the OpenIoT context is the human user who takes advantage of the IoT services that are deployed in the OpenIoT cloud environment.
- [3]A (**software) developer** is a person concerned with facets of the software development process. Their work includes researching, designing,

[1] http://en.wikipedia.org/wiki/User_%28computing%29

[2] http://www.webopedia.com/TERM/E/end_user.html

[3] http://en.wikipedia.org/wiki/Software_developer

implementing, and testing software. A software developer may take part in design, computer programming, or software project management. They may contribute to the overview of the project on the application level rather than component-level or individual programming tasks. In OpenIoT context, software developers are the entities in charge of developing the software that comprises the OSS platform of the project.

- Contributors are approved guardians of OSS developed on the basis of OpenIoT platform, components and services and who maintain OpenIoT compatible-licenced software in OSS repositories, eg, github. In the scope of OpenIoT, contributors are people and organizations that contribute (usually software) to the open source platform of the project towards improving and enhancing its functionality.

The communities of users, developers and contributors are intersecting, interacting and cooperating for the benefit of OpenIoT OSS advancement.

Definitions of OSS developers, users and contributors have not been standardised and are open to various project- or initiative-related nuances and interpretations. Further discussion of various roles is in section 2.2 below.

1.3 Summary

This is the second deliverable of WP6 following a deliverable D6.1 which reported on the establishment of the OpenIoT open source software portal and wiki. The establishment of the open source software portal and wiki raised many issues, such as the selection of an open source licensing scheme, the selection of collaboration tools, as well as the selection of appropriate governance and collaboration procedures. The decisions taken by the OpenIoT consortium were described in deliverable D6.1. It is also worth mentioning that the OpenIoT open source software portal and wiki are evolving in the course of the project. This deliverable D6.2.1 addresses the issues of building developer and user communities, describing the OpenIoT experience in the process up until now, describing practical roles, issues of open source governance and infrastructure, providing statistics on use of OpenIoT repositories, tools and components. It also describes the current set of users and contributors to the OpenIoT open source project. The second report on user, developer communities and deployment of OpenIoT software will be addressed in the deliverable D6.2.2.

1.4 Structure

The deliverable is structured as follows: section 2 addresses the issues of OpenIoT open source software governance, including roles and partner responsibilities; section 3 describes the OpenIoT open source infrastructure; section 4 describes how the developers’ community is being built and what are the issues with supporting OpenIoT software developers; section 5 describes how the OpenIoT consortium intends to build and support user community; section 6 provides some statistics on OpenIoT software usage; section 7 concludes the document and describes some lessons learnt.

2 OPENIOT OPEN SOURCE GOVERNANCE

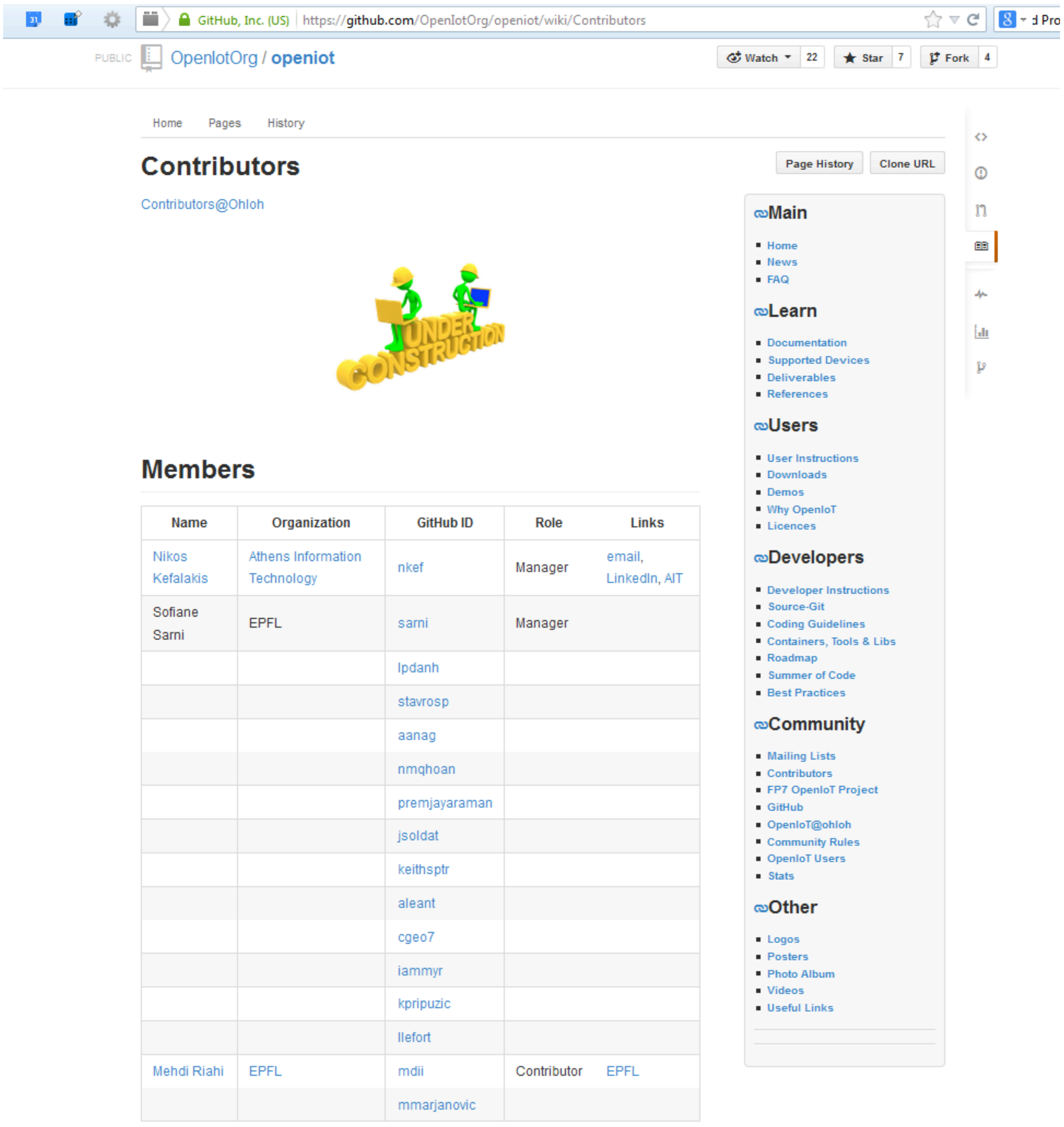
2.1 Structure

OpenIoT open source project has selected a proper governance scheme, which regulates the interactions between the members of the open source community, including key roles and responsibilities for the development and expansion of the project's software code. OpenIoT adopts an incremental, iterative and evolutionary software development process, notably based on agile development techniques. Such a development process alleviates the limitations of conventional waterfall software engineering processes, while at the same time enables the continued development and evolution of the open source software beyond the end of the FP7 OpenIoT project (i.e. as part of the sustainability phase of the project). Agile methodologies go beyond central planning, which facilitate the expansion of the project towards additional innovative directions. Such directions include the development of components, which were not foreseen at the first place in the scope of the OpenIoT architecture. Furthermore, agile methodologies are in generally in-line with the iterative nature of the OpenIoT work plan, which foresees iterative incremental delivery of key deliverables in WP4, WP5 and WP6. Last but not least, agile methodologies allow OpenIoT contributors (both teams and/or individual contributors) to incrementally plan, design and deliver open source software.

The OpenIoT consortium made the following decisions:

- A master-governed approach is the starting scheme associated with the establishment, governance and initial evolution of the OpenIoT open source project. The goal of this decision is to ensure proper integration of the various parts of the project, at least in the initial phase of the project where some critical mass has to be developed. It is the phase where the project will be looking for good reputation among the IoT open source communities.
- EPFL members act as master(s) for the part of the project that concerns the lower-level sensor/ICO information acquisition and filtering, notably on the basis of the enhancements to be realized on top of the Global Sensor Networks (GSN) middleware.
- DERI members act as master(s) for the part of the project dealing with semantic annotation of sensors/ICO and related discovery and filtering processes.
- AIT members will act as master(s) for the part of the project dealing with service requests formulation and delivery.
- Following the first release of the OpenIoT integrated software platform the consortium doesn't see the need for switching to a collaborative planning approach (for one or more of the above (sub) projects).

The example of contributions to Github repository is below



Contributors

Contributors@Ohloh

Members

Name	Organization	GitHub ID	Role	Links
Nikos Kefalakis	Athens Information Technology	nkef	Manager	email, LinkedIn, AIT
Sofiane Sarni	EPFL	sarni	Manager	
		lpdanh		
		stavrosp		
		aanag		
		nmqhoan		
		premjayaraman		
		jsoldat		
		keithsptr		
		aleant		
		cgeo7		
		iammyr		
		kpripuzic		
		llefort		
Mehdi Riahi	EPFL	mdii	Contributor	EPFL
		mmarjanovic		

Main
- Home
- News
- FAQ

Learn
- Documentation
- Supported Devices
- Deliverables
- References

Users
- User Instructions
- Downloads
- Demos
- Why OpenIoT
- Licences

Developers
- Developer Instructions
- Source-Git
- Coding Guidelines
- Containers, Tools & Libs
- Roadmap
- Summer of Code
- Best Practices

Community
- Mailing Lists
- Contributors
- FP7 OpenIoT Project
- GitHub
- OpenIoT@ohloh
- Community Rules
- OpenIoT Users
- Stats

Other
- Logos
- Posters
- Photo Album
- Videos
- Useful Links

Figure 1. Screenshot of OpenIoT Github repository showing contributors

2.2 Defining developers, users and respective roles

Developer roles and specialisations are extensively discussed in [Aalto 2013]. The relevant extracts from this discussion are presented below.

The participants of an open-source community can be divided into three groups based on their level of contributions. A joiner is someone who has just recently joined the community and does not have access to the repository yet. When that person has made his first changes to the repository, he becomes a newcomer. A developer is a fully-fledged contributor that actively adds new code to the repository. [von Krogh 2003].

A developer often starts out by making bug fixes that are related to his work and interests. The bug fixes are not randomly scattered around the software but they tend to focus on the same modules. Gradually he gains acceptance and a higher status in the community through his bug fixes and participation in discussions and debates about new features. This process characterizes how a developer becomes an expert on some part of the architecture and is able to influence its development. [Ducheneaut, 2005].

Many software developers and users participate in OSS development and communities because they want to learn. The system architecture can be designed in a modularized way to create independent tasks with progressive difficulties so that newcomers can participate and move on gradually to take care of harder tasks. This approach can encourage more users to become developers. Developers at the centre of OSS communities should focus on developing the system as well as having enough attention to the creation and maintenance of a dynamic and self-reproducing OSS community. [Ye 2003].

A joining script refers to a barrier that a joiner has to overcome before he can understand the architecture and is accepted by other developers. It is a type of activity that he has to go through before he actually starts contributing code, and the failure to comply with the joining script can result in being denied to access to the developer community. Usually the joiner starts out by observing the technical discussion for several weeks or months without commenting anything himself. When the joiner starts participating in the discussion, it is most often about an existing, on-going issue. Joiners do not start out by making completely new technical suggestions. [von Krogh 2003].

The contributions of newcomers are not equally distributed across all modules. The newcomers tend to specialize on modules that have low contribution barriers. This contribution barrier depends on how easy the module is to modify and how easily it can be plugged into the overall architecture of the system. The pluggability of the architecture reduces the learning curve that is needed to make changes. Therefore it is important not to have intertwined modules if newcomers are expected to work on those modules. [Aberdour 2007]; [von Krogh 2003].

Only a few people fully understand the architecture of the OSS system [Von Krogh 2003]. Often developers contribute only to a small part of the software and they increasingly become knowledgeable about that part of the architecture, while their understanding of other parts of the architecture remains low. However, the contributions they make do not always exist in isolation from other artifacts and people. When adding a new piece of software, other people and artifacts might have to change to accommodate that addition. This in turn can create opposition in the interrelationships of people and artefacts that has to be managed before the addition is fully implemented. [Ducheneaut 2005]

According to Apache project terminology [Fielding, 1999], a *user* is someone that uses our software. Users contribute to the Apache projects by providing feedback to developers in the form of bug reports and feature suggestions. Users participate in the Apache community by helping other users on mailing lists and user support forums. A developer is a user who contributes to a project in the form of code or documentation. Users take extra steps to participate in a project, are active on the developer mailing list, participate in discussions, provide patches, documentation, suggestions, and criticism. *Developers* are also known as contributors. A *committer* is a developer that was given write access to the code repository and has a signed Contributor License Agreement (CLA) on file. Committers have an apache.org mail address. Not needing to depend on other people for the patches, they are actually making short-term decisions for the project. The project manager committee (PMC) can (even tacitly) agree and approve it into permanency, or they can reject it. Remember that the PMC makes the decisions, not the individual people.

In the Eclipse project [4] the terminology uses the following definitions. There are two different types of Project leadership at Eclipse: The Project Management Committee (PMC) and Project Leads. Both forms of leadership are required to:

- ensure that their Project is operating effectively by guiding the overall direction and by removing obstacles, solving problems, and resolving conflicts;
- operate using open source rules of engagement: meritocracy, transparency, and open participation; and
- ensure that the Project and its Sub-Projects (if any) conform to the Eclipse Foundation IP Policy and Procedures.

The leadership for a Project is composed of the Project's Project Lead(s), the leadership of the parent Project (if any) and the PMC Leads and PMC Members for the Top-Level Project. Eclipse Projects are managed by one or more Project Leads. Project Leads are responsible for ensuring that their Project's Committers are following the Eclipse Development Process, and that the project is engaging in the right sorts of activities to develop vibrant communities of users, adopters, and contributors. The initial project leadership is appointed and approved in the creation review. Subsequently, additional Project Leads must be elected by the project's Committers and approved by the Project's PMC and the EMO(ED).

In the unlikely event that a member of the Project leadership becomes disruptive to the process or ceases to contribute for an extended period, the member may be removed by the unanimous vote of the remaining Project Leads (if there are at least two other Project Leads), or unanimous vote of the Project's PMC.

In exceptional situations, such as projects with zero active committers or projects with disruptive Committers and no effective Project Leads, the Project Leadership Chain has the authority to make changes (add, remove) to the set of committers and/or Project Leads of that project. Each Project has a Development Team, led by the Project Leaders. The Development Team is composed of Committers and Contributors. Contributors are individuals who contribute code, fixes, tests, documentation, or other work that is part of the Project. Committers have write access to the Project's resources (source code repository, bug tracking system, website, build server, downloads, etc.) and are expected to influence the Project's development.

[4] http://www.eclipse.org/projects/dev_process/development_process.php#4_Structure_and_Organization

Contributors who have the trust of the Project's Committers can, through election, be promoted Committer for that Project. The breadth of a Committer's influence corresponds to the breadth of their contribution. A Development Team's Contributors and Committers may (and should) come from a diverse set of organizations. A Committer gains voting rights allowing them to affect the future of the Project. Becoming a Committer is a privilege that is earned by contributing and showing discipline and good judgment. It is not responsibility that should be neither given nor taken lightly, nor is it a right based on employment by an Eclipse Member company or any company employing existing committers.

At times, Committers may become inactive for a variety of reasons. The decision making process of the Project relies on active committers who respond to discussions and vote in a constructive and timely manner. The Project Leaders are responsible for ensuring the smooth operation of the Project. A Committer who is disruptive, does not participate actively, or has been inactive for an extended period may have his or her commit status revoked by the Project Leaders. (Unless otherwise specified, "an extended period" is defined as "no activity for more than six months".)

Active participation in the user forums/newsgroups and the appropriate developer mailing lists is a responsibility of all Committers, and is critical to the success of the Project. Committers are required to monitor and contribute to the user forums/newsgroups.

Committers are required to monitor the mailing lists associated with the Project. This is a condition of being granted commit rights to the Project. It is mandatory because committers must participate in votes (which in some cases require a certain minimum number of votes) and must respond to the mailing list in a timely fashion in order to facilitate the smooth operation of the Project. When a Committer is granted commit rights they will be added to the appropriate mailing lists. A Committer must not be unsubscribed from a developer mailing list unless their associated commit privileges are also revoked.

Committers are required to track, participate in, and vote on, relevant discussions in their associated Projects and components. There are three voting responses: +1 (yes), -1 (no, or veto), and 0 (abstain).

Committers are responsible for proactively reporting problems in the bug tracking system, and annotating problem reports with status information, explanations, clarifications, or requests for more information from the submitter. Committers are responsible for updating problem reports when they have done work related to the problem.

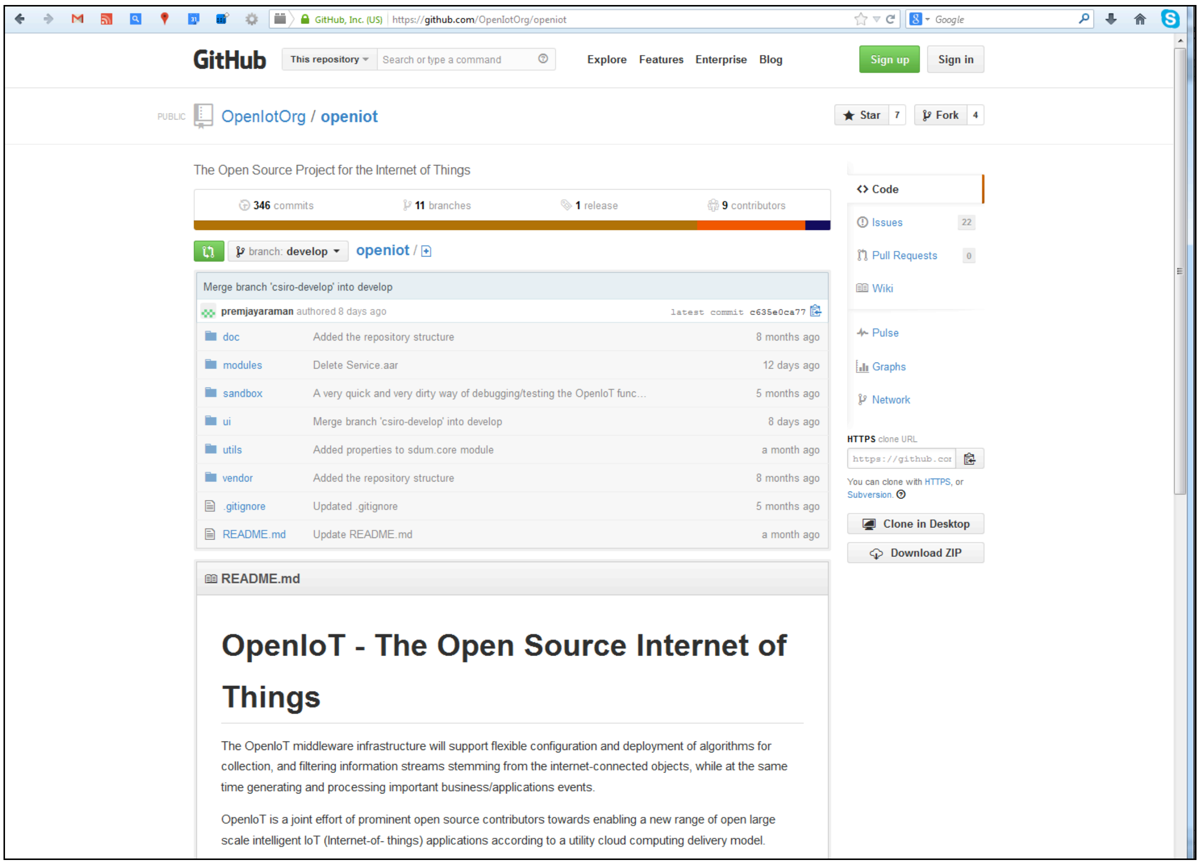
Committer, PMC Lead, Project Lead, and Council Representative(s) are roles; an individual may take on more than one of these roles simultaneously.

3 OPEN SOURCE INFRASTRUCTURE (AIT)

3.1 Source Code & Binaries

3.1.1 Source Code Availability

The OpenIoT repository (Figure 2) is hosted at the GitHub[5] which can be found at the following link: https://github.com/OpenIotOrg/openiot



OpenIotOrg / openiot

The Open Source Project for the Internet of Things

346 commits | 11 branches | 1 release | 9 contributors

branch: develop | openiot / +

Merge branch 'csiro-develop' into develop
premjayaraman authored 8 days ago | latest commit c635e0ca77

doc	Added the repository structure	8 months ago
modules	Delete Service.aar	12 days ago
sandbox	A very quick and very dirty way of debugging/testing the OpenIoT func…	5 months ago
ui	Merge branch 'csiro-develop' into develop	8 days ago
utils	Added properties to sdum.core module	a month ago
vendor	Added the repository structure	8 months ago
.gitignore	Updated .gitignore	5 months ago
README.md	Update README.md	a month ago

README.md

OpenIoT - The Open Source Internet of Things

The OpenIoT middleware infrastructure will support flexible configuration and deployment of algorithms for collection, and filtering information streams stemming from the internet-connected objects, while at the same time generating and processing important business/applications events.

OpenIoT is a joint effort of prominent open source contributors towards enabling a new range of open large scale intelligent IoT (Internet-of- things) applications according to a utility cloud computing delivery model.

Figure 2. Start screenshot of OpenIoT Github repository

The OpenIoT repository is divided in branches. The branches are divided in two thematic categories. One is the Documentation (i.e., site storage hosted at the "gh-pages" and Wiki). The other one is the Open IoT source code branch. The source code category is then divided in two branch types which are:

- Main branches with an infinite lifetime:
  - Master branch
  - Develop branch
- Supporting branches:

[5] http://github.com/

    - Feature branches
    - Release branches
    - Hotfix branches

3.1.2 Source code Structure

The OpenIoT source code is divided in functionality themes. For example all utilities are under the “utils” folder and all user interfaces under the “ui” folder. The code structure (which is available here: https://github.com/OpenIotOrg/openiot) is provided below:

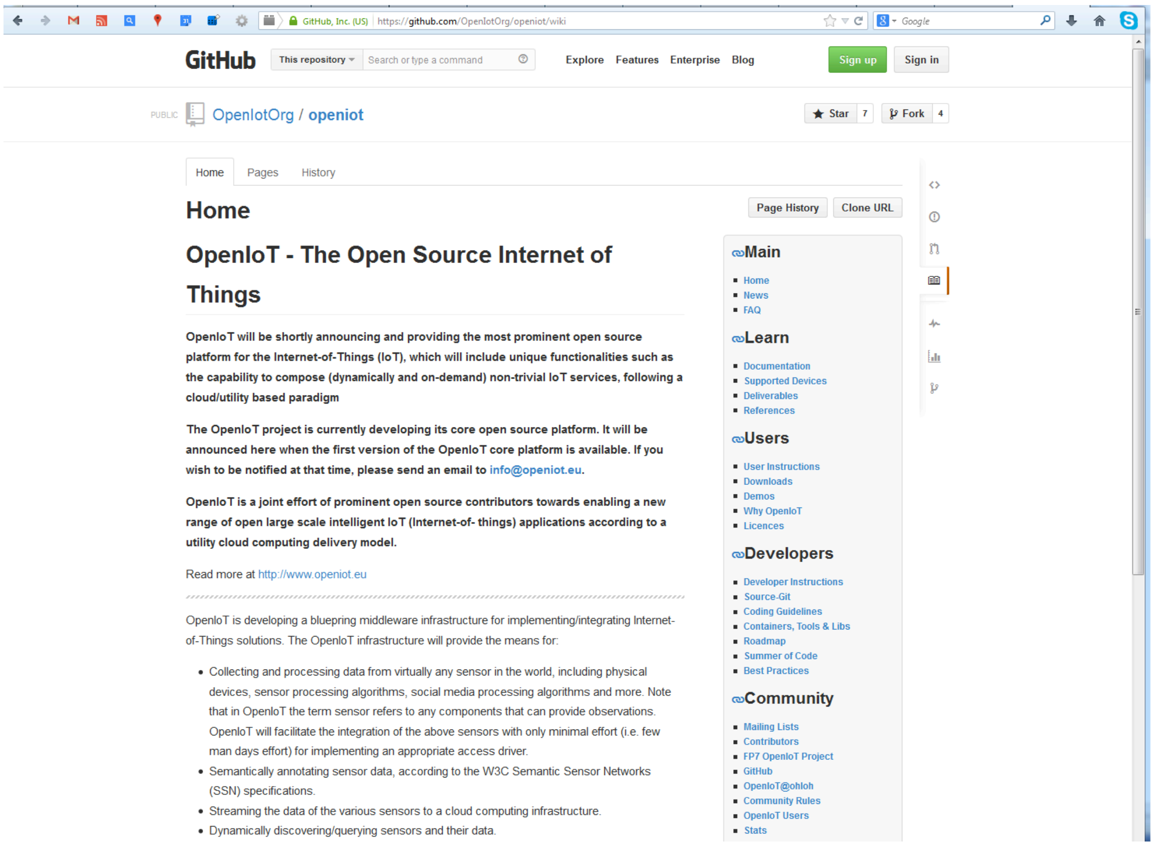
- doc: provides all the related documents with the platform.
- Modules: provides the core modules of the platform
    - x-gsn
    - scheduler
        - scheduler.core
        - scheduler.client
    - sdum
        - sdum.core
        - sdum.client
    - lsm-light
        - lsm-light.client
        - lsm-light.server
    - security
        - security-client
        - security-server
- sandbox: provides space for developers to store their test code/apps (developers “playground”).
- ui: provides all the modules related to the User Interface
    - ui.requestDefinition
    - ui.requestPresentation
    - Integrated Development Environment (IDE)
    - RDFSensorSchemaEditor
- utils: provides utilities related with the platform
    - demoData
    - lib
    - utils.commons

3.1.3 Binaries Availability

OpenIoT binaries are available through the OpenIoT Wiki site[6] under the Users>Downloads[7] section. They follow the versioning of the stable releases and are currently in the alpha stage. They are available as standalone executables per module that can be downloaded separately or in groups where one can download the complete platform in one zip file. Currently only a manual[8] installation option is available but an auto-configured option will be investigated in future releases (i.e. VirtualBox[9] with everything preinstalled).

3.2 Documentation (Wiki)

The OpenIoT project will use the GitHub Wiki (Figure 3). This wiki uses gollum[10]. Gollum is a simple wiki system built on top of Git that powers GitHub Wikis.



Home

OpenIoT - The Open Source Internet of Things

OpenIoT will be shortly announcing and providing the most prominent open source platform for the Internet-of-Things (IoT), which will include unique functionalities such as the capability to compose (dynamically and on-demand) non-trivial IoT services, following a cloud/utility based paradigm

The OpenIoT project is currently developing its core open source platform. It will be announced here when the first version of the OpenIoT core platform is available. If you wish to be notified at that time, please send an email to info@openiot.eu.

OpenIoT is a joint effort of prominent open source contributors towards enabling a new range of open large scale intelligent IoT (Internet-of- things) applications according to a utility cloud computing delivery model.

Read more at http://www.openiot.eu

OpenIoT is developing a bluepring middleware infrastructure for implementing/integrating Internet-of-Things solutions. The OpenIoT infrastructure will provide the means for:

- Collecting and processing data from virtually any sensor in the world, including physical devices, sensor processing algorithms, social media processing algorithms and more. Note that in OpenIoT the term sensor refers to any components that can provide observations. OpenIoT will facilitate the integration of the above sensors with only minimal effort (i.e. few man days effort) for implementing an appropriate access driver.
- Semantically annotating sensor data, according to the W3C Semantic Sensor Networks (SSN) specifications.
- Streaming the data of the various sensors to a cloud computing infrastructure.
- Dynamically discovering/querying sensors and their data.

Main
- Home
- News
- FAQ

Learn
- Documentation
- Supported Devices
- Deliverables
- References

Users
- User Instructions
- Downloads
- Demos
- Why OpenIoT
- Licences

Developers
- Developer Instructions
- Source-Git
- Coding Guidelines
- Containers, Tools & Libs
- Roadmap
- Summer of Code
- Best Practices

Community
- Mailing Lists
- Contributors
- FP7 OpenIoT Project
- GitHub
- OpenIoT@ohloh
- Community Rules
- OpenIoT Users
- Stats

Figure 3. Home screenshot of OpenIoT Github repository

[6] https://github.com/OpenIotOrg/openiot/wiki

[7] https://github.com/OpenIotOrg/openiot/wiki/Downloads

[8] https://github.com/OpenIotOrg/openiot/wiki/User-Instructions

[9] https://www.virtualbox.org/

[10] https://github.com/gollum/gollum#readme

Gollum wikis are simply Git repositories (OpenIoT is available here: https://github.com/OpenIotOrg/openiot/wiki) that adhere to a specific format. Gollum pages may be written in a variety of formats and can be edited in a number of ways depending on your needs. You can edit your wiki locally:

- With your favourite text editor or IDE (changes will be visible after committing).
- With the built-in web interface.
- With the Gollum Ruby API.

Gollum follows the rules of Semantic Versioning and uses TomDoc[11] for inline documentation.

Gollum useful links:

- Demos: https://github.com/mojombo/gollum-demo
- Documentation: https://github.com/gollum/gollum/blob/master/README.md
- Markdown Cheatsheet: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
- GitHub Markup: https://github.com/github/markup

3.2.1 Wiki[12] Structure

The Wiki has an appropriate structure intended to address the needs of users, developers and contributors. This structure is indicated in the following paragraphs.

Main

- **Home**: Directs to the Wiki home page
- **News**: Directs to the OpenIoT news
- **FAQ**: Directs to the OpenIoT Frequently Asked Questions list

Learn

- **Documentation**: Directs to an OpenIoT wiki page with an architecture figure enhanced with HTML image map redirecting to the component's documentation. The documentation menu provides information on the following topics/subtopics:
    - OpenIoT Architecture
    - Scheduler
        - Use
        - Develop

[11] http://tomdoc.org/

[12] https://github.com/OpenIotOrg/openiot/wiki

    - Service Delivery & Utility Manager
        - Use
        - Develop
    - Data Platform (LSM)
        - Use
            - Sensor Data
            - Functional Data
        - Develop
            - Sensor Data
            - Functional Data
    - X-GSN (Extended Global Sensor Network)
        - Use
        - Develop
    - User Interfaces
        - Request Definition
            - Use
            - Develop
        - Request Presentation
            - Use
            - Develop
        - Virtual Sensor RDF Schema Editor
            - Use
            - Develop
        - Services Monitoring
            - Use
            - Develop
    - Deliverables
    - Glossary
    - Demos
- **Supported devices**: Provides a list with supported sensors.
- **Deliverables**: Provides a list of available public deliverables and links to download them
- **References**: Provides a list of papers/publications authored from OpenIoT consortium within the context of the OpenIoT project.

Users

- **User Instructions**: Provide a link to an OpenIoT wiki page with an architecture figure enhanced with HTML image map redirecting to the component's **User** documentation header
- **Downloads**: Provide directions on how/where to download the platform
- **Demos**: Directs to the Wiki Demos page where different demos will be provided demonstrating the functionality of the different core components. Moreover demonstrations of complete complex or simple scenarios will also be provided.
- **Why OpenIoT?** : Lists why someone should use OpenIoT platform (benefits).
- **Licenses**: Provide the list of licenses OpenIoT uses (for software and documents).

Developers

- **Developer Instructions**: Provide a link to an OpenIoT wiki page with an architecture figure enhanced with HTML image map redirecting to the component's **Developer** documentation header
- **Source-Git Access**: Provide instructions on how to download the source code (HTTPS, SSH)
- **Coding guidelines**: Provides directions on how the OpenIoT code should be authored
- **Containers, Tools & Libs:** Provides instructions on how containers, tools and libs are used in OpenIoT
- **Roadmap**: Provide a roadmap for future OpenIoT development steps (list of features with importance and dates)
- **Summer of code**: Provides a list of programming tasks (simple or complex) that students could work on and may provide to the project.
- **Best Practices:** Provides a set of good practices on how to use the OpenIoT platform and how to develop solutions with it.

Community

- **Mailing Lists**: List of the provided OpenIoT open mailing lists (Users/Developers/Info/Commit) and directions on how to register/unregister
- **Contributors**: List of the OpenIoT contributors (name/affiliation/e-mail/URL)
- **FP7 OpenIoT Project**: Links to the OpenIoT project official site.
- **GitHub**: Links to the OpenIoT project GitHub organization
- **OpenIoT@ohloh**: Links to the OpenIoT ohloh page

- **Rules**: Provides the OpenIoT community rules and structure (Managers/Users/Developers/Testers.).
- **OpenIoT Users**: Provides a list of known OpenIoT users (companies or individuals).
- **Stats:** Statistics regarding OpenIoT source code.

Other

- **Logos**: Provide a list of available OpenIoT Logos
- **Posters**: Provide a list of available OpenIoT Posters
- **Photo Album**: Provides a list of available OpenIoT related photos (Meetings, Conferences, Presentations, Demonstrations)
- **Videos**: Provide a list of available OpenIoT related videos (Scenarios presentation, Demonstrations, …)
- **Useful Links**: Provide a list of Useful Links

3.3 Source Code

3.3.1 Git Repository

The OpenIoT repository is hosted at the GitHub[13] which can be found at the following link: https://github.com/OpenIotOrg/openiot . The OpenIoT repository is divided in branches. The branches are divided in two thematic categories. One is the Documentation (i.e. Site storage hosted at the “gh-pages”) and the other is the OpenIoT source code branch. Under the source code category various branches will exist, the two main categories are:

- Main branches with an infinite lifetime:
  - Master branch
  - Develop branch
- Supporting branches:
  - Feature branches
  - Release branches
  - Hotfix branches

[13] http://github.com/

3.4 Mailing Lists

OpenIoT project uses two Google groups as open mailing lists: one is for Developers and another one is for Users. One can register by following the links provided in **Table 1** below.

Table 1 OpenIoT Mailing Lists

List	**Description**	**Link**
openiot (openiot@googlegroups.com)	General mailing list	https://groups.google.com/d/forum/openiot
openiot-dev (openiot-dev@googlegroups.com)	Developer's mailing list for technical discussions	https://groups.google.com/d/forum/openiot-dev

In case someone don't have and don't want to create a Google account so as to register to the OpenIoT list the following steps below can be followed to register:

1. Type the following in your browser Nav bar.
    - for “openiot” mailing list: http://groups.google.com/openiot/mojolicious/boxsubscribe?email=your-email-address-here
    - for “openiot-dev” mailing list: http://groups.google.com/openiot-dev/mojolicious/boxsubscribe?email=your-email-address-here

2. Replace your-email-address-here with your actual email address. Press Enter. This will link you to the Group and you’ll see this Message: Almost there! In order to confirm your request to join .... Until then you can...
3. Check your email. You should have received one from the Administrator. Just follow the instructions…

Moreover OpenIoT uses a private google group for the internal developers mailing list (openiot-dev-internal@googlegroups.com) where anyone to join needs to be authorised to join. This mailing list is used for OpenIoT FP7 project internal development issues.

4 OPENIOT OSS DEVELOPERS SUPPORT

4.1 Who is contributing to OpenIoT

All OpenIoT consortium partners are contributing to development of OpenIoT software platform in various ways. In particular,

- EPFL acts as master(s) for the part of the project that concerns the lower-level sensor/ICO information acquisition and filtering, notably on the basis of the enhancements to be realized on top of the Global Sensor Networks (GSN) middleware. EPFL has contributed the XGSN component, as well as a security & privacy component which is currently under development.
- DERI acts as master(s) for the part of the project dealing with semantic annotation of sensors/ICO and related discovery and filtering processes. DERI has contributed the LSM – Linked Sensor Middleware component.
- AIT acts as master(s) for the part of the project dealing with service requests formulation and delivery. AIT has contributed the scheduler component as well as the utility module.
- IOSB acts as a coordinator of Smart Cities use cases and are responsible for developing a mapping of CampusGuide use case on OpenIoT software platform.
- CSIRO acts as a contributor to platform integration and is also responsible for the Digital Agriculture (Phenonet) use case. CSIRO also assists other partners in software development.
- SENSAP acts as a contributor to use case development and mapping the Intelligent Manufacturing scenario onto OpenIoT software platform, as well as acting as testers for the integrated OpenIoT software platform.
- AL acts as a contributor to use case development and mapping the Silver Angel scenario onto OpenIoT middleware, as well as acting as testers for the integrated OpenIoT software platform.
- FER acts as a contributor to mobile publish-subscribe middleware development as well as developing the Urban Crowdsourcing platform and Air Quality monitoring use case.

4.2 Developers Instructions

The developers can find instructions on how to download, integrate with Eclipse and develop with the different OpenIoT modules source code here: https://github.com/OpenIotOrg/openiot/wiki/Developer-Instructions .

4.2.1 System requirements

To build OpenIoT projects is Java 7.0 (Java SDK 1.7) or later, Maven 3.0 or later. Most of the applications OpenIoT project produces are designed to be run on JBoss Enterprise Application Platform 6 or JBoss AS 7.1.

4.2.2 Deploy from the source code

If you have not yet done so, you must Configure Maven before testing the scheduler deployment. After that:

- Start the JBoss Enterprise Application Platform 6 or JBoss AS 7.1 with the Web Profile
  1. Open a command line and navigate to the root of the JBoss server directory.
  2. The following shows the command line to start the server with the web profile:
     - For Linux: JBOSS_HOME/bin/standalone.sh
     - For Windows: JBOSS_HOME\bin\standalone.bat

- Build and Deploy the module
  - NOTE: The following build command assumes you have configured your Maven user settings. If you have not, you must include Maven setting arguments on the command line.
  1. Make sure you have started the JBoss Server as described above.
  2. Open a command line and navigate to the root directory of the module's Project.
  3. Type this command to build and deploy the archive:
     - mvn clean package jboss-as:deploy
  4. This will deploy target/MODULE_NAME.war to the running instance of the server.

- Access the application
  - The application will be running at the following URL: http://localhost:8080/MODULE_NAME/.

- Un-deploy the Archive
  1. Make sure you have started the JBoss Server as described above.
  2. Open a command line and navigate to the root directory of the module's Project.
  3. When you are finished testing, type this command to undeploy the archive:
     - mvn jboss-as:undeploy.

4.2.3 Run in Eclipse

You can start JBoss Application Server and deploy the OpenIoT modules from Eclipse using JBoss tools. Detailed instructions on how to integrate and start JBoss AS from Eclipse with JBoss Tools are available at the following link: https://docs.jboss.org/author/display/AS7/Starting+JBoss+AS+from+Eclipse+with+JBoss+Tools

4.2.3.1 Integrating and deploying Scheduler

To integrate and deploy the OpenIoT modules in Eclipse one should follow the steps below:

1. Import Existing maven project “File>Import>Maven>Existing Maven Projects”
2. Click the “Browse” button and navigate to the module’s source code directory that has been previously downloaded.
3. Choose the module’s directory and click the Finish button.
4. Right click on the “MODULE_NAME” project and choose “Run As>Maven Build…”
5. Insert the following to:
    a. Goals: “clean package jboss-as:deploy”
    b. Profiles: “arq-jbossas-remote”
    c. Name: “MODULE_NAME package-deploy” (or your preferred name)
6. Click the Run button (the JBoss Server should be already running). The project will automatically build itself, get deployed and run at the JBoss AS running instance. From now on this configuration should be available at the Eclipse Run Configurations under Maven Build.

To un-deploy the module from the running instance of the JBoss AS follow the steps below:

1. Right click on the “MODULE_NAME” project and choose “Run As>Maven Build…”
2. Insert the following to:
    a. Goals: “jboss-as:undeploy”
    b. Profiles: “arq-jbossas-remote”
    c. Name: “scheduler.core undeploy” (or your preferred name)

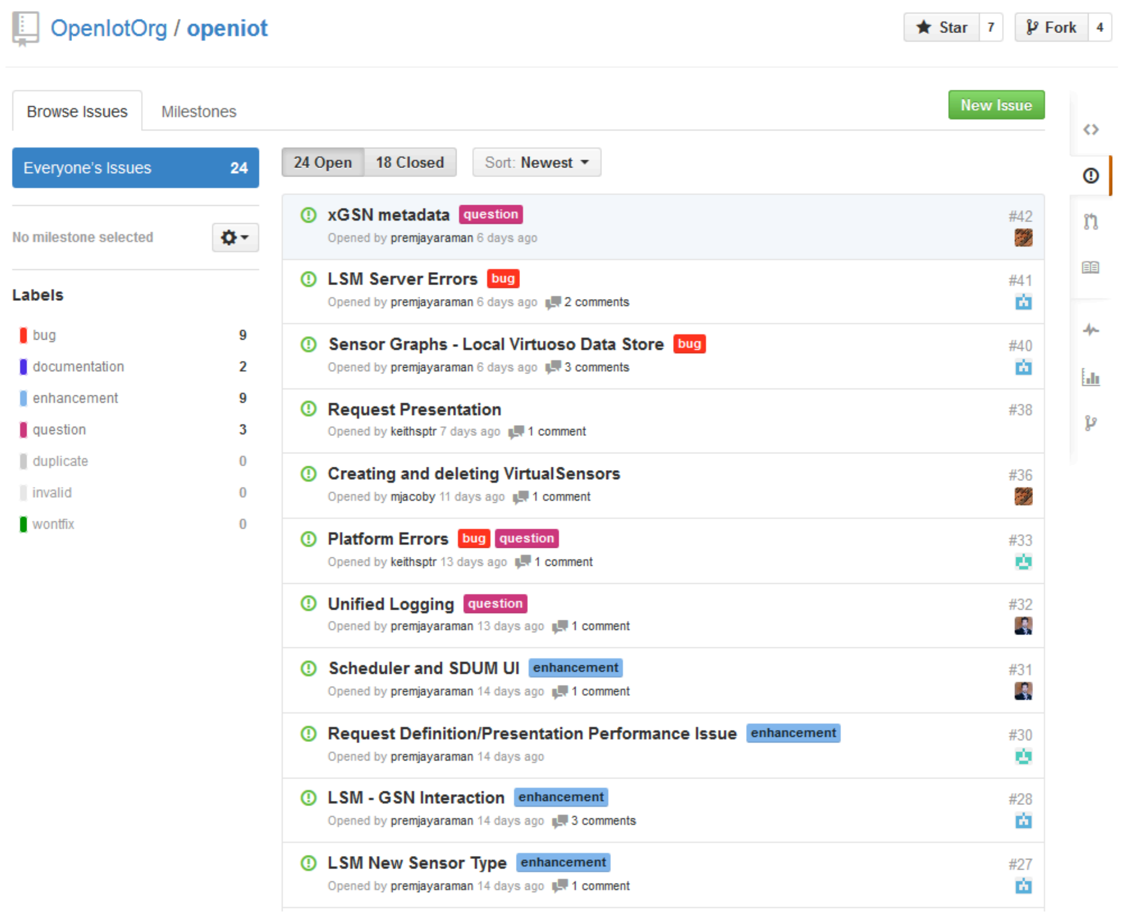
Click the Run button (the JBoss Server should be already running). The project will automatically be un-deployed from the JBoss AS running instance. From now on this configuration should be available at the Eclipse Run Configurations under Maven Build.

4.3 Issues Tracked

OpenIoT uses the GitHub issue tracker[14] utility (Figure 4) to tack issues related with the OpenIoT development process. For every reported issue different labels are used depending on the issue type. These labels are:

- Bug: used to report bugs,
- Documentation: used to report missing or incorrect documentation,
- Enhancement: used to report the need of an enhancement,
- Question: used to report questions,
- Duplicate: used to report duplicate issues,
- Invalid: used to mark an issue as invalid, and
- Wontfix: used to mark an issue that won't be fixed.

So far 42 issues have been reported of which 24 are open and 18 are closed as illustrated below.



Labels

bug	9
documentation	2
enhancement	9
question	3
duplicate	0
invalid	0
wontfix	0

xGSN metadata question #42
Opened by premjayaraman 6 days ago

LSM Server Errors bug #41
Opened by premjayaraman 6 days ago 2 comments

Sensor Graphs - Local Virtuoso Data Store bug #40
Opened by premjayaraman 6 days ago 3 comments

Request Presentation #38
Opened by keithsptr 7 days ago 1 comment

Creating and deleting VirtualSensors #36
Opened by mjacoby 11 days ago 1 comment

Platform Errors bug question #33
Opened by keithsptr 13 days ago 1 comment

Unified Logging question #32
Opened by premjayaraman 13 days ago 1 comment

Scheduler and SDUM UI enhancement #31
Opened by premjayaraman 14 days ago 1 comment

Request Definition/Presentation Performance Issue enhancement #30
Opened by premjayaraman 14 days ago

LSM - GSN Interaction enhancement #28
Opened by premjayaraman 14 days ago 3 comments

LSM New Sensor Type enhancement #27
Opened by premjayaraman 14 days ago 1 comment

Figure 4. OpenIoT Github Repository Issue Tracker Screenshot

[14] https://github.com/OpenIotOrg/openiot/issues

4.4 Coding Guidelines

4.4.1 Namespaces

4.4.1.1 Trunk and branches

Sources must be prefixed by the following namespaces.

- Java: `org.openiot`
- Bundles symbolic names: `org.openiot`
- JMX ObjectNames for end-to-end management: `org.openiot:type=component,name=concern`
- XML (XML Schema) : `urn:org:openiot`

4.4.1.2 Sandboxes

- Java : `org.openiot.sandbox`
- Bundles symbolic names : `org.openiot.sandbox`
- XML (XML Schema) : `urn:org:openiot`

4.4.1.3 Sub project group and maven artifact ids

Names and ids must be prefixed by the following namespaces.

- Group ID: `org.openiot`
- Artifact Id: `[module].[functionality]` (i.e. scheduler.core)
- Bundle Name : `[groupId].[artifactId]`
- Project Name: `[artifactId]`

4.4.1.4 Files

Two mandatory files (NOTICE and LICENSE) must be provided in each project directory

- NOTICE contains the description of the dependencies and their licenses.
- LICENSE contains the license text.

Those files should automatically added in the artifact after the package phase

4.4.1.4.1 NOTICE file

Table 3 provides an example of a Notice File:

Table 2: Notice File example

```
Copyright 2011-2014 OpenIoT Project
https://github.com/OpenIotOrg/openiot/wiki

I. Included Software

This product includes software developed at
The OpenIoT Consortium (http://openiot.eu/).
Licensed under the LGPL v3.0.

II. Used Software

This product uses software developed at
the FooBar project (http://code.foo.org/bar).
licensed under the Apache License 2.0.

This product uses software developed at
the TicTac project (http://code.tic.org/tac).
licensed under the LGPL v2.1.

III. License Summary
- LGPL v3.0
```

4.4.1.4.2 LICENSE file

A copy of the license file that should be provided with every OpenIoT library distribution Details are provided below.

4.4.1.5 Beginning comments

All source files (Java, XML, ...) must begin with the comments shown in the following code sample below at Table 3.

Table 3: Beginning Comments example

```
/**
* Copyright (c) 2011-2014, OpenIoT
*
* …………License Disclaimer……………..
*
* Contact: OpenIoT mailto: info@openiot.eu
*/

/**
*
*<File Documentation/Description>
*
* @author <AuthorName> (<author alias>) e-mail:
                        <author_e-mail>@mailserver.com
*
*/
```

4.4.2 License Disclaimers

OpenIoT uses LGPL 3.0 so it should provide these terms and conditions . To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found (Table 4)

Table 4: License disclaimer example

```
/**
* Copyright (c) 2011-2014, OpenIoT
*
* This file is part of OpenIoT.
*
* OpenIoT is free software: you can redistribute it and/or modify
* it under the terms of the GNU Lesser General Public License as published by
* the Free Software Foundation, version 3 of the License.
*
* OpenIoT is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU Lesser General Public License for more details.
*
* You should have received a copy of the GNU Lesser General Public License
* along with OpenIoT.  If not, see <http://www.gnu.org/licenses/>.
*
* Contact: OpenIoT mailto: info@openiot.eu
*/
```

If you have copied code from other programs covered by the same license, copy their copyright notices too. Put all the copyright notices together, right near the top of each file.

You should also include a copy of the license itself in the distribution of your program (attached) usually in the first folder of your project, usually in a file called “COPYING.LESSER.txt”[15]. Please note that, since the LGPL is a set of additional permissions on top of the GPL, it's important to include both licenses so users have all the materials they need to understand their rights “COPYING.txt”[16]

[15] https://raw.github.com/wiki/OpenIotOrg/openiot/files/COPYING.LESSER.txt

[16] https://raw.github.com/wiki/OpenIotOrg/openiot/files/COPYING.txt

4.5 External collaborators

4.5.1 Joint Copyright Assignment

In the case an individual would like to become a contributor to the OpenIoT project he should provide a signed joint copyright assignment document to the OpenIoT project. A sample of the Joint Copyright Assignment is provided in Table 5 below:

Table 5: Joint Copyright Assignment Sample

OpenIoT Open Source Project
Joint Copyright Assignment by Contributor To OpenIoT

Contact Information (the "Contributor")
Full name:
E-mail:
Mailing address:
Telephone:
Facsimile:
Country:

1. Contributor owns, and has sufficient rights to contribute, all source code and related material intended to be compiled or integrated with the source code for the OpenIoT open source product (the "Contribution") which Contributor has ever delivered, and OpenIoT consortium has accepted, for incorporation into the technology made available under the OpenIoT open source project.
2. Contributor hereby assigns to OpenIoT consortium joint ownership in all worldwide common law and statutory rights associated with the copyrights, copyright application, copyright registration and moral rights in the Contribution to the extent allowable under applicable local laws and copyright conventions. Contributor agrees that this assignment may be submitted by OpenIoT consortium to register a copyright in the Contribution. Contributor retains the right to use the Contribution for Contributor's own purposes. This Joint Copyright Assignment supersedes and replaces all prior copyright assignments made by Contributor to OpenIoT consortium under the OpenIoT project.
3. Contributor is legally entitled to grant the above assignment and agrees not to provide any Contribution that violates any law or breaches any contract.

Signed: Date:
Printed name:

The Contributor should email a scanned signed original of this assignment to: info@openiot.eu

4.6 Example of OpenIoT platform setup, deployment and integration

4.6.1 Introduction

4.6.1.1 Purpose of the section

This document is primarily aimed at developers/engineers, with no previous OpenIoT project experience. The document elaborated the steps involved in integrating, setting up and testing the OpenIoT components in detail.

This integrated document builds on documentation provided by the individual OpenIoT component developers. This document will supplement the information currently available in the Wiki. This document describes the steps taken and the issues encountered during setup, testing and integration of the components.

This document follows a simple step-by-step approach to enable developers/engineers and tech-savvy end user to download, implement, deploy and run the OpenIoT platform. Figure 5 provides an overview of the OpenIoT platform architecture. For more detailed information, please refer to the OpenIoT Cookbook.

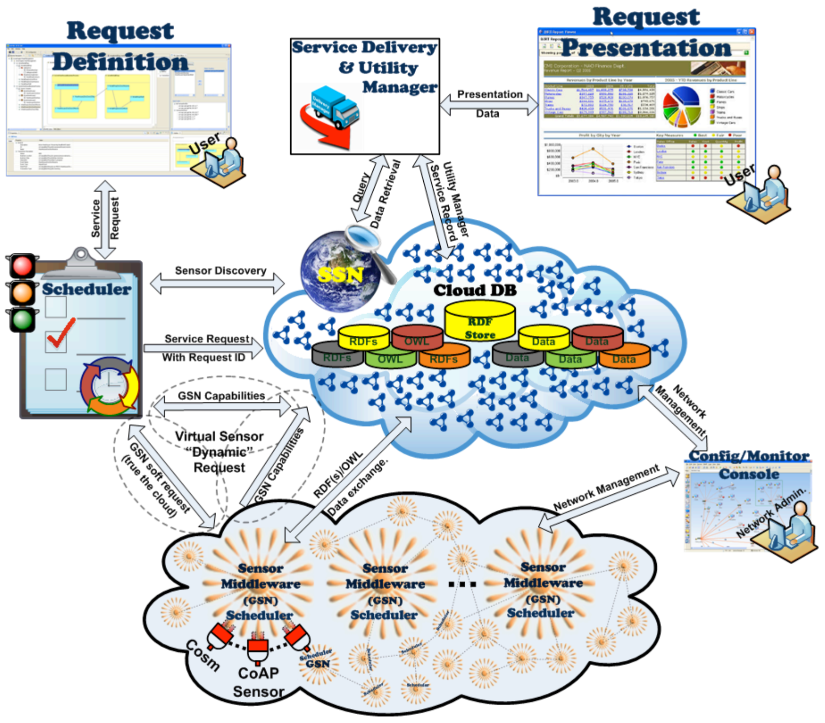


Figure 5: OpenIoT Architecture Overview (functional view)

4.6.2 Integration Environment setup

4.6.2.1 Source code location

- https://github.com/OpenIotOrg/openiot

4.6.2.2 Documentation

- https://github.com/OpenIotOrg/openiot/wiki#developers
- https://subversion.deri.ie/openiot/

4.6.2.3 Development Platform

Netbeans environment was used for its JBoss and maven integration abilities.

- Netbeans Installation procedure - https://netbeans.org/community/releases/71/install.html
- Netbeans version information -
    - Product Version: NetBeans IDE 7.4 (Build 201310111528)
    - Java: 1.7.0_45; Java HotSpot(TM) 64-Bit Server VM 24.45-b08
    - Runtime: Java(TM) SE Runtime Environment 1.7.0_45-b18

4.6.2.4 Additional components required

The following are the additional components required with the development platforms

- Maven - http://maven.apache.org/download.cgi
- JBoss - http://www.jboss.org/jbossas/downloads/

4.6.2.5 Environmental Path

- The following variables needs to be set for successful running of maven, JBOSS and java
    - JAVA_HOME
    - MAVEN_HOME
- The following path needs to be set
    - Java bin directory
    - Maven bin directory

  The following is a useful link on how to setup maven.

  http://www.mkyong.com/maven/how-to-install-maven-in-windows/

4.6.3 netbeans setup with OpenIoT code

4.6.3.1 Open the projects

- Choose file->open project
- Select the directory of the each OpenIoT Module.
    - Under UI Folder
        - ui.requestCommons
        - ui.requestDefinition
        - ui.requestPresentation
        - ide/ide.core

- Under modules
  - Scheduler
    - scheduler.core
    - scheduler.client
  - SDUM
    - sdum.core
    - sdum.client
  - LSM Lite (If this gives error, don't worry. For now ignore LSM as we will use DERI's LSM server. I will update the document when I successfuly install and configure LSM locally).
  - x-gsn
- Under utils
  - Utils.commons

4.6.3.2 Pre-requisites

- Install the lsm-lib into the local maven repository.
  - Under develop\utils\lib\lsmApiLibraryMavenInstall
  - Run the bat file to install the lib to your local maven repository (ensure maven is installed and configured before running this step).

4.6.3.3 Creation of jar and war files from source code

The procedure to compile and build the source code into Jar and war files is outlined here (https://netbeans.org/kb/docs/java/quickstart.html#build). (run-> clean and build). The following source code needs to be compiled into jars in the same order mentioned.

- x-gsn (you should see a build success message)
- utils.commons
- ui.requestCommons
- ui.requestDefinition
- ui.requestPresentation
- ide.core (under ui/ide)
- Scheduler (under modules)
  - scheduler.core
  - scheduler.client
- SDUM
  - sdum.core
  - sdum.client

The clean and build operation will generate the corresponding jar and war files. The war and jar files are under target folder. In netbeans, change to the files tab in the project explorer window to view the target folder.

note: if you get an error “javac: invalid target release: 1.7”, please open the pom file and change the tag value of the <maven.compiler.target> from 1.7 to 1.6.

4.6.3.3.1 JBoss and Netbeans integration

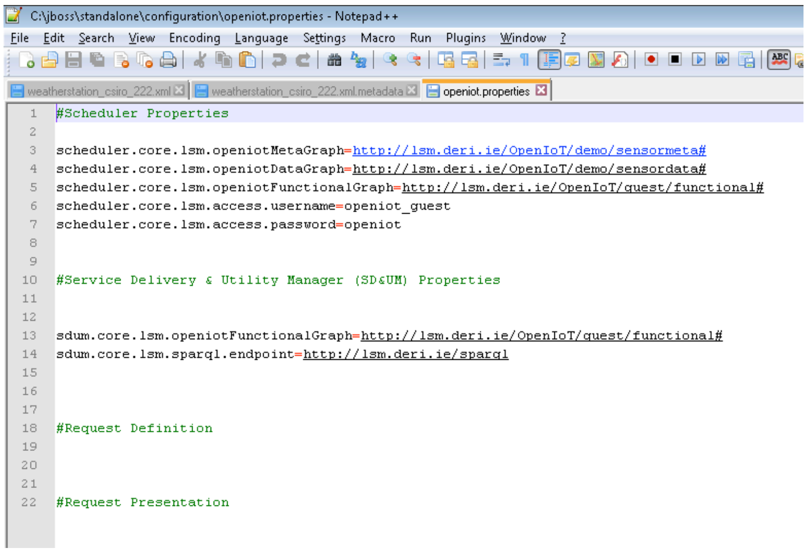
The following link shows how to integrate JBoss into the Netbeans.

http://developinjava.com/articles/using-jboss-as-7-with-netbeans/

4.6.3.3.2 Moving the configuration properties file to jBoss location

The properties file was required to be copied from source to the destination location below to enable the web-server to use the properties [Figure 6].

- Source: \\openiot-develop\utils\utils.commons\src\main\resources\properties\
- Destination: \\jboss\standalone\configuration


C:\jboss\standalone\configuration\openiot.properties - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
weatherstation_csiro_222.xml
weatherstation_csiro_222.xml.metadata
openiot.properties

```
#Scheduler Properties

scheduler.core.lsm.openiotMetaGraph=http://lsm.deri.ie/OpenIoT/demo/sensormeta#
scheduler.core.lsm.openiotDataGraph=http://lsm.deri.ie/OpenIoT/demo/sensordata#
scheduler.core.lsm.openiotFunctionalGraph=http://lsm.deri.ie/OpenIoT/guest/functional#
scheduler.core.lsm.access.username=openiot_guest
scheduler.core.lsm.access.password=openiot


#Service Delivery & Utility Manager (SD&UM) Properties


sdum.core.lsm.openiotFunctionalGraph=http://lsm.deri.ie/OpenIoT/guest/functional#
sdum.core.lsm.sparql.endpoint=http://lsm.deri.ie/sparql



#Request Definition



#Request Presentation
```

Figure 6. Web Server OpenIoT Properties

4.6.4 Deployment Start-up sequence

The following is the sequence in which the main software modules were deployed as a web service to JBOSS previously configured in netbeans.

If you use a standalone JBOSS install, the war files in the target folders needs to be copied to JBOSS installation directory **(standalone/deployments)**. Refer to the OpenIoT Wiki sections on user documentation for more details.

1. Scheduler – Web Service
2. SDUM – Web Service
3. LSM Lite – Web Service

4. Request Definition – Web Service
5. Request Presentation - Web Service
6. IDE Core - Web Service

4.6.5 Running the openiot platform

To run the platform, we first need to configure xGSN so that we have some live data being pushed into LSM. The current documentation is limited to pushing data to LSM server running on DERI. Once we have a successful xGSN setup, we can run the Request Definition and Request Presentation user interfaces to compose a new service.

4.6.5.1 xGSN

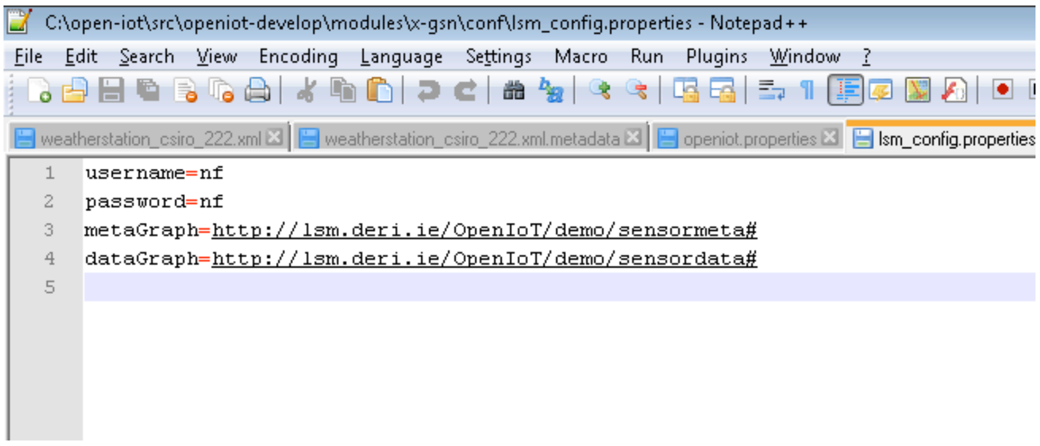
The first component setup was xGSN. This was to ensure that there was sensor data available to be used later on by the request application

4.6.5.2 Wiki reference

- https://github.com/OpenIotOrg/openiot/wiki/X-GSN-Use

4.6.5.3 LSM configuration properties

The following **Figure 7** illustrates the LSM configuration properties used

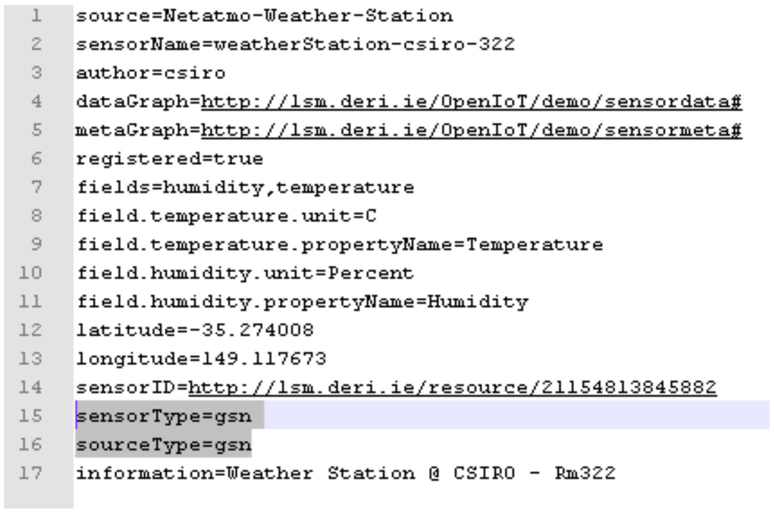

C:\open-iot\src\openiot-develop\modules\x-gsn\conf\lsm_config.properties - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
weatherstation_csiro_222.xml
weatherstation_csiro_222.xml.metadata
openiot.properties
lsm_config.properties

```
username=nf
password=nf
metaGraph=http://lsm.deri.ie/OpenIoT/demo/sensormeta#
dataGraph=http://lsm.deri.ie/OpenIoT/demo/sensordata#
```

Figure 7. LSM Configuration properties

4.6.5.4 Creating virtual sensor xml file

During tested it was observed that the only “SensorType” and “SourceType” property that worked was “GSN”. This also leads to the issue discussed later on all sensor sources are visible on the request definition map as sensor type GSN. This makes it difficult to differentiate between sources as shown in **Figure 8**.

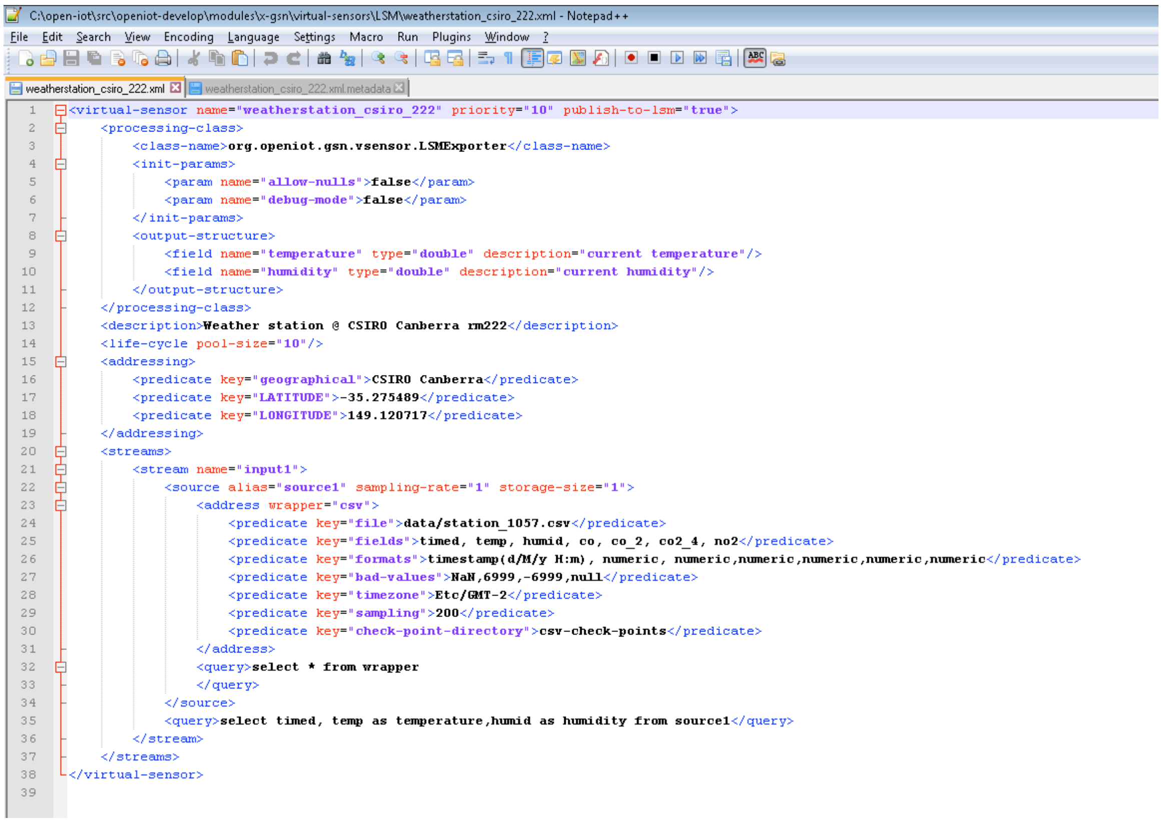


```
source=Netatmo-Weather-Station
sensorName=weatherStation-csiro-322
author=csiro
dataGraph=http://lsm.deri.ie/OpenIoT/demo/sensordata#
metaGraph=http://lsm.deri.ie/OpenIoT/demo/sensormeta#
registered=true
fields=humidity,temperature
field.temperature.unit=C
field.temperature.propertyName=Temperature
field.humidity.unit=Percent
field.humidity.propertyName=Humidity
latitude=-35.274008
longitude=149.117673
sensorID=http://lsm.deri.ie/resource/21154813845882
sensorType=gsn
sourceType=gsn
information=Weather Station @ CSIRO - Rm322
```

Figure 8. Metatdata Description of Virtual Sensor in XML

4.6.5.5 *Creating virtual sensor metadata file*

For testing purposes, a CSV file was used as the source for test data to upload to LSM. This can be seen in the wrapper below in **Figure 9**. The CSV file is in the GSN folder under data directory

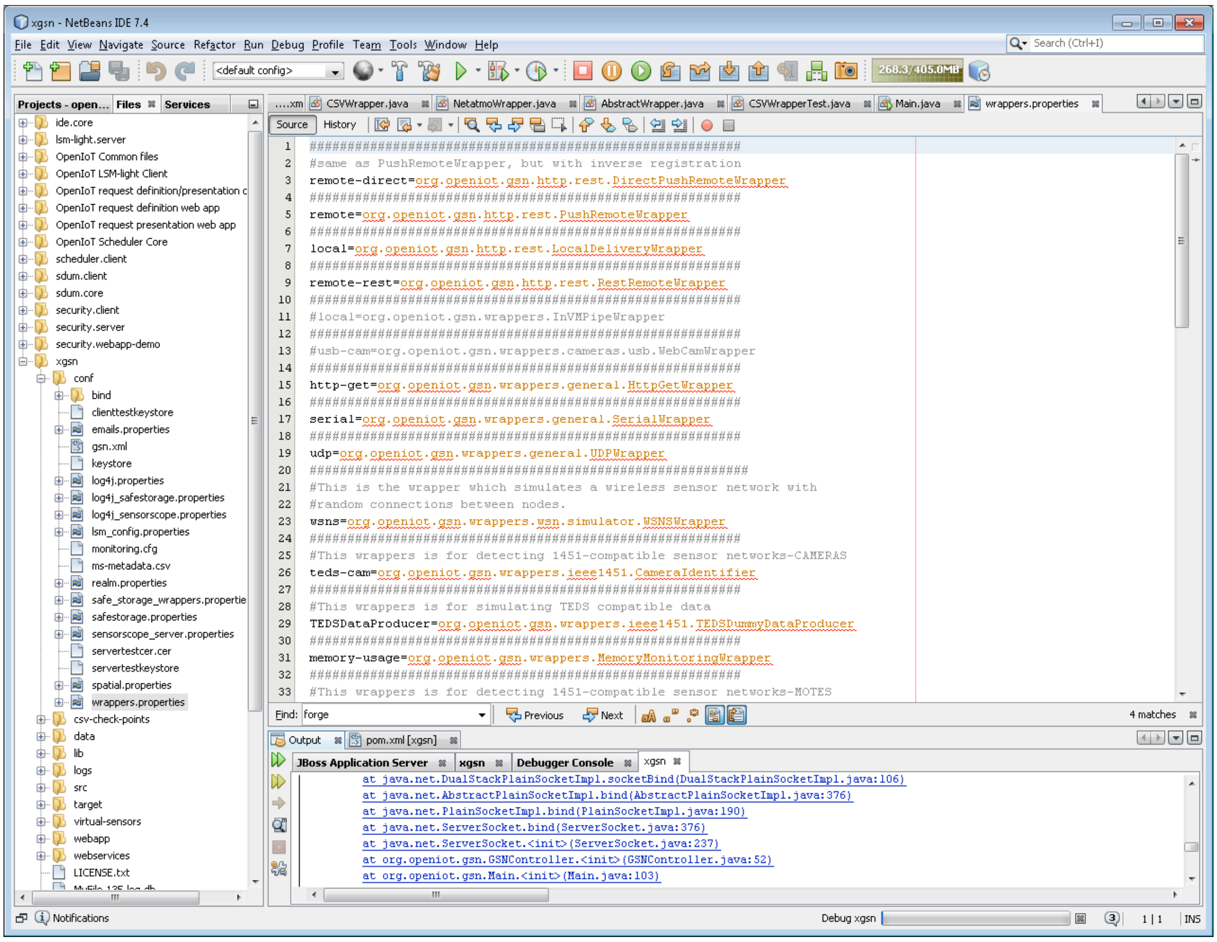


C:\open-iot\src\openiot-develop\modules\x-gsn\virtual-sensors\LSM\weatherstation_csiro_222.xml - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
weatherstation_csiro_222.xml
weatherstation_csiro_222.xml.metadata

```
<virtual-sensor name="weatherstation_csiro_222" priority="10" publish-to-lsm="true">
    <processing-class>
        <class-name>org.openiot.gsn.vsensor.LSMExporter</class-name>
        <init-params>
            <param name="allow-nulls">false</param>
            <param name="debug-mode">false</param>
        </init-params>
        <output-structure>
            <field name="temperature" type="double" description="current temperature"/>
            <field name="humidity" type="double" description="current humidity"/>
        </output-structure>
    </processing-class>
    <description>Weather station @ CSIRO Canberra rm222</description>
    <life-cycle pool-size="10"/>
    <addressing>
        <predicate key="geographical">CSIRO Canberra</predicate>
        <predicate key="LATITUDE">-35.275489</predicate>
        <predicate key="LONGITUDE">149.120717</predicate>
    </addressing>
    <streams>
        <stream name="input1">
            <source alias="source1" sampling-rate="1" storage-size="1">
                <address wrapper="csv">
                    <predicate key="file">data/station_1057.csv</predicate>
                    <predicate key="fields">timed, temp, humid, co, co_2, co2_4, no2</predicate>
                    <predicate key="formats">timestamp(d/M/y H:m), numeric, numeric,numeric,numeric,numeric,numeric</predicate>
                    <predicate key="bad-values">NaN,6999,-6999,null</predicate>
                    <predicate key="timezone">Etc/GMT-2</predicate>
                    <predicate key="sampling">200</predicate>
                    <predicate key="check-point-directory">csv-check-points</predicate>
                </address>
                <query>select * from wrapper
                </query>
            </source>
            <query>select timed, temp as temperature,humid as humidity from source1</query>
        </stream>
    </streams>
</virtual-sensor>
```

Figure 9. Virtual Sensor Description

4.6.6 Setting up new sensor in wrappers.properties (Figure 10)



```
#########################################################
#same as PushRemoteWrapper, but with inverse registration
remote-direct=org.openiot.gsn.http.rest.DirectPushRemoteWrapper
#########################################################
remote=org.openiot.gsn.http.rest.PushRemoteWrapper
#########################################################
local=org.openiot.gsn.http.rest.LocalDeliveryWrapper
#########################################################
remote-rest=org.openiot.gsn.http.rest.RestRemoteWrapper
#########################################################
#local=org.openiot.gsn.wrappers.InVMPipeWrapper
#########################################################
#usb-cam=org.openiot.gsn.wrappers.cameras.usb.WebCamWrapper
#########################################################
http-get=org.openiot.gsn.wrappers.general.HttpGetWrapper
#########################################################
serial=org.openiot.gsn.wrappers.general.SerialWrapper
#########################################################
udp=org.openiot.gsn.wrappers.general.UDPWrapper
##########################################################
#This is the wrapper which simulates a wireless sensor network with
#random connections between nodes.
wsns=org.openiot.gsn.wrappers.wsn.simulator.WSNSWrapper
#########################################################
#This wrappers is for detecting 1451-compatible sensor networks-CAMERAS
teds-cam=org.openiot.gsn.wrappers.ieee1451.CameraIdentifier
#########################################################
#This wrappers is for simulating TEDS compatible data
TEDSDataProducer=org.openiot.gsn.wrappers.ieee1451.TEDSDummyDataProducer
#########################################################
memory-usage=org.openiot.gsn.wrappers.MemoryMonitoringWrapper
#########################################################
#This wrappers is for detecting 1451-compatible sensor networks-MOTES
```

```
        at java.net.DualStackPlainSocketImpl.socketBind(DualStackPlainSocketImpl.java:106)
        at java.net.AbstractPlainSocketImpl.bind(AbstractPlainSocketImpl.java:376)
        at java.net.PlainSocketImpl.bind(PlainSocketImpl.java:190)
        at java.net.ServerSocket.bind(ServerSocket.java:376)
        at java.net.ServerSocket.<init>(ServerSocket.java:237)
        at org.openiot.gsn.GSNController.<init>(GSNController.java:52)
        at org.openiot.gsn.Main.<init>(Main.java:103)
```

Figure 10. wapper.properties file

4.6.6.1 Registering new sensor with LSM – through xGSN

The wiki provides documentation on how to register a sensor using the existing batch file in the xGSN folder. However due to issues with class path length, the code within the batch file was re-written to the following (**Figure 11**).

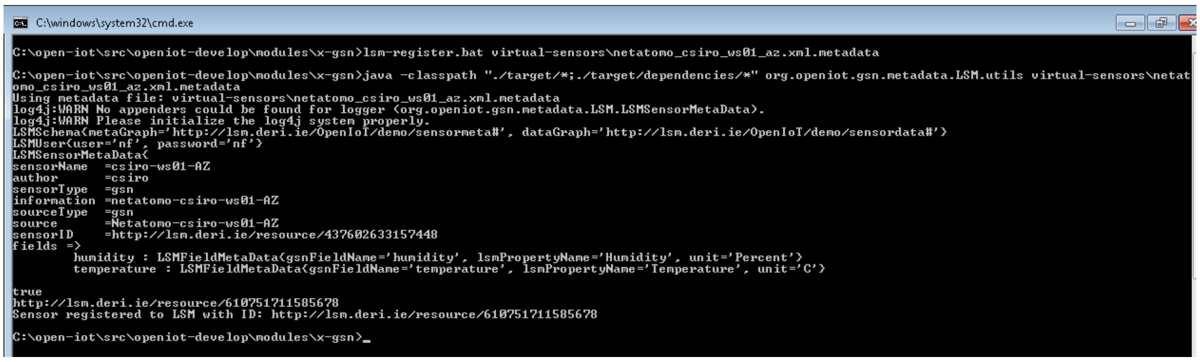
```

java -classpath "./target/*;./target/dependencies/*" org.openiot.gsn.metadata.LSM.utils %1

```

Figure 11. Sensor registration

The following command is executed against the file.

- *"lsm-register.bat virtual-sensors\netatomo_csiro_ws01_az.xml.metadata"*

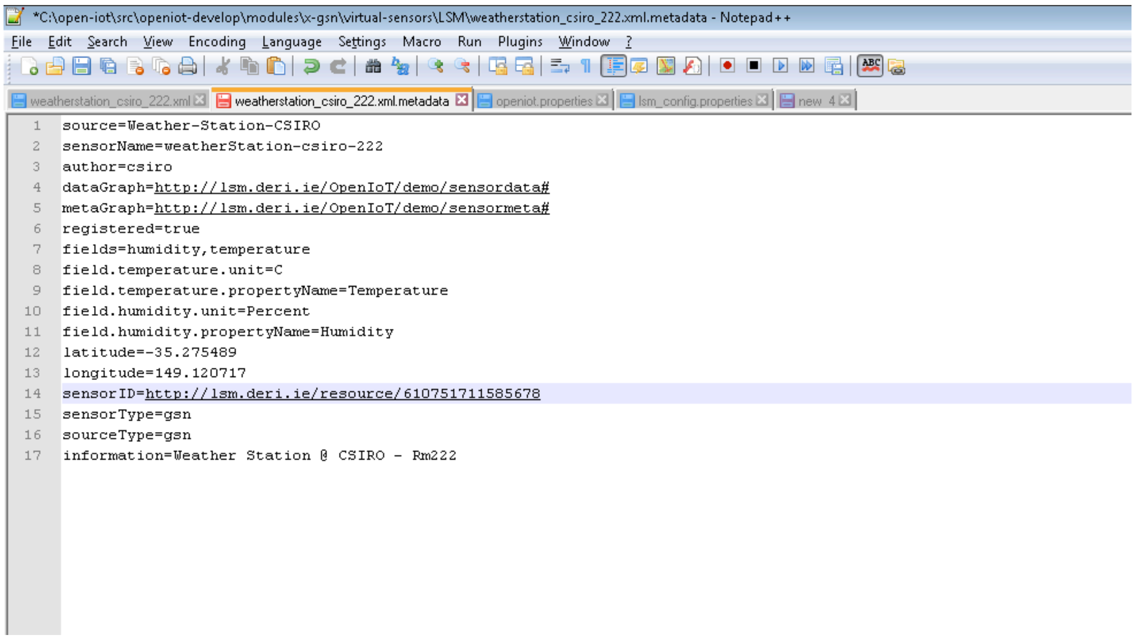
When the metadata configuration file is passed to this method we have the following results (**Figure 12**)



```
C:\open-iot\src\openiot-develop\modules\x-gsn>lsm-register.bat virtual-sensors\netatomo_csiro_ws01_az.xml.metadata

C:\open-iot\src\openiot-develop\modules\x-gsn>java -classpath "./target/*;./target/dependencies/*" org.openiot.gsn.metadata.LSM.utils virtual-sensors\netat
omo_csiro_ws01_az.xml.metadata
Using metadata file: virtual-sensors\netatomo_csiro_ws01_az.xml.metadata
log4j:WARN No appenders could be found for logger (org.openiot.gsn.metadata.LSM.LSMSensorMetaData).
log4j:WARN Please initialize the log4j system properly.
LSMSchema{metaGraph='http://lsm.deri.ie/OpenIoT/demo/sensormeta#', dataGraph='http://lsm.deri.ie/OpenIoT/demo/sensordata#'}
LSMUser{user='nf', password='nf'}
LSMSensorMetaData{
sensorName  =csiro-ws01-AZ
author      =csiro
sensorType  =gsn
information =netatomo-csiro-ws01-AZ
sourceType  =gsn
source      =Netatomo-csiro-ws01-AZ
sensorID    =http://lsm.deri.ie/resource/437602633157448
fields =>
        humidity : LSMFieldMetaData{gsnFieldName='humidity', lsmPropertyName='Humidity', unit='Percent'}
        temperature : LSMFieldMetaData{gsnFieldName='temperature', lsmPropertyName='Temperature', unit='C'}

true
http://lsm.deri.ie/resource/610751711585678
Sensor registered to LSM with ID: http://lsm.deri.ie/resource/610751711585678

C:\open-iot\src\openiot-develop\modules\x-gsn>_
```

Figure 12. Sensor registration Successful

Once this is complete the new sensor ID is provided by the LSM registration process and the user must update the metadata file with the ID (**Figure 13**).


*C:\open-iot\src\openiot-develop\modules\x-gsn\virtual-sensors\LSM\weatherstation_csiro_222.xml.metadata - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
weatherstation_csiro_222.xml
weatherstation_csiro_222.xml.metadata
openiot.properties
lsm_config.properties
new 4

```
source=Weather-Station-CSIRO
sensorName=weatherStation-csiro-222
author=csiro
dataGraph=http://lsm.deri.ie/OpenIoT/demo/sensordata#
metaGraph=http://lsm.deri.ie/OpenIoT/demo/sensormeta#
registered=true
fields=humidity,temperature
field.temperature.unit=C
field.temperature.propertyName=Temperature
field.humidity.unit=Percent
field.humidity.propertyName=Humidity
latitude=-35.275489
longitude=149.120717
sensorID=http://lsm.deri.ie/resource/610751711585678
sensorType=gsn
sourceType=gsn
information=Weather Station @ CSIRO - Rm222
```

Figure 13. Change sensor ID

4.6.6.2 Running xGSN from Command Prompt

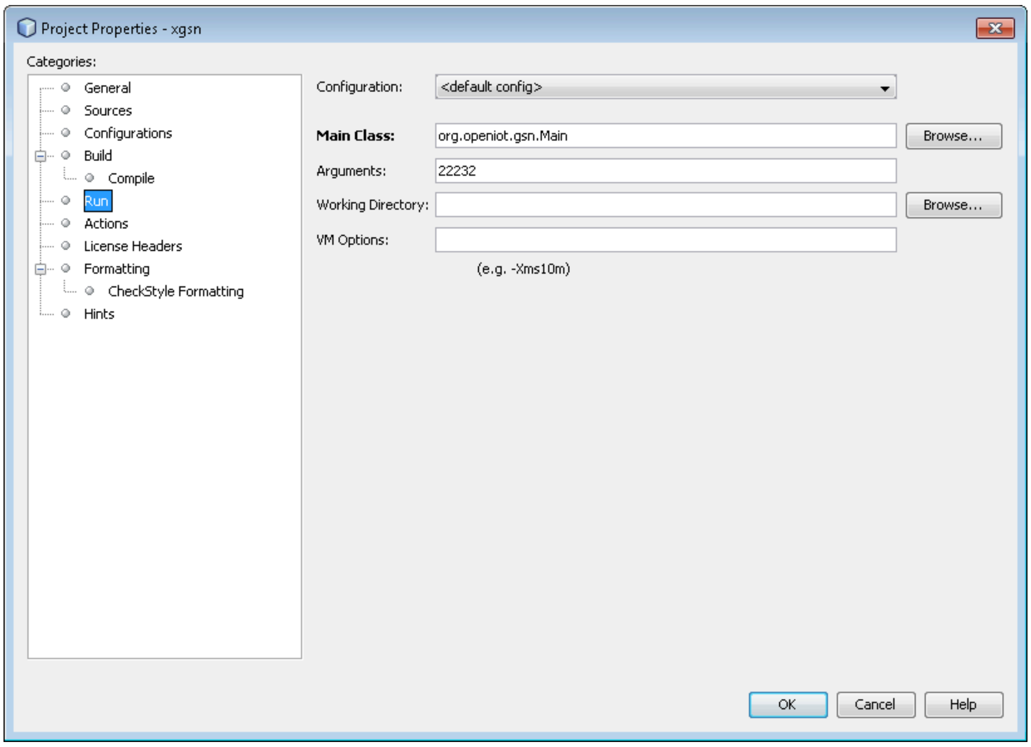
To run xGSN from command prompt (windows), please update the entire .bat file with the following java –classpath "./target.*;./target/dependencies/*" -splash:lib/logo.png -Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger -Dorg.mortbay.log.LogFactory.noDiscovery=false org.openiot.gsn.Main 22232

The following command is executed to start xGSN

gsn-start.bat

4.6.6.3 Debugging / Running xGSN through development environment

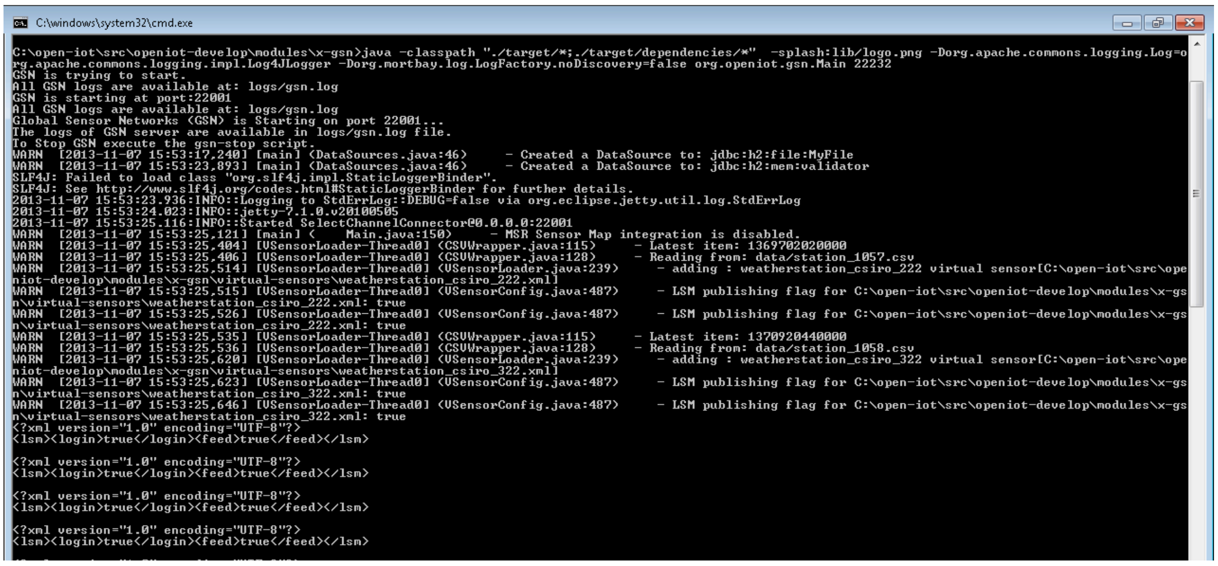
To run the Main method within the project, the port number on which the server will be set up on needs to be passed as arguments (**Figure 14**).



Configuration: <default config>
Main Class: org.openiot.gsn.Main
Arguments: 22232
Working Directory:
VM Options:

Figure 14. Run/Debug Setup

When executed the following debug (**Figure 15**) shows a successful pushing of data to the LSM cloud



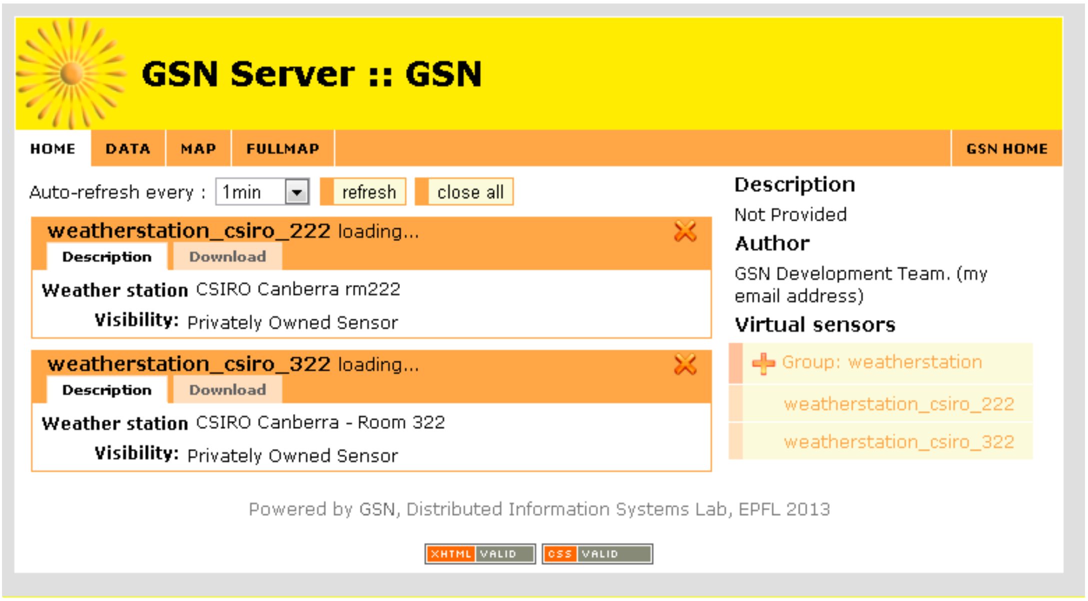
C:\windows\system32\cmd.exe

```
C:\open-iot\src\openiot-develop\modules\x-gsn>java -classpath "./target/*;./target/dependencies/*"  -splash:lib/logo.png -Dorg.apache.commons.logging.Log=o
rg.apache.commons.logging.impl.Log4JLogger -Dorg.mortbay.log.LogFactory.noDiscovery=false org.openiot.gsn.Main 22232
GSN is trying to start.
All GSN logs are available at: logs/gsn.log
GSN is starting at port:22001
All GSN logs are available at: logs/gsn.log
Global Sensor Networks (GSN) is Starting on port 22001...
The logs of GSN server are available in logs/gsn.log file.
To Stop GSN execute the gsn-stop script.
WARN  [2013-11-07 15:53:17,240] [main] (DataSources.java:46)     - Created a DataSource to: jdbc:h2:file:MyFile
WARN  [2013-11-07 15:53:23,893] [main] (DataSources.java:46)     - Created a DataSource to: jdbc:h2:mem:validator
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
2013-11-07 15:53:23.936:INFO::Logging to StdErrLog::DEBUG=false via org.eclipse.jetty.util.log.StdErrLog
2013-11-07 15:53:24.023:INFO::jetty-7.1.0.v20100505
2013-11-07 15:53:25.116:INFO::Started SelectChannelConnector@0.0.0.0:22001
WARN  [2013-11-07 15:53:25,121] [main] (    Main.java:150)     - MSR Sensor Map integration is disabled.
WARN  [2013-11-07 15:53:25,404] [VSensorLoader-Thread0] (CSVWrapper.java:115)     - Latest item: 1369702020000
WARN  [2013-11-07 15:53:25,406] [VSensorLoader-Thread0] (CSVWrapper.java:128)     - Reading from: data/station_1057.csv
WARN  [2013-11-07 15:53:25,514] [VSensorLoader-Thread0] (VSensorLoader.java:239)     - adding : weatherstation_csiro_222 virtual sensor[C:\open-iot\src\ope
niot-develop\modules\x-gsn\virtual-sensors\weatherstation_csiro_222.xml]
WARN  [2013-11-07 15:53:25,515] [VSensorLoader-Thread0] (VSensorConfig.java:487)     - LSM publishing flag for C:\open-iot\src\openiot-develop\modules\x-gs
n\virtual-sensors\weatherstation_csiro_222.xml: true
WARN  [2013-11-07 15:53:25,526] [VSensorLoader-Thread0] (VSensorConfig.java:487)     - LSM publishing flag for C:\open-iot\src\openiot-develop\modules\x-gs
n\virtual-sensors\weatherstation_csiro_222.xml: true
WARN  [2013-11-07 15:53:25,535] [VSensorLoader-Thread0] (CSVWrapper.java:115)     - Latest item: 1370920440000
WARN  [2013-11-07 15:53:25,536] [VSensorLoader-Thread0] (CSVWrapper.java:128)     - Reading from: data/station_1058.csv
WARN  [2013-11-07 15:53:25,620] [VSensorLoader-Thread0] (VSensorLoader.java:239)     - adding : weatherstation_csiro_322 virtual sensor[C:\open-iot\src\ope
niot-develop\modules\x-gsn\virtual-sensors\weatherstation_csiro_322.xml]
WARN  [2013-11-07 15:53:25,623] [VSensorLoader-Thread0] (VSensorConfig.java:487)     - LSM publishing flag for C:\open-iot\src\openiot-develop\modules\x-gs
n\virtual-sensors\weatherstation_csiro_322.xml: true
WARN  [2013-11-07 15:53:25,646] [VSensorLoader-Thread0] (VSensorConfig.java:487)     - LSM publishing flag for C:\open-iot\src\openiot-develop\modules\x-gs
n\virtual-sensors\weatherstation_csiro_322.xml: true
<?xml version="1.0" encoding="UTF-8"?>
<lsm><login>true</login><feed>true</feed></lsm>

<?xml version="1.0" encoding="UTF-8"?>
<lsm><login>true</login><feed>true</feed></lsm>

<?xml version="1.0" encoding="UTF-8"?>
<lsm><login>true</login><feed>true</feed></lsm>

<?xml version="1.0" encoding="UTF-8"?>
<lsm><login>true</login><feed>true</feed></lsm>
```

Figure 15. successful logging of data

4.6.6.4 *User interface*

The following **Figure 16** illustrates the user interface available for xGSN. The interface seems not provide any functionality to enable the user to see data being logged



GSN Server :: GSN

weatherstation_csiro_222 loading...

Weather station CSIRO Canberra rm222

Visibility: Privately Owned Sensor

weatherstation_csiro_322 loading...

Weather station CSIRO Canberra - Room 322

Visibility: Privately Owned Sensor

Description

Not Provided

Author

GSN Development Team. (my email address)

Virtual sensors

Group: weatherstation

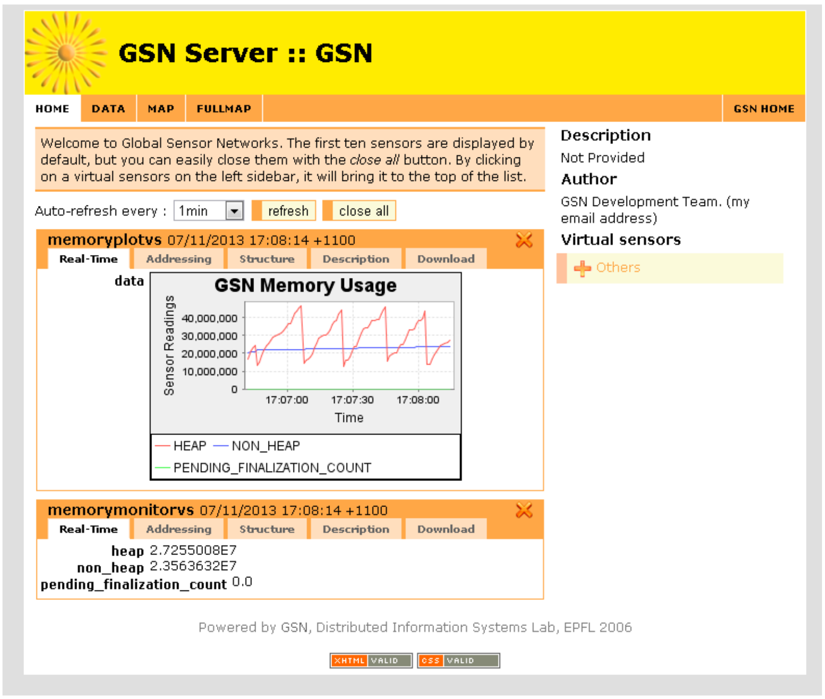
weatherstation_csiro_222

weatherstation_csiro_322

Powered by GSN, Distributed Information Systems Lab, EPFL 2013

Figure 16. xGSN Interface

The normal GSN application, on the other hand, provides an interface to view the logged data. Having such an interface (**Figure 17**) could perhaps make it much simpler for the debugging stage of setup.



GSN Server :: GSN

HOME DATA MAP FULLMAP GSN HOME

Welcome to Global Sensor Networks. The first ten sensors are displayed by default, but you can easily close them with the *close all* button. By clicking on a virtual sensors on the left sidebar, it will bring it to the top of the list.

Auto-refresh every : 1min refresh close all

memoryplotvs 07/11/2013 17:08:14 +1100

Real-Time Addressing Structure Description Download

data

memorymonitorvs 07/11/2013 17:08:14 +1100

Real-Time Addressing Structure Description Download

heap	2.7255008E7
non_heap	2.3563632E7
pending_finalization_count	0.0

Description
Not Provided
Author
GSN Development Team. (my email address)
Virtual sensors
Others

Powered by GSN, Distributed Information Systems Lab, EPFL 2006

Figure 17. GSN Interface

Note: Figure 15 is GSN not xGSN. xGSN has been stripped of these features.

4.6.7 Scheduler Core/Client

The scheduler core is the core service that will be deployed on JBOSS. The scheduler service can be tested by opening the following URL

http://localhost:8080/scheduler.core/rest/services

The scheduler client (**Figure 18**) is a java application with Swing UI to test the services of the scheduler.core service.

The wiki / cookbook does not seem to provide any documentation on this client application. Here is a quick summary of the function. Some of the functions are still unclear.

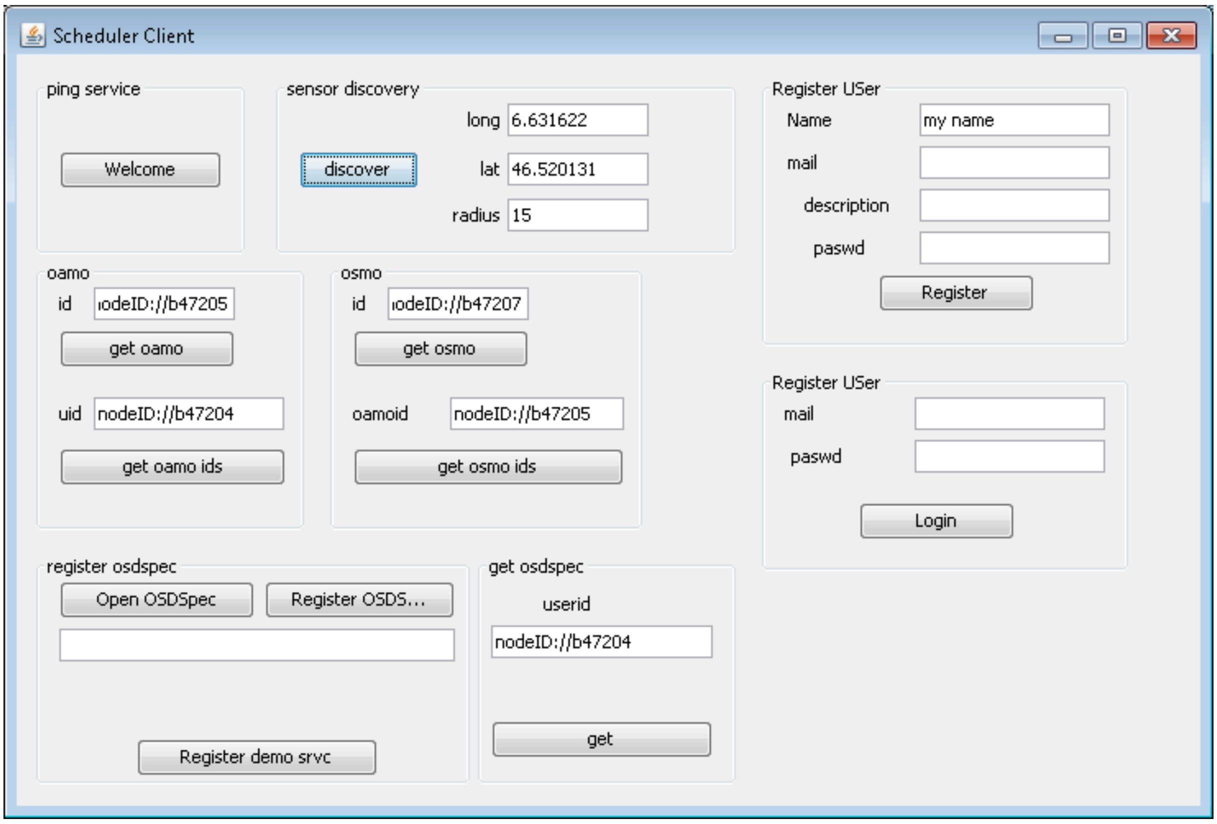
Welcome - Will ping the server to test it is running

Discover – The discover service will discover the list of sensors within the given latitude and longitude

Register user – will register a new user with LSM (the new user will be stored into the graph as indicated by the configuration property file previously copied to the JBOSS store).

Login – the second register user section will test if a user is valid by authenticating the user against LSM.

Oamo and Osmo – These functions are used to get RDF information from the LSM store.



long: 6.631622
lat: 46.520131
radius: 15
Name: my name
oamo id: ıodeID://b47205
oamo uid: nodeID://b47204
osmo id: ıodeID://b47207
osmo oamoid: nodeID://b47205
get osdspec userid: nodeID://b47204

Figure 18. Scheduler client

4.6.8 sDUM

The SDUM core is the core service that will be deployed on JBOSS. The SDUM service can be tested by opening the following URL

http://localhost:8080/sdum.core/rest/services

The sdum client (**Figure 19**) is a Java application with Swing UI to test the services of the sdum.core service.

Similar to the scheduler client application, the wiki / cookbook does not seem to provide any documentation on this client application. This makes it difficult for user to start using.

From experimentation, the following is the function of the client UI

Welcome – will test if the sdum.core service has been successfully deployed

Poll – the poll for service will take a service id and get the RDF data related to the service id

note: currently, there is no way to find the service id when using the user interfaces. The scheduler API if used directly without the request definition ui can provide the user with service id's

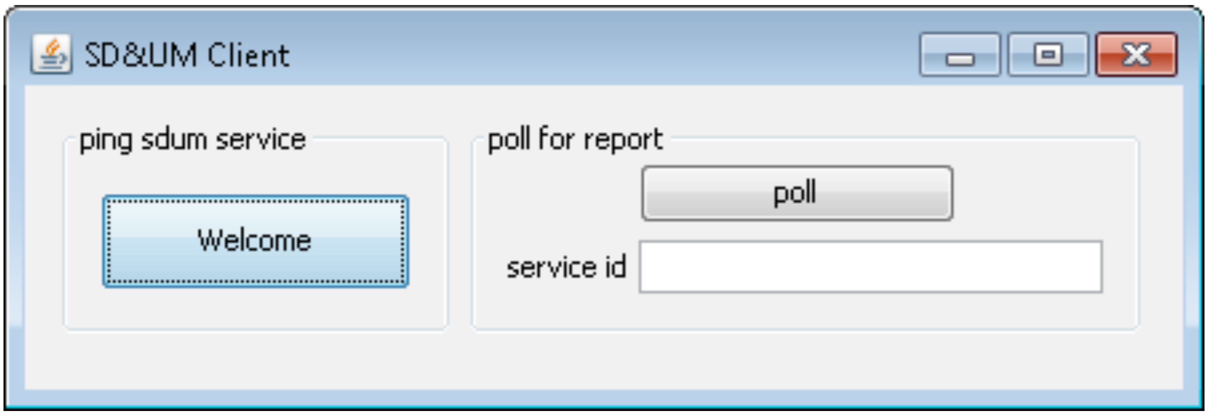


Figure 19. Sdum client

4.6.9 Request Definition

The request definition interface can be accessed from http://localhost:8080/ui.requestDefinition

1. Create a new user
2. Login with the new user
3. Compose a new service
4. Validate the design
5. Save the design

Please refer to the OpenIoT Wiki on more detailed procedures. Please explore both user and develop sections.

4.6.10 Request Presentation

The request definition interface can be accessed from http://localhost:8080/ui.requestPresentation

1. Login with same user id
2. Load your application
3. Click on current application-force dashboard refresh
4. Data should become available in the selected output format

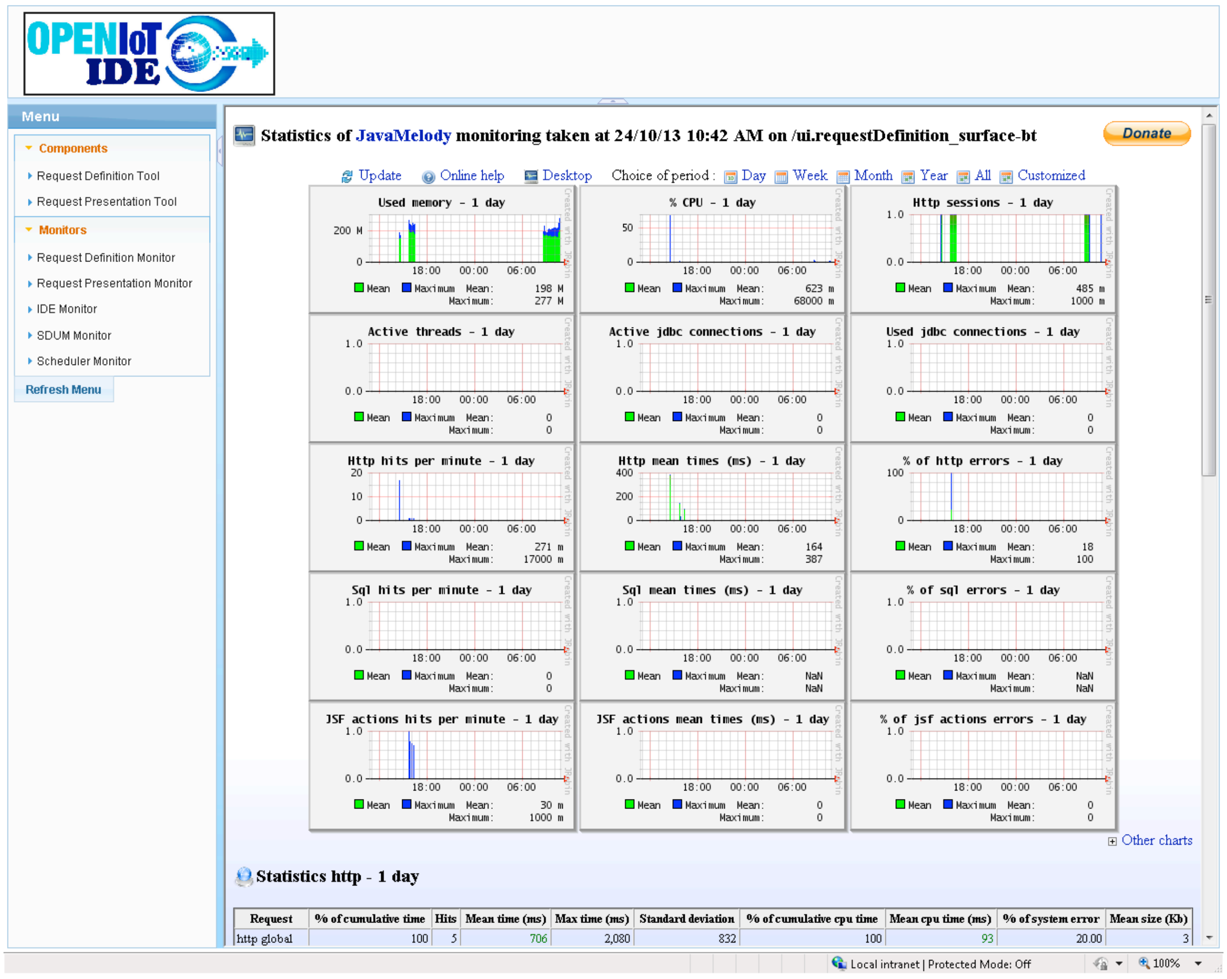
Please refer to the OpenIoT Wiki on more detailed procedures. Please explore both user and develop sections.

4.6.11 Other Interfaces – ide.core

The ide.core accessed from http://localhost:8080/ide.core is an integrated interface for the request Definition and Presentation UI along with the monitoring tools shown below.

4.6.12 Monitor Statistics – graphs

Currently there does not seem to be any documentation on this interface on the wiki (**Figure 20**) If the standard JavaMelody online help is to be used by users of the OpenIoT platform then a link from the wiki to JavaMelody online help page should be created.


OPENIoT IDE
Menu
Components
Request Definition Tool
Request Presentation Tool
Monitors
Request Definition Monitor
Request Presentation Monitor
IDE Monitor
SDUM Monitor
Scheduler Monitor
Refresh Menu
Donate
Update
Online help
Desktop
Choice of period :
Day
Week
Month
Year
All
Customized
Used memory - 1 day
Mean: 198 M Maximum: 277 M
% CPU - 1 day
Mean: 623 m Maximum: 68000 m
Http sessions - 1 day
Mean: 485 m Maximum: 1000 m
Active threads - 1 day
Mean: 0 Maximum: 0
Active jdbc connections - 1 day
Mean: 0 Maximum: 0
Used jdbc connections - 1 day
Mean: 0 Maximum: 0
Http hits per minute - 1 day
Mean: 271 m Maximum: 17000 m
Http mean times (ms) - 1 day
Mean: 164 Maximum: 387
% of http errors - 1 day
Mean: 18 Maximum: 100
Sql hits per minute - 1 day
Mean: 0 Maximum: 0
Sql mean times (ms) - 1 day
Mean: NaN Maximum: NaN
% of sql errors - 1 day
Mean: NaN Maximum: NaN
JSF actions hits per minute - 1 day
Mean: 30 m Maximum: 1000 m
JSF actions mean times (ms) - 1 day
Mean: 0 Maximum: 0
% of jsf actions errors - 1 day
Mean: 0 Maximum: 0
Other charts
Local intranet | Protected Mode: Off
100%

Statistics of JavaMelody monitoring taken at 24/10/13 10:42 AM on /ui.requestDefinition_surface-bt

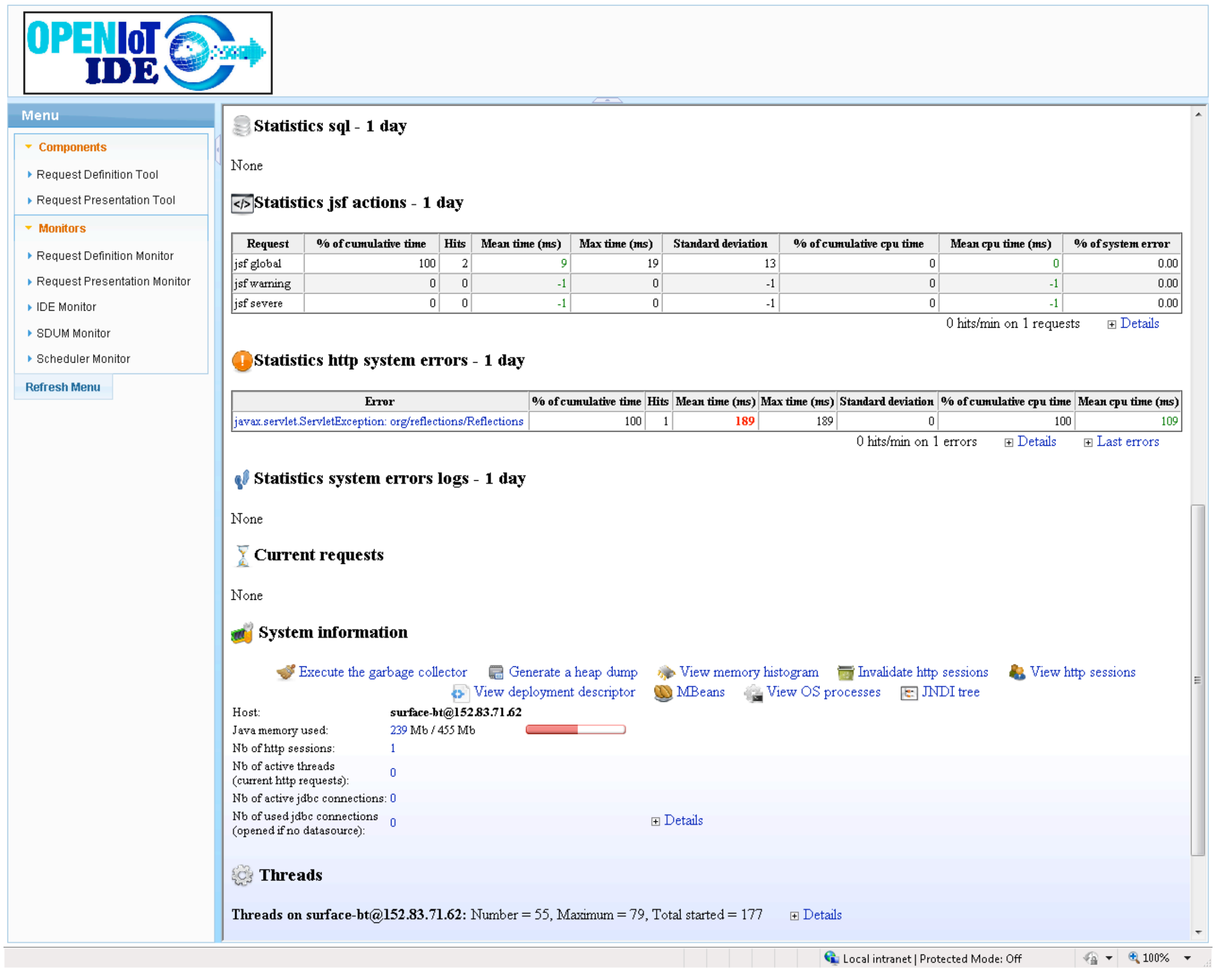
Statistics http - 1 day

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	% of system error	Mean size (Kb)
http global	100	5	706	2,080	832	100	93	20.00	3

Figure 20. Monitor statistics – graphs

4.6.12.1Monitor Statistics – detailed

As above, currently there does not seem to be any documentation on this interface on the wiki (**Figure 21**). If the standard JavaMelody online help is to be used by users of the OpenIoT platform then a link from the wiki to JavaMelody online help page should be created.



Statistics sql - 1 day

None

Statistics jsf actions - 1 day

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	% of system error
jsf global	100	2	9	19	13	0	0	0.00
jsf warning	0	0	-1	0	-1	0	-1	0.00
jsf severe	0	0	-1	0	-1	0	-1	0.00

0 hits/min on 1 requests ⊞ Details

Statistics http system errors - 1 day

Error	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)
javax.servlet.ServletException: org/reflections/Reflections	100	1	189	189	0	100	109

0 hits/min on 1 errors ⊞ Details ⊞ Last errors

Statistics system errors logs - 1 day

None

Current requests

None

System information

Host:	**surface-bt@152.83.71.62**
Java memory used:	239 Mb / 455 Mb
Nb of http sessions:	1
Nb of active threads (current http requests):	0
Nb of active jdbc connections:	0
Nb of used jdbc connections (opened if no datasource):	0

⊞ Details

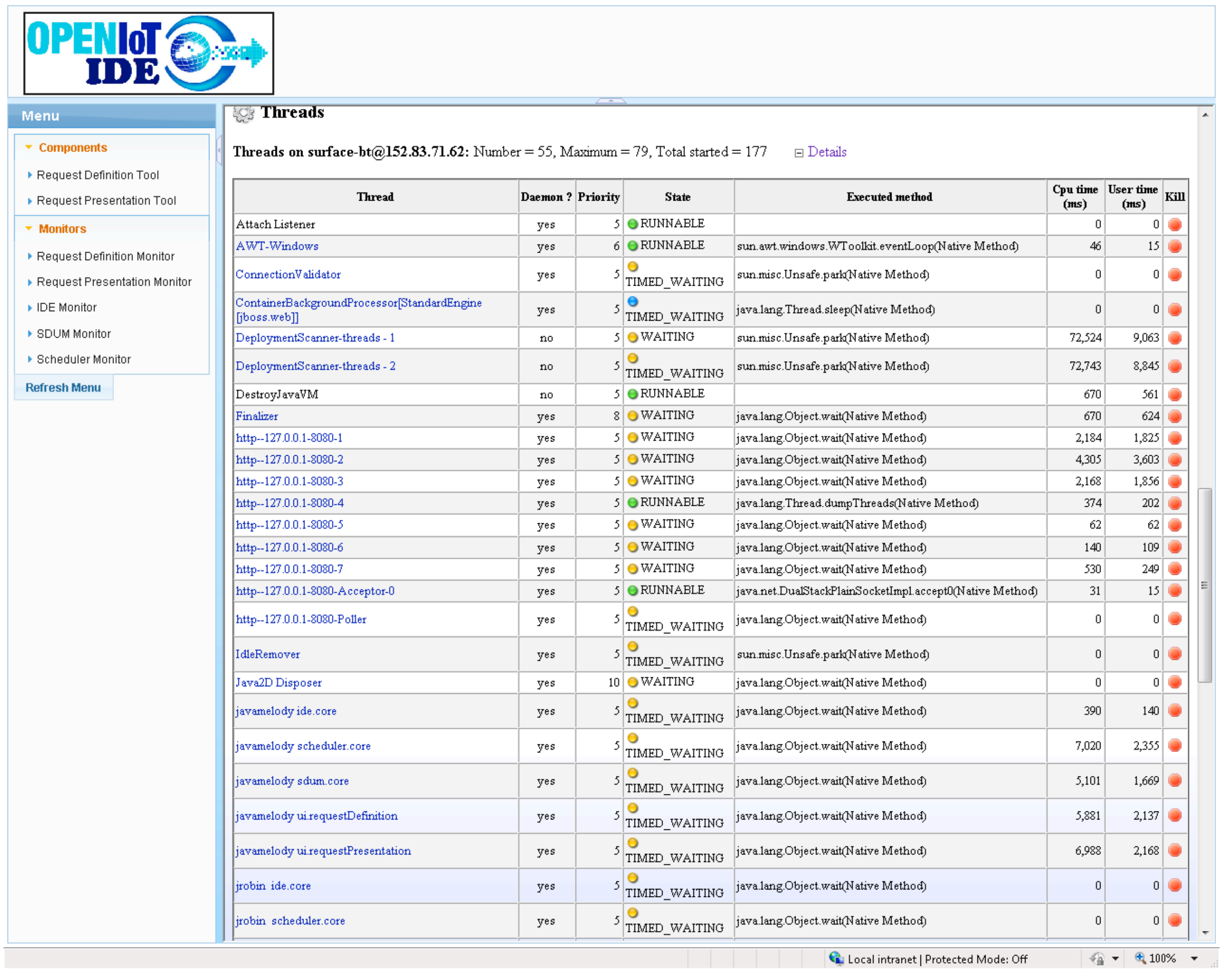
Threads

Threads on surface-bt@152.83.71.62: Number = 55, Maximum = 79, Total started = 177 ⊞ Details

Figure 21. Monitor statistics - details

4.6.12.2Monitor – Threads

As above, currently there does not seem to be any documentation on this interface on the wiki (**Figure 22**). If the standard JavaMelody online help is to be used by users of the OpenIoT platform then a link from the wiki to JavaMelody online help page should be created.


OPENIoT IDE
Menu
Components
Request Definition Tool
Request Presentation Tool
Monitors
Request Definition Monitor
Request Presentation Monitor
IDE Monitor
SDUM Monitor
Scheduler Monitor
Refresh Menu
Local intranet | Protected Mode: Off
100%

Threads

Threads on surface-bt@152.83.71.62: Number = 55, Maximum = 79, Total started = 177 ⊟ Details

Thread	Daemon ?	Priority	State	Executed method	Cpu time (ms)	User time (ms)	Kill
Attach Listener	yes	5	● RUNNABLE		0	0	●
AWT-Windows	yes	6	● RUNNABLE	sun.awt.windows.WToolkit.eventLoop(Native Method)	46	15	●
ConnectionValidator	yes	5	● TIMED_WAITING	sun.misc.Unsafe.park(Native Method)	0	0	●
ContainerBackgroundProcessor[StandardEngine [jboss.web]]	yes	5	● TIMED_WAITING	java.lang.Thread.sleep(Native Method)	0	0	●
DeploymentScanner-threads - 1	no	5	● WAITING	sun.misc.Unsafe.park(Native Method)	72,524	9,063	●
DeploymentScanner-threads - 2	no	5	● TIMED_WAITING	sun.misc.Unsafe.park(Native Method)	72,743	8,845	●
DestroyJavaVM	no	5	● RUNNABLE		670	561	●
Finalizer	yes	8	● WAITING	java.lang.Object.wait(Native Method)	670	624	●
http--127.0.0.1-8080-1	yes	5	● WAITING	java.lang.Object.wait(Native Method)	2,184	1,825	●
http--127.0.0.1-8080-2	yes	5	● WAITING	java.lang.Object.wait(Native Method)	4,305	3,603	●
http--127.0.0.1-8080-3	yes	5	● WAITING	java.lang.Object.wait(Native Method)	2,168	1,856	●
http--127.0.0.1-8080-4	yes	5	● RUNNABLE	java.lang.Thread.dumpThreads(Native Method)	374	202	●
http--127.0.0.1-8080-5	yes	5	● WAITING	java.lang.Object.wait(Native Method)	62	62	●
http--127.0.0.1-8080-6	yes	5	● WAITING	java.lang.Object.wait(Native Method)	140	109	●
http--127.0.0.1-8080-7	yes	5	● WAITING	java.lang.Object.wait(Native Method)	530	249	●
http--127.0.0.1-8080-Acceptor-0	yes	5	● RUNNABLE	java.net.DualStackPlainSocketImpl.accept0(Native Method)	31	15	●
http--127.0.0.1-8080-Poller	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	0	0	●
IdleRemover	yes	5	● TIMED_WAITING	sun.misc.Unsafe.park(Native Method)	0	0	●
Java2D Disposer	yes	10	● WAITING	java.lang.Object.wait(Native Method)	0	0	●
javamelody ide.core	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	390	140	●
javamelody scheduler.core	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	7,020	2,355	●
javamelody sdum.core	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	5,101	1,669	●
javamelody ui.requestDefinition	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	5,881	2,137	●
javamelody ui.requestPresentation	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	6,988	2,168	●
jrobin ide.core	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	0	0	●
jrobin scheduler.core	yes	5	● TIMED_WAITING	java.lang.Object.wait(Native Method)	0	0	●

Figure 22. Monitor Threads

5 USERS SUPPORT

5.1 Who is Using OpenIoT

We remind that by “users” we mean people that use the OpenIoT middleware in one way or another. Putting it differently, users *create value* for themselves by using the OpenIoT middleware and/or platform.

Example users are the OpenIoT validating use cases: PhenoNet by CSIRO; Materials Tracing by SENSAP, Smart Campus by IOSB, Urban Crowdsensing by FER and Silver Angel by AcrossLimits. These are further described in D6.3.1.

The project is actively promoting the OpenIoT released middleware to potential external users, like SME system integrators, service providers and other research groups interested in creating services based on OpenIoT. The *end-users* (the users of services created by the OpenIoT users) are not directly supported by the project.

In the next version of this deliverable (D6.2.2) all known internal and external users of the OpenIoT middleware will be listed. Below are some highlights from each internal user.

5.1.1 CampusGuide Use Case

The CampusGuide is currently using the X-GSN sensor value acquisition software and the Virtuoso RFD-Store for building the Smart Campus Ontology. Some experiments have been done with the TinyGSN version.

The LSM software has been deployed on a local server within the Fraunhofer IOSB facility and the next step will be the connection to the local GSN node.

5.1.2 Digital Agriculture use case

The digital agriculture use case is currently using the X-GSN data stream engine middleware for data collection/streaming out of the data store which contains Phenonet sample data.

The OpenIoT integrated platform has been deployed on a local server while the LSM service runs out of a DERI server. CSIRO experience in integrating and deploying the OpenIoT software platform has been described in Section 4.6.

5.1.3 SilverAngel use case

SilverAngel use case is in the process of being deployed on mobile devices and partial implementation.

5.1.4 Intelligent Manufacturing use case

The intelligent manufacturing use case is in the process of selecting OpenIoT software components for deployment and experimentation.

5.2 Users Instructions

The users find instructions that are available at the OpenIoT Wiki here: https://github.com/OpenIotOrg/openiot/wiki/User-Instructions (work in progress) to download and use the different OpenIoT modules. The binaries of the different modules can be found at the Downloading section here: https://github.com/OpenIotOrg/openiot/wiki/Downloads.

5.2.1 System requirements

To run most of OpenIoT projects you need Java 7.0 (Java SDK 1.7) or later and JBoss Enterprise Application Platform 6 or JBoss AS 7.1.
You can download those binaries through the OpenIoT Wiki[17] under the Users>Downloads[18] section.

5.2.2 Deployment/Un-deployment

5.2.2.1 JBoss AS 6.0

To run most of OpenIoT projects you need Java 7.0 (Java SDK 1.7) or later and JBoss Enterprise Application Platform 6 or JBoss AS 7.1.
You can download those binaries through the OpenIoT Wiki[19] under the Users>Downloads[20] section.

5.2.2.2 JBoss AS 7.0

Deploy: To deploy OpenIoT modules on JBoss AS 7.0, copy the corresponding files "MODULE_NAME.war" to the server's "standalone/deployments" directory.

Undeploy: To undeploy the application, you need to remove the ".deployed" marker file that is generated upon successful deployment of the module.

You can find more detailed directions on the ins and outs of deployment on JBoss AS7 here: https://docs.jboss.org/author/display/AS7/Application+deployment

5.3 User Support Channels

Since all OpenIoT users so far are internal to the project, user support has been given on a case-by-case basis, between the developers working on using the OpenIoT middleware for the validating use cases (see D6.3.1) and with the developers of the middleware itself.

The public mailing list openiot@gouglegroups.com is monitored for queries from OpenIoT users, including project-internal users.

[17] https://github.com/OpenIotOrg/openiot/wiki

[18] https://github.com/OpenIotOrg/openiot/wiki/Downloads

[19] https://github.com/OpenIotOrg/openiot/wiki

[20] https://github.com/OpenIotOrg/openiot/wiki/Downloads

6 PROJECT STATISTICS (AIT)

The statistics are provided by Ohloh (https://www.ohloh.net/) and were taken on 09/12/2013 (**Figure 23**). In a Nutshell, OpenIoT[21] has had 731 commits made by 13 contributors representing 108,118 lines of code. It is mostly written in Java with an medium number of source code comments. It has a codebase with a very short history maintained by a large development team with stable Y-O-Y commits. It has required an estimated 27 manyears[22] of effort (COCOMO model) starting with its first commit in April, 2013 ending with its most recent commit early December 2013.

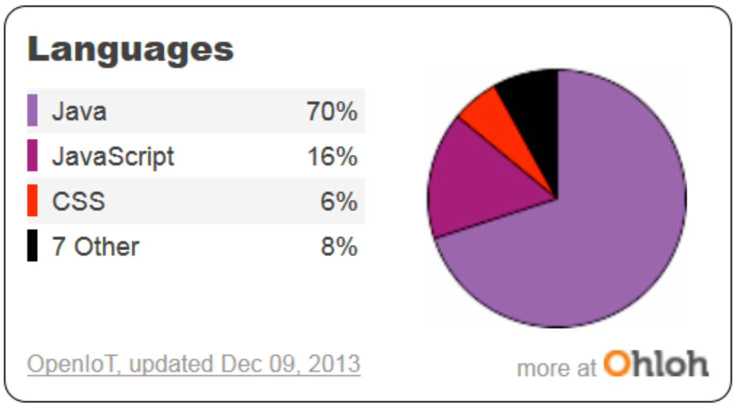
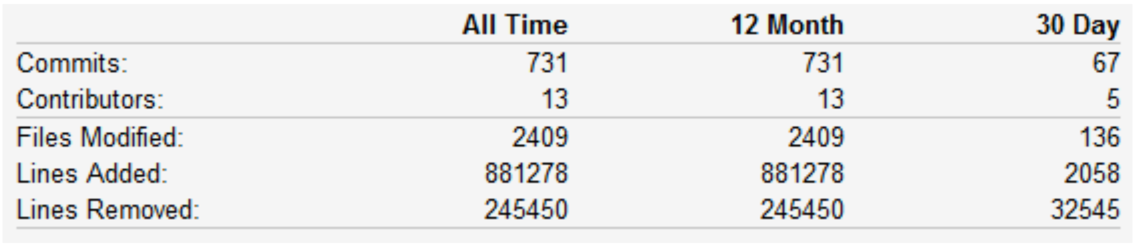


Figure 23 Ohloh Language Stats summary

6.1 Commits

It is illustrated at (**Figure 24** and **Figure 25**)



	All Time	12 Month	30 Day
Commits:	731	731	67
Contributors:	13	13	5
Files Modified:	2409	2409	136
Lines Added:	881278	881278	2058
Lines Removed:	245450	245450	32545

Figure 24: OpenIoT Statistics – Contributors and Commits

[21] https://www.ohloh.net/p/OpenIoT

[22] https://www.ohloh.net/p/OpenIoT/estimated_cost

Commits per Month:

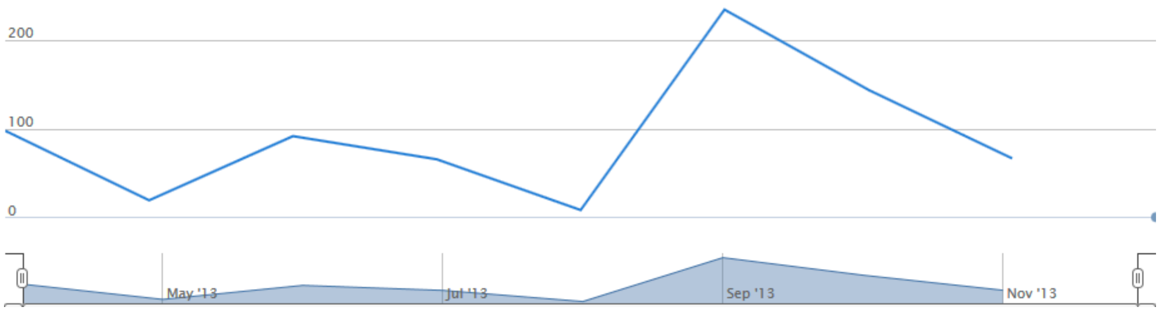


Figure 25 OpenIoT Statistics – Commits per Month

6.2 Contributors

Commits by Top Contributors (

Figure 26):

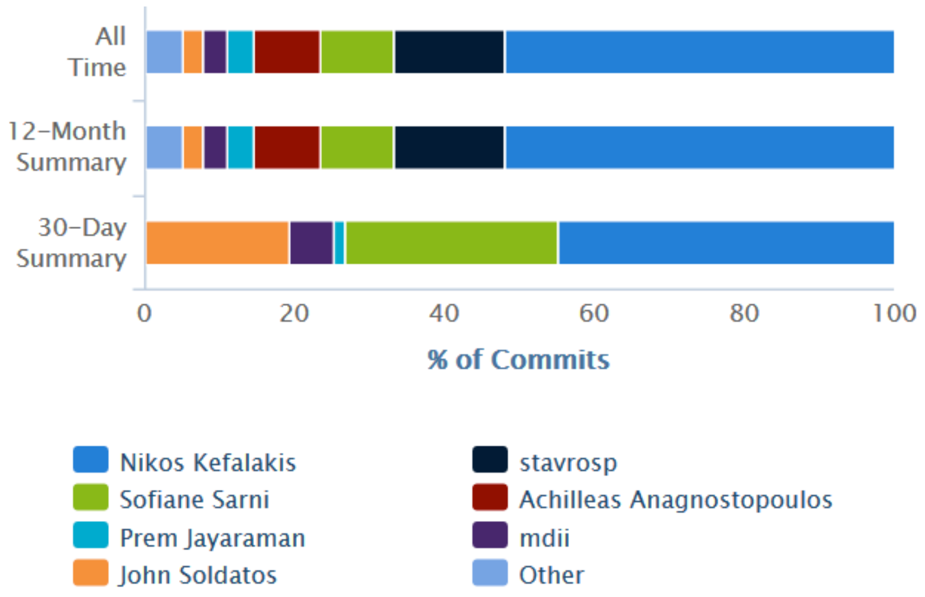
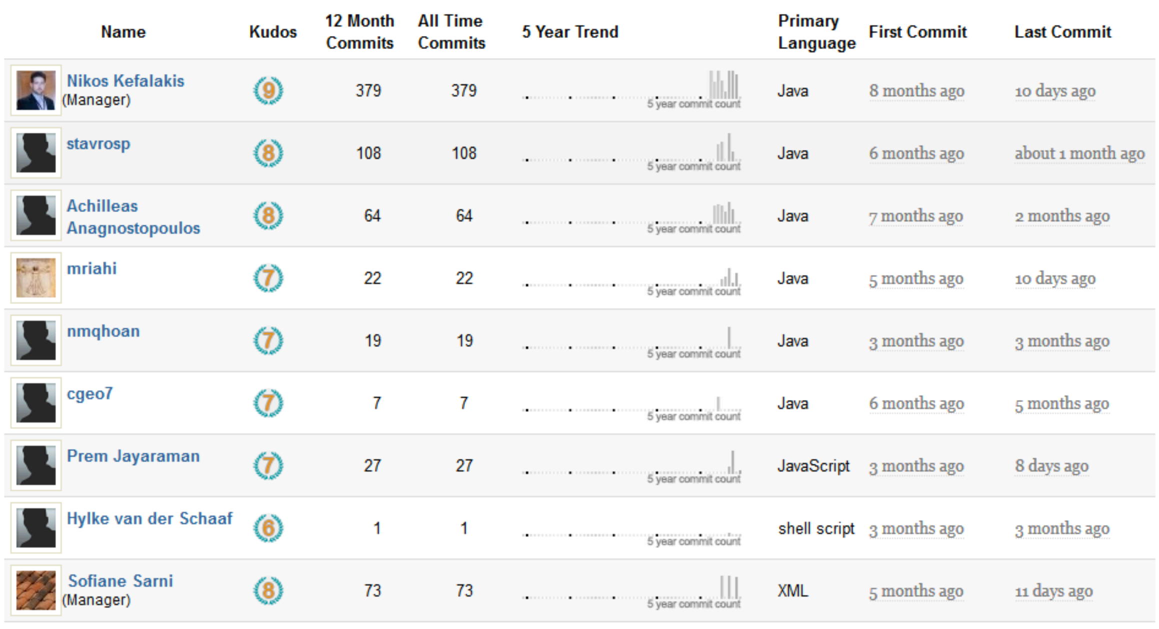


Figure 26 OpenIoT Statistics – Top Contributors

List Of Contributors (**Figure 27**):



Name	Kudos	12 Month Commits	All Time Commits	5 Year Trend	Primary Language	First Commit	Last Commit
Nikos Kefalakis (Manager)	9	379	379	5 year commit count	Java	8 months ago	10 days ago
stavrosp	8	108	108	5 year commit count	Java	6 months ago	about 1 month ago
Achilleas Anagnostopoulos	8	64	64	5 year commit count	Java	7 months ago	2 months ago
mriahi	7	22	22	5 year commit count	Java	5 months ago	10 days ago
nmqhoan	7	19	19	5 year commit count	Java	3 months ago	3 months ago
cgeo7	7	7	7	5 year commit count	Java	6 months ago	5 months ago
Prem Jayaraman	7	27	27	5 year commit count	JavaScript	3 months ago	8 days ago
Hylke van der Schaaf	6	1	1	5 year commit count	shell script	3 months ago	3 months ago
Sofiane Sarni (Manager)	8	73	73	5 year commit count	XML	5 months ago	11 days ago

Figure 27 OpenIoT Statistics – List of Contributors

6.3 Languages

Total Lines : 164,673

Code Lines : 108,118

Percent Code Lines : 66%

Number of Languages : 10

Total Comment Lines :33,335

Percent Comment Lines : 20%

Total Blank Lines : 23,220

Percent Blank Lines : 14%

Code, Comments and Blank Lines (

Figure 28):

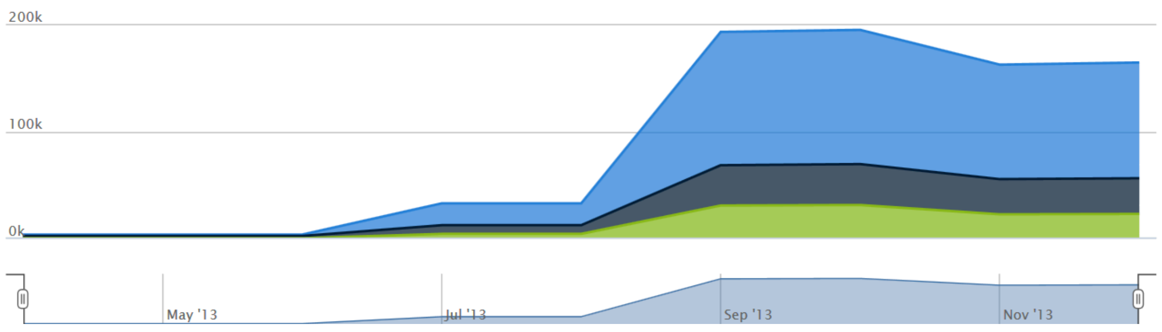
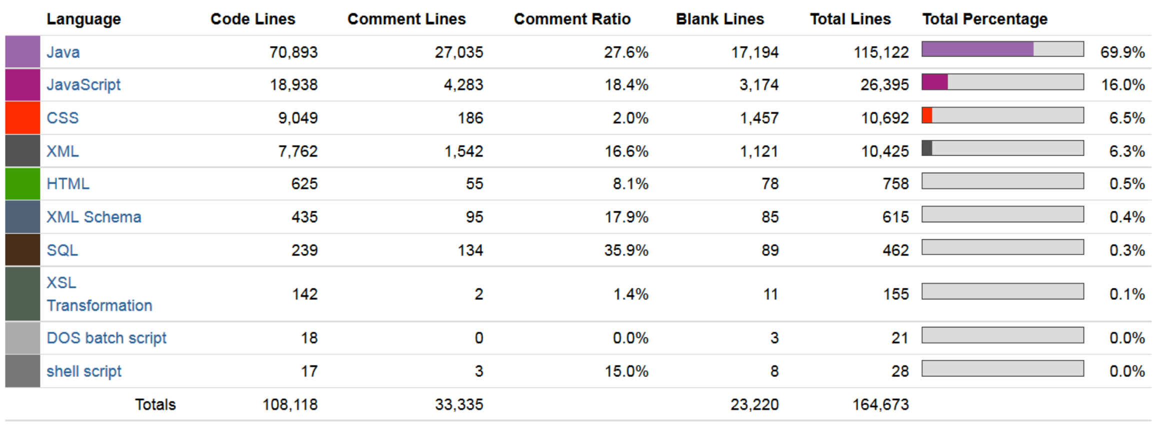


Figure 28 OpenIoT Statistics – Code Comments and Blank Lines

Language Breakdown

Figure 29):



	Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
🟪	Java	70,893	27,035	27.6%	17,194	115,122	69.9%
🟪	JavaScript	18,938	4,283	18.4%	3,174	26,395	16.0%
🟥	CSS	9,049	186	2.0%	1,457	10,692	6.5%
⬛	XML	7,762	1,542	16.6%	1,121	10,425	6.3%
🟩	HTML	625	55	8.1%	78	758	0.5%
⬛	XML Schema	435	95	17.9%	85	615	0.4%
🟫	SQL	239	134	35.9%	89	462	0.3%
⬛	XSL Transformation	142	2	1.4%	11	155	0.1%
⬜	DOS batch script	18	0	0.0%	3	21	0.0%
⬛	shell script	17	3	15.0%	8	28	0.0%
	Totals	108,118	33,335		23,220	164,673	

Figure 29 OpenIoT Statistics – Language Breakdown

7 CONCLUSIONS

This document has reported on further evolution of the OpenIoT open source software portal and Wiki, which began to serve as the basis for the collaborative development of the OpenIoT software. The current document follows the previous deliverable of WP6, namely, D6.1 which accompanied the establishment of the OpenIoT space on Github (available at: http://github.com/openiot/OpenIoT). OpenIoT Github provided the collaborative space and tools used by the OpenIoT open source community. While the OpenIoT software platform is under development as an on going project, the users and contributors are all from the OpenIoT consortium. However, based on the dissemination plans, more users/contributors will be attracted and engaged from outside the OpenIoT consortium.

This document has also reported on the current experience with OpenIoT open source software governance structure, partner contributions and roles. It also describes the OpenIoT open source development infrastructure, including how source code and binaries are organised, reports on documentation structure and maintenance of mailing lists. For the sake of generality, the consortium will be using definitions of developers, users and contributors introduced in section 1.2. However, the roles may be further specialised depending on the context.

This deliverable presents the consortium efforts on building developers and user communities during the OpenIoT software platform development and since the first release of the integrated OpenIoT software platform. It also documents the experience of integrating and deploying the OpenIoT platform at a partner site that highlights important lessons and knowledge to be shared between all consortium partners.

This deliverable presents the statistics related to OpenIoT platform development efforts, software component contributions and OpenIoT software use until M24.

8 REFERENCES

[Aalto 2013] Marja Aalto, Hoang Long & Jukka Pekkala (2013), The Role of Architecture in Open Source Software Projects, *https://www.google.com.au/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CCsQFjAA&url=https%3A%2F%2Fdocs.google.com%2Fdocument%2Fd%2F1wxD-nNS5QDu2VNoiTONQ89J103_eJgk2BaF9jGMCyTQ%2Fedit&ei=Q46tUoTgFeWviQf_g4CwDA&usg=AFQjCNEKvG1Tmcn7vclAgwY89E6rOFfl4A&sig2=ldV5dk0nyvy1lPZHTGfZFQ, Accessed 10.12.2013*

[Aberdour 2007] Aberdour, M. (2007). Achieving quality in open-source software. *Software, IEEE*, *24*(1), 58-64.

[Ducheneaut 2005] Ducheneaut, N. (2005). Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work*, 323-368.

[Fielding 1999] Fielding, R. T. (1999). Shared leadership in the Apache project. *Communications of the ACM*, *42*(4), 42-43

[von Krogh 2003] von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, *32*(7), 1217-1241.

[von Krogh 2003] von Krogh, G., & von Hippel, E. (2003). Special issue on open source software development. *Research Policy*, *32*(7), 1149-1157.

[Ye 2003] Ye, Y., & Kishida, K. (2003, May). Toward an understanding of the motivation of open source software developers. In *Software Engineering, 2003. Proceedings. 25th International Conference on* (pp. 419-429). IEEE.