



# ACTIVAGE PROJECT

ACTivating InnoVative IoT smart living environments for AGEing well

## Developers toolkit and deployment support

<b>Deliverable No.</b>	D4.1	<b>Due Date</b>	31-Dec-2017
<b>Type</b>	Report	<b>Dissemination Level</b>	<i>Public</i>
<b>Version</b>	1.0	<b>Status</b>	Release 1
<b>Description</b>	Provides a set of methods and tools for (non-) technical developers to create applications and services within AIOTES.		
<b>Work Package</b>	WP4 – ACTIVAGE Application support tools and services layer.		



## Authors

Name	Partner	e-mail
Byron Ortiz Sanchez	03 TVES	<a href="mailto:byrort@televes.com">byrort@televes.com</a>
Pilar Sala	04 MYSPHERA	<a href="mailto:psala@mysphera.com">psala@mysphera.com</a>
Alejandro M. Medrano Gil	05 UPM	<a href="mailto:amedrano@lst.tfo.upm.es">amedrano@lst.tfo.upm.es</a>
Saied Tazari	06 Fh-IGD	<a href="mailto:saied.tazari@igd.fraunhofer.de">saied.tazari@igd.fraunhofer.de</a>
Helmi Ben Hmida	06 Fh-IGD	<a href="mailto:helmi.ben.hmida@igd.fraunhofer.de">helmi.ben.hmida@igd.fraunhofer.de</a>
Stéphane Bergeon	07 CEA	<a href="mailto:stephane.bergeon@cea.fr">stephane.bergeon@cea.fr</a>
Mathieu Gallissot	07 CEA	<a href="mailto:mathieu.gallissot@cea.fr">mathieu.gallissot@cea.fr</a>
Nikolaos Kaklanis	08 CERTH	<a href="mailto:nkak@iti.gr">nkak@iti.gr</a>
Konstantinos Votis	08 CERTH	<a href="mailto:kvotis@iti.gr">kvotis@iti.gr</a>
Dimitrios Tzovaras	08 CERTH	<a href="mailto:Dimitrios.Tzovaras@iti.gr">Dimitrios.Tzovaras@iti.gr</a>
Thanos Stavropoulos	08 CERTH	<a href="mailto:athstavr@iti.gr">athstavr@iti.gr</a>
Spiros Nikolopoulos	08 CERTH	<a href="mailto:nikolopo@iti.gr">nikolopo@iti.gr</a>
Ioannis Kompatsiaris	08 CERTH	<a href="mailto:ikom@iti.gr">ikom@iti.gr</a>
Clara Valero	11 UPV	<a href="mailto:clavalpe@dcom.upv.es">clavalpe@dcom.upv.es</a>
Huy Le Van	13 NUIG	<a href="mailto:huy.levan@insight-centre.org">huy.levan@insight-centre.org</a>
Pierre Barralon	15 TEC	<a href="mailto:pierre.barralon@tecnalia.com">pierre.barralon@tecnalia.com</a>
Javier Arcas	15 TEC	<a href="mailto:javier.arcas@tecnalia.com">javier.arcas@tecnalia.com</a>
Rohit Ail	20 IE	<a href="mailto:rohit.ail@samsung.com">rohit.ail@samsung.com</a>
Andrea Carboni	23 CNR	<a href="mailto:andrea.carboni@isti.cnr.it">andrea.carboni@isti.cnr.it</a>
Michele Girolami	23 CNR	<a href="mailto:michele.girolami@isti.cnr.it">michele.girolami@isti.cnr.it</a>
Dario Russo	23 CNR	<a href="mailto:dario.russo@isti.cnr.it">dario.russo@isti.cnr.it</a>
Rami Mäkelä	45 SEN	<a href="mailto:rami.makela@seniorsome.com">rami.makela@seniorsome.com</a>

# History

Date	Version	Change
30-Nov-2017	0.01	Structure of the document and task assignments
December 2017	0.02	First contributions from CEA in development tools sections / sensiNact sub-sections
	0.1	CEA contribution update
January 2018	0.11	Highlighted assignments in table of contents
	0.15	CEA contribution in deployment tools section (4.2.6)
	0.18	Added first contribution by UPV (in section 4.1) + fixed DS1 and DS2 contributors
	0.2	reorganized 3.2 section (about ACTIVAGE dev tools) in order to have platforms as main entries (7 section = 1 by platform), and doc/API/tuto... as sub sections
	0.3	Added contributions by CERTH, NUIG, UPV and TVES
	0.33	Added contribution by UPM
	0.37	Added contribution by TVES in deployment tools section Updates from CERTH and CEA
	0.4	Updates from CEA in DS6 deployment tools
	0.42	Added contribution by CNR in deployment section for DS4 RER
February 2018	0.45	Added contribution by SAMSUNG in deployment tools section for DS8 UK
	0.47	Updated contributions by CERTH and TVES
	0.5	Added contribution by SAMSUNG in deployment section for DS8
	0.6	Polishing of Section 4 and introduction sections by CEA
	0.65	Added contribution by FIN, FhG and updates by UPM
	0.68	Added contribution by MYS
March 2018	0.7	New ToC. Formatting D4.1 version1-0 to new template. Contribution TEC to section 2, section 3, section 4.3
	0.73	Added contribution by CERTH regarding development / deployment tools architecture and IoTivity development / deployment tools

	0.75	Added details by CERTH regarding the development / deployment tools architecture. Added contribution to the interactions between deployment tools and ACTIVAGE marketplace.
	0.8	Integration of contributions by TEC, TVES, CERTH, HOPU, Fh-IGD , IE , UPV, UPM and SEN
April 2018	0.83	Updates by TEC, UPV, SEN, CERTH, SAMSUNG and CEA
	0.85	Updates by UPM (4.2.1) , CEA (1 and 5.2.5), INSIGHT (5.2.4), Fh-IGD (3.2.1)
	0.87	Contribution by HOPU (4.2.8 and 5.2.9) Updates from UPM (5.2.1 and 5.2.1), UPV (4.2.5 and 5.2.6)
	0.89	Contribution by MYS (6) and updates by CEA (1 and 5.2.5.1.4) Fixed lists of tables and figures
	0.9	Internally reviewed by A.Duclos [MADOPA] and updated by CERTH and TECNALIA
May 2018	0.92	Peer reviewed by Elena Tamburini [MAD] + Carlos Palau [UPV] + Antonio Jara [HOP]
	0.94	Updates by INSIGHT
	0.96 – 0.98	Updates by UPV, HOPU, FhG, TEC, CERTH, TVES UPM, MYS, SEN, CERTH
	0.99	Final polishing
04-Jun-2018	1.0	Official Release

## Key data

<b>Keywords</b>	ACTIVAGE, AHA, development toolkit, deployment tool, API, wiki, documentation, source code, swagger, tutorial, IoT platform, FIWARE, IoTivity, OpenIoT, SOFIA2, SENIORSOME, sensiNact, universAAL	
<b>Lead Editor</b>	Stéphane Bergeon	07 CEA
	Mathieu Gallissot	07 CEA
<b>Internal Reviewer(s)</b>	Alexandre Duclos	16 MAD
	Elena Tamburini	14 MEDEA
	Antonio Jara	12 HOPU
	Philippe Dallemagne	19 CSEM

## Abstract

This document is the deliverable D4.1 “Developers toolkit and deployment support” and presents the outcome of the activities T4.1 “ACTIVAGE Development tools” and T4.3 “ACTIVAGE Deployment tools” which are part of Work Package WP4 “ACTIVAGE Application support tools and services layer”. The document makes a status of existing development and deployment tools prior to the ACTIVAGE project and describes the ones that are under progress to fulfill ACTIVAGE dedicated requirements.

The requirements are presented in Section 2, and the Use Cases in Section 3, precisising the uses by the kind of developer profiles listed in section 3.1.

The presented tools in Section 4 allow the development of applications (developed for ACTIVAGE by third parties) on top of each given platform among the seven selected as ACTIVAGE IoT platforms (Fiware, openiot, IoTivity, seniorsome, sensiNact, Sofia2 and universAAL). They are used by application developers and platform contributors. For more details about the seven ACTIVAGE IoT platforms, please refer to the D3.1 Report on IoT European Platforms [1].

The presented tools in Section 5 are the ones used in the nine ACTIVAGE deployment sites to support their hardware device installations and middleware installation on the resulting IT infrastructure. They are used by installation/deployment and IT teams.

These existing development and deployment tools are available today and constitute the first step of ACTIVAGE toolkit: they are heterogeneous tools available for dedicated platforms and for specific deployment sites.

Further works in tasks T4.1 and T4.3 will aim to complete the IoT platform dedicated tools with ACTIVAGE common development and deployment tools based on the common ACTIVAGE APIs under specification.

## Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

# Table of contents

<b>TABLE OF CONTENTS</b> .....	<b>5</b>
<b>LIST OF TABLES</b> .....	<b>7</b>
<b>LIST OF FIGURES</b> .....	<b>9</b>
<b>1 ABOUT THIS DOCUMENT</b> .....	<b>12</b>
1.1 DELIVERABLE CONTEXT.....	12
1.2 THE RATIONALE BEHIND THE STRUCTURE .....	13
<b>2 REQUIREMENTS FOR DEVELOPMENT AND DEPLOYMENT</b> .....	<b>14</b>
2.1 AIOTES DEVELOPMENT TOOL KIT FUNCTIONALITIES.....	14
2.1.1 <i>Support consumption</i> .....	14
2.1.2 <i>Implementation</i> .....	14
2.1.3 <i>Data processing</i> .....	14
2.1.4 <i>IoT infrastructure management</i> .....	14
2.2 AIOTES DEPLOYMENT TOOLKIT FUNCTIONALITIES .....	16
2.2.1 <i>IoT infrastructure management</i> .....	16
2.2.2 <i>Distribution/deployment</i> .....	16
<b>3 USE CASES</b> .....	<b>17</b>
3.1 DEVELOPER PROFILES.....	17
3.2 INDICATIVE USE CASES FOR DEVELOPMENT .....	18
3.2.1 <i>Indicative use cases for non technical developers</i> .....	19
3.3 INDICATIVE USE CASES FOR DEPLOYMENT .....	24
<b>4 DEVELOPMENT TOOLS</b> .....	<b>26</b>
4.1 ARCHITECTURE .....	26
4.1.1 <i>Semantic Interoperability Layer tools</i> .....	29
4.1.2 <i>Data Lake tools</i> .....	32
4.1.3 <i>Data / visual analytics tools</i> .....	34
4.1.4 <i>Integrated Development Environment (IDE)</i> .....	39
4.1.5 <i>Support</i> .....	46
4.1.6 <i>Mapping between development tools requirements and modules</i> .....	46
4.2 AVAILABLE DEVELOPMENT TOOLS SUPPORTED BY THE ACTIVAGE IOT PLATFORMS.....	48
4.2.1 <i>universAAL</i> .....	48
4.2.2 <i>SOFIA2</i> .....	65
4.2.3 <i>OpenIoT</i> .....	70
4.2.4 <i>SensiNact</i> .....	79
4.2.5 <i>FIWARE</i> .....	97
4.2.6 <i>IoTivity</i> .....	105
4.2.7 <i>SeniorSome</i> .....	112

4.2.8	<i>Summary of existing tools</i> .....	113
4.2.9	<i>Mapping between development tools requirements and modules</i> .....	117
4.3	TOOL DEVELOPMENT PLAN .....	119
4.3.1	<i>Planning</i> .....	119
<b>5</b>	<b>DEPLOYMENT TOOLS</b> .....	<b>120</b>
5.1	ARCHITECTURE .....	120
5.1.1	<i>IoT infrastructure management tools</i> .....	123
5.1.2	<i>Deployment management tools</i> .....	127
5.1.3	<i>Mapping between deployment tools requirements and modules</i> .....	130
5.2	AVAILABLE DEPLOYMENT TOOLS SUPPORTED BY THE ACTIVAGE IOT PLATFORMS.....	132
5.2.1	<i>Platform independent Available Deployment tools</i> .....	132
5.2.2	<i>universAAL</i> .....	134
5.2.3	<i>SOFIA2</i> .....	138
5.2.4	<i>OPENIOT</i> .....	140
5.2.5	<i>SensiNact</i> .....	141
5.2.6	<i>FIWARE</i> .....	145
5.2.7	<i>IoTivity</i> .....	146
5.2.8	<i>SeniorSome</i> .....	154
5.2.9	<i>Summary of existing tools</i> .....	155
5.3	DEVELOPMENT WITHIN AIOTES.....	157
5.3.1	<i>Functional description</i> .....	157
5.3.2	<i>Deployment tools description</i> .....	158
5.3.3	<i>Interactions between the Marketplace and deployment tools</i> .....	173
<b>6</b>	<b>CONCLUSION / FUTURE WORK</b> .....	<b>177</b>
	<b>REFERENCES</b> .....	<b>179</b>



# List of tables

TABLE 1: UNIVERSAAL SUPPORT TOOLS.....	48
TABLE 2: UNIVERSAAL API .....	50
TABLE 3: UNIVERSAAL TOOLS MAPPINGS WITH ACTIVAGE DEVELOPMENT TOOLS .....	63
TABLE 4: SOFIA2 EXISTING SAMPLES AND SOURCE CODE .....	67
TABLE 5: MAPPING BETWEEN SOFIA2 AND ACTIVAGE DEVELOPMENT TOOLS. ....	69
TABLE 6: LIST OF PRIMITIVES COMPRISING THE OPENIOT SD&UM IMPLEMENTED API.....	70
TABLE 7: SERVICE DELIVERY & UTILITY MANAGER IMPLEMENTED API DEFINITION .....	71
TABLE 8: LIST OF PRIMITIVES COMPRISING THE OPENIOT LSM-LIGHT API .....	72
TABLE 9: LSM-LIGHT API SPECIFICATION .....	72
TABLE 10: MAPPING BETWEEN OPENIOT AND ACTIVAGE DEVELOPMENT TOOLS. ....	78
TABLE 11: SENSI NACT RESOURCES TYPES AND DESCRIPTION.....	81
TABLE 12: SENSI NACT RESOURCE'S ACCESS METHOD. ....	82
TABLE 13: THE EIGHT IMPLEMENTED SENSI NACT AHA FUNCTIONS. ....	84
TABLE 14: TWO SUPPLEMENTARY AHA FUNCTIONS UNDER DEVELOPMENT IN SENSI NACT. ....	84
TABLE 15: SENSI NACT AHA HISTORICAL STATISTICS AGENT (30 DAYS HORIZON) .....	91
TABLE 16: SUMMARY OF THE SENSI NACT ACTIVAGE DEVELOPMENT TOOLS .....	96
TABLE 17: MAPPING BETWEEN SENSI NACT AND ACTIVAGE DEVELOPMENT TOOLS .....	97
TABLE 18: FIWARE EXISTING TOUR GUIDES .....	101
TABLE 19: MAPPING BETWEEN IOTIVITY AND ACTIVAGE DEVELOPMENT TOOLS.....	104
TABLE 20: MAPPING BETWEEN IOTIVITY AND ACTIVAGE DEVELOPMENT TOOLS.....	111
TABLE 21: MAPPING BETWEEN SENIORSOME AND ACTIVAGE DEVELOPMENT TOOLS.....	113
TABLE 22: HIGH LEVEL PLATFORM OVERVIEW.....	113
TABLE 23: DEVELOPMENT OVER PLATFORM.....	114
TABLE 24: DEVELOPMENT HELPING TOOLS.....	115
TABLE 25: SEMANTIC READY PLATFORM.....	115
TABLE 26: SUPPORT TOOLS.....	116
TABLE 27: IDE TOOLS .....	116
TABLE 28: DATA/VISUAL ANALYTICS TOOLS.....	117
TABLE 29: DATA LAKE TOOLS .....	117
TABLE 30: SEMANTIC INTEROPERABILITY LAYER TOOLS .....	117
TABLE 31: PLANNING.....	119
TABLE 32: THE FUNCTIONALITIES OFFERED BY THE ACTIVAGE DEPLOYMENT TOOLS. ....	121
TABLE 33: MAPPING BETWEEN UNIVERSAAL AND ACTIVAGE DEPLOYMENT TOOLS. ....	137
TABLE 37: DEVICE PARAMETERS ACCORDING TO THEIR COMMUNICATION TYPE.....	148
TABLE 38: MAPPING BETWEEN IOTIVITY AND ACTIVAGE DEPLOYMENT TOOLS.....	154
TABLE 39: MAPPING BETWEEN SENIORSOME AND ACTIVAGE DEPLOYMENT TOOLS.....	155
TABLE 40: DEPLOYMENT TECHNOLOGIES PER PLATFORM.....	155
TABLE 41: DEPLOYMENT TOOLS FOR DEVICE CONFIGURATION.....	156
TABLE 42: IOT INFRASTRUCTURE MANAGEMENT TOOLS .....	156

TABLE 43: DEPLOYMENT MANAGEMENT TOOLS..... 156

TABLE 44: MARKETPLACE FUNCTIONALITY AND DEPENDENCIES ON DEPLOYMENT TOOLS..... 175

# List of figures

FIGURE 1: DEVELOPMENT AND DEPLOYMENT TOOLS FUNCTIONALITIES.....	15
FIGURE 2. PROFILES OF DEVELOPERS THAT ARE FORESSEN TO USE ACTIVAGE DEVELOPMENT AND DEPLOYMENT TOOLS	17
FIGURE 3: USE CASE DIAGRAM OF THE DEVELOPMENT TOOLS FUNCTIONALITIES. ....	18
FIGURE 4: SEQUENCE DIAGRAM OF THE DEVELOPMENT TOOLS FUNCTIONALITIES.....	19
FIGURE 6: DEVISES RELATED CASE DIAGRAM OF THE CLICKDIGITAL IDE VISUAL TOOL. ....	21
FIGURE 7: AIOTES CLICKDIGITAL API USE CASES.....	22
FIGURE 8: RULES RELATED CASE DIAGRAM OF THE CLICKDIGITAL IDE VISUAL TOOL.....	23
FIGURE 9: VISUALIZATION RELATED CASE DIAGRAM OF THE CLICKDIGITAL IDE VISUAL TOOL.....	24
FIGURE 10: USE CASE DIAGRAM OF THE DEPLOYMENT TOOLS FUNCTIONALITIES. ....	25
FIGURE 11: POSITIONING OF THE ACTIVAGE DEVELOPMENT TOOLS WITHIN THE OVERALL ACTIVAGE ARCHITECTURE. .....	26
FIGURE 12: ARCHITECTURE OF THE ACTIVAGE DEVELOPMENT TOOLS COMPONENT AND ITS CONNECTION TO THE OTHER ACTIVAGE COMPONENTS.....	27
FIGURE 13: ACTIVAGE DEVELOPMENT TOOLS.....	28
FIGURE 14: THE SEMANTIC INTEROPERABILITY LAYER (SIL) TOOLS.....	29
FIGURE 15: FUNCTIONALITIES AND COMMUNICATION OF THE ACTIVAGE ONTOLOGY EXPLORER.....	30
FIGURE 16: FUNCTIONALITIES AND COMMUNICATION OF THE QUERY TRANSLATOR.....	30
FIGURE 17: CONNECTION OF THE DEVICE SEMANTICS EDITOR TO THE SIL.....	31
FIGURE 18: CONNECTION OF THE DEVICE SEMANTICS EDITOR TO THE SIL.....	31
FIGURE 19: THE DATA LAKE MODULES.....	32
FIGURE 20: FUNCTIONALITIES AND COMMUNICATION OF THE ACTIVAGE DATA MODEL WORKBENCH.....	33
FIGURE 21: FUNCTIONALITIES AND COMMUNICATION OF THE METADATA STORAGE EXPLORER. ....	33
FIGURE 22: THE DATA / VISUAL ANALYTICS DEVELOPMENT TOOLS.....	34
FIGURE 23: FUNCTIONALITIES AND COMMUNICATION OF THE DATA MANIPULATOR DEVELOPMENT TOOL. ....	35
FIGURE 24: FUNCTIONALITIES AND COMMUNICATION OF THE DATA ANALYSER DEVELOPMENT TOOL. ....	36
FIGURE 25: FUNCTIONALITIES AND COMMUNICATION OF THE FEATURE / RESULT VIEWER DEVELOPMENT TOOL. ....	37
FIGURE 26: FUNCTIONALITIES AND COMMUNICATION OF THE VISUALIZATION EXPLORER DEVELOPMENT TOOL. ....	38
FIGURE 27: THE INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) COMPONENTS. ....	39
FIGURE 28: FUNCTIONALITIES AND COMMUNICATION OF THE CODE GENERATOR DEVELOPMENT TOOL. ....	39
FIGURE 29: FUNCTIONALITIES AND COMMUNICATION OF THE CODE TEMPLATES DEVELOPMENT TOOL. ....	40
FIGURE 30: FUNCTIONALITIES AND COMMUNICATION OF THE SERVICE COMPOSER DEVELOPMENT TOOL. ....	41
FIGURE 31: CLICKDIGITAL IDE FOR PROGRAMMING IOT SOLUTIONS ON TOP OF SEVERAL IOT PLATFORMS ....	43
FIGURE 32: FUNCTIONALITIES AND COMMUNICATION OF THE CLICKDIGITAL IDE. ....	44
FIGURE 33: ADDED VALUES OF CLICKDIGITAL IDE.....	44
FIGURE 34: PLUG CAPABILITIES OF THE CLICKDIGITAL IDE.....	45
FIGURE 35: CREATE CAPABILITIES OF THE CLICKDIGITAL IDE.....	45
FIGURE 36: THE DELIVER CAPABILITIES OF THE CLICKDIGITAL IDE. ....	45
FIGURE 37: THE SUPPORT DEVELOPMENT TOOLS. ....	46
FIGURE 38: MAPPING BETWEEN REQUIREMENTS (ORANGE) TO THE ACTIVAGE DEVELOPMENT TOOLS (GREEN).....	47

FIGURE 39 UNIVERSAAL IOT TOOL SET (BLUE), MODULES (VIOLET), AND WORKFLOW WHICH CAN BE INTERESTING TO ACTIVAGE..... 53

FIGURE 40 INSTALLATION PROCESS OF AAL STUDIO SUITE IN ECLIPSE..... 54

FIGURE 41 SOURCE DOWNLOAD INTERFACE ..... 55

FIGURE 42 BASIC INFORMATION PROVIDED FOR THE PROJECT WIZARD ..... 56

FIGURE 43 CUSTOMIZATION OF THE PROJECT DEPENDENCIES AND COMPONENTS IN THE PROJECT WIZARD. .... 56

FIGURE 44 BASIC INFORMTION FOR THE ONTOLOGY PROJECT WIZARD..... 57

FIGURE 45 VIEW OF RECENTLY CREATED PROJECT. .... 58

FIGURE 46 TOOLTIPS TO CREATE ONTOLOGIES GRAPHICALLY. .... 58

FIGURE 47 A CODE-GENERATION JAVA GENERATION EXAMPLE ..... 62

FIGURE 48: REQUEST DEFINITION USER INTERFACE (UI) ..... 74

FIGURE 49: REQUEST PRESENTATION UI ..... 75

FIGURE 50: SENSINACT GENERIC DATA MODEL. THIS GENERIC DATA MODEL ALLOWS A SIMILAR ACCESS TO THE SENSORS AND ACTUATORS USING HETEROGENEROUS PROTOCOLS..... 81

FIGURE 51: THE SENSINACT AHA FUNCTIONS (IMPLEMENTED AND IN PROGRESS) ..... 85

FIGURE 52: SENSINACT AHA SERVICE API SPECIALIZATION..... 85

FIGURE 53: LIST OF AVAILABLE SENSINACT AHA SERVICES..... 86

FIGURE 54: SWAGGER SCREENSHOT FOR THE SENSINACT AHA SERVICE REST API ..... 88

FIGURE 55: IOTIVITY JAVA API DOCUMENTATION..... 105

FIGURE 56: A VIEW OF THE IOTIVITY ARCHITECTURE WITH CLOUD FUNCTIONALITY ..... 108

FIGURE 57: ONEIOTA DATA MODELS DEVELOPMENT PROCESS. .... 109

FIGURE 58: SERVICE PROVIDER PERSPECTIVE OF IOTIVITY SIMULATOR ..... 110

FIGURE 59: CLIENT CONTROLLER PERSPECTIVE OF IOTIVITY SIMULATOR..... 110

FIGURE 60: MAPPING BETWEEN REQUIREMENTS (ORANGE) TO THE ACTIVAGE DEVELOPMENT TOOLS (GREEN)..... 118

FIGURE 61: POSITIONING OF THE ACTIVAGE DEPLOYMENT TOOLS COMPONENT WITHIN THE OVERALL ACTIVAGE ARCHITECTURE. .... 120

FIGURE 62: ARCHITECTURE OF THE ACTIVAGE DEPLOYMENT TOOLS COMPONENT AND ITS CONNECTION TO THE OTHER ACTIVAGE COMPONENTS..... 123

FIGURE 63: THE ACTIVAGE DEPLOYMENT TOOLS. .... 123

FIGURE 64: THE IOT INFRASTRUCTURE MANAGEMENT DEPLOYMENT TOOLS..... 124

FIGURE 65: FUNCTIONALITIES AND COMMUNICATION OF THE DEVICE MANAGER DEPLOYMENT TOOL..... 124

FIGURE 66: FUNCTIONALITIES AND COMMUNICATION OF THE SERVICE MANAGER DEPLOYMENT TOOL. .... 125

FIGURE 67: FUNCTIONALITIES AND COMMUNICATION OF THE SEMANTIC AUTO-DISCOVERY PLATFORM DEPLOYMENT TOOL..... 126

FIGURE 68: FUNCTIONALITIES AND COMMUNICATION OF THE BENCHMARKING DEPLOYMENT TOOL..... 127

FIGURE 69: THE DEPLOYMENT MANAGEMENT TOOLS. .... 127

FIGURE 70: FUNCTIONALITIES AND COMMUNICATION OF THE DEPLOYMENT MANAGER TOOL. .... 128

FIGURE 71: FUNCTIONALITIES AND COMMUNICATION OF THE COMPONENT CONFIGURATION DEPLOYMENT TOOL.... 129

FIGURE 72: FUNCTIONALITIES AND COMMUNICATION OF THE MAINTENANCE PANEL DEPLOYMENT TOOL. .... 129

FIGURE 73: FUNCTIONALITIES AND COMMUNICATION OF THE UPDATE MANGER DEPLOYMENT TOOL. .... 130

FIGURE 74: MAPPING BETWEEN REQUIREMENTS (ORANGE) TO THE ACTIVAGE DEPLOYMENT TOOLS (GREEN)..... 131

FIGURE 75 RUN CONFIGURATION OF PAX RUNNER FOR ECLIPSE ..... 135

FIGURE 76: A SOFIA2 DEPLOYMENT TOOL FOR SELECTING ACTIVE APPLICATIONS PER USER ..... 139

FIGURE 77: INSTALLER TOOL MOCK ‘UP: DISCOVERING THE GROUND PLANE OF THE INSTALLATION ..... 142

FIGURE 78: INSTALLER TOOL MOCK ‘UP: DISCOVERING THE EXISTING ASSET, IN THIS CASE THE WATER COUNTER AND VALVE..... 142

FIGURE 79: SCREENSHOT OF THE SENSINACT STUDIO WEB DEVICE AND SERVICE NAVIGATOR ..... 143

FIGURE 80: SENSINACT STUDIO TOOL OVERVIEW ..... 144

FIGURE 81: SENSINACT STUDIO VIEW AND EDITOR COMPONENTS ..... 144

FIGURE 82: SAMPLES OF DSL SCRIPT IN SENSINACT STUDIO ..... 145

FIGURE 83: AUTO-COMPLETION IN SENSINACT STUDIO DSL EDITOR..... 145

FIGURE 84: IOTIVITY EASY SETUP ..... 147

FIGURE 85: DEVICE MANAGEMENT MECHANISM FOR IOTIVITY PLATFORM ..... 149

FIGURE 86: SEQUENCE DIAGRAM FOR DEVICE REGISTRATION FOR IOTIVITY PLATFORM..... 150

FIGURE 87: SEQUENCE DIAGRAM FOR DEVICE UPDATE FOR IOTIVITY PLATFORM ..... 151

FIGURE 88: SEQUENCE DIAGRAM FOR DEVICE REMOVAL FOR IOTIVITY PLATFORM..... 152

FIGURE 89: SEQUENCE DIAGRAM FOR GETTING REGISTERED DEVICES FOR IOTIVITY PLATFORM ..... 153

FIGURE 90: DEPLOYMENT TOOLS IN AIOTES HIGH LEVEL ARCHITECTURE ..... 158

FIGURE 91: DEVICE MANAGER SEQUENCE DIAGRAM..... 160

FIGURE 92: DEVICE MANAGER GUI MOCKUP ..... 160

FIGURE 93: SEVICE MANAGER SEQUENCE DIAGRAM ..... 162

FIGURE 94: SEVICE MANAGER GUI MOCKUP ..... 162

FIGURE 95: SEMANTIC DISCOVERY TOOL SEQUENCE DIAGRAM ..... 164

FIGURE 96: SEMANTIC DISCOVERY TOOL GUI MOCKUP ..... 164

FIGURE 97: BENCHMARKING SEQUENCE DIAGRAM ..... 166

FIGURE 98: BENCHMARKING GUI MOCKUP ..... 166

FIGURE 99: INVENTORY VIEWER SEQUENCE DIAGRAM..... 168

FIGURE 100: INVENTORY VIEWER GUI MOCKUP ..... 168

FIGURE 101: COMPONENT CONFIGURATION SEQUENCE DIAGRAM..... 169

FIGURE 102: COMPONENT CONFIGURATION GUI MOCKUP ..... 170

FIGURE 103: MAINTENANCE PANEL SEQUENCE DIAGRAM ..... 171

FIGURE 104: MAINTENANCE PANEL GUI MOCKUP..... 171

FIGURE 105: UPDATE MANAGER SEQUENCE DIAGRAM ..... 172

FIGURE 106: UPDATE MANAGER GUI MOCKUP..... 173

FIGURE 107: MARKETPLACE HOME PAGE MOCKUP ..... 174

FIGURE 108: MARKETPLACE HOME PAGE IMPLEMENTATION..... 174

FIGURE 109: MARKETPLACE APPLICATION VIEW MOCKUP. .... 175

# 1 About This Document

This deliverable contributes directly to the creation of the ACTIVAGE IoT Ecosystem Suite (AIOTES) by dealing with additional software tools on top of the AIOTES core that help non-Activage contributors to develop and deploy AIOTES-related components. The final goal is that third parties will be able to access the services and features provided by ACTIVAGE by means of a Web-based API wrapped into a Software Development Kit (SDK) integrated in a developer’s toolkit.

D4.1 strives for achieving the following more concrete objectives:

- To provide the ACTIVAGE developing infrastructure to assist the development and usage of available IoT services/tools (existing and new): Web-based API wrapped into a Software Development Kit (SDK) for third parties to enable the extension and development of new applications.
- To provide, a cloud-based semantic auto-discovery platform component to support the overall deployment process.
- To provide the ACTIVAGE technology integrators with tools to help the hardware installation and the software deployment
- To improve the re-usability, interoperability and sharing of ACTIVAGE cross pilot IOT services/applications
- To provide all the necessary tools for the creation of services/applications from users with minimum technical training.

As a technical document, the expected readers of this report are mainly technical teams involved in application development, i.e. software developers, on one side, and in site deployments, i.e. deployment and installation teams, on the other side.

Work on this deliverable started with the identification of the different use cases for application development and solution deployment in order to approach the two classes of tools within a relevant context.

In a next step, we then tried to provide answers for a set of “how to” questions in the context of each use case. For example, in the context of the application development use cases, we examined the question “How to use the ACTIVAGE AHA<sup>1</sup> APIs?” from the perspective of application developers; from the perspective of service providers and platform contributors, as another example, we examined the question “how to contribute to ACTIVAGE?”

Similarly, with regard to deployment tools we tried to deal with questions, such as “How to deploy an ACTIVAGE solution?” and “how to support the installation of the hardware and software elements that constitute the whole of an ACTIVAGE solution?”

## 1.1 Deliverable context

Project item	Relationship
<b>Objectives</b>	D4.1 contributes directly to the following ACTIVAGE objective: <u>O1. To deliver the ACTIVAGE IoT Ecosystem Suite (AIOTES)</u>
<b>Exploitable</b>	By contributing to O1, D4.1 becomes one of the necessary outputs for

<sup>1</sup> Active and Healthy Ageing

<b>results</b>	creating the ACTIVAGE IoT Ecosystem Suite as one of the major exploitable results of the project.
<b>Work plan</b>	<p>This deliverable reports about the progresses of the following tasks belonging to WP4: T4.1 “ACTIVAGE Development tools” and T4.3 “ACTIVAGE Deployment tools” which are part of Work Package WP4 “ACTIVAGE Application support tools and services layer”.</p> <ul style="list-style-type: none"> <li>– The T4.1 task works on the implementation of the ACTIVAGE developing infrastructure to assist the development and usage of available IoT services/tools (existing and new). It supports a basic service-oriented functionality (implemented in T4.3) such as registration, resolving, discovery of services and their composition to create service federations in order to minimize development efforts of developers/integrators that they would be part of the ACTIVAGE IOT ecosystem. This work aims to provide a Web-based API wrapped into a Software Development Kit (SDK) for third parties to enable the extension and development of new applications.</li> <li>– The T4.3 task refers to the actual deployment and continuous operation of the ACTIVAGE IOT pilot services implemented in WP9, along with their testing, validation, evaluation and upgrading during pilot use cases realisation. Task 4.3 works on the implementation of a cloud-based semantic auto-discovery platform component in order to support the overall deployment process whenever is needed (e.g. new services offered, updated).</li> </ul>
<b>Milestones</b>	D4.1 is a complementary deliverable for assessing the achievement of MS2 - DEMONSTRATE.
<b>Deliverables</b>	D4.1 uses D3.1 and D3.2 as input. It is an important input for the next WP5 deliverables. A next version of D4.1 will be published in Month 30 (D4.4 ≡ D4.1.2).
<b>Risks</b>	<p>D4.1 contributes to gaining control of the following risk:</p> <p>Rk5: Failure to attract proposals for open call (the existence of tools makes the open calls more attractive)</p> <p>Rk15: Risk of time consuming due to multiple technology (tools are appropriate means for mitigating such risk)</p>

## 1.2 The rationale behind the structure

After a short overview of the requirements in Section 2, we present the Development and Deployment (D&D) use cases in Section 3. Then, in order to assist the developer, the report describes (a) the available development tools supported by the ACTIVAGE IoT platforms (Section 4.2) and then (b) the ongoing AIOTES development tools (Section 0).

In the last part of the report (Section 5), available deployment tools supported by the ACTIVAGE IoT platforms are described (Section 5.2) as well as development within AIOTES (Section 5.3).

## 2 Requirements for development and deployment

After the analysis of AIOTES requirements extracted from different sources (i.e. background IoT platforms, deliverable D2.1 on requirements, technical experts from deployment sites...), 23 requirements related to development tools have been identified and organised into four categories: support consumption, implementation, data processing and IoT infrastructure management.

Similarly, a total of 14 requirements related to deployment tools has been organised in two categories, such as IoT infrastructure management, both devices and services, and distribution and deployment.

Therefore, and in order to provide effective D&D tools (e.g. shortest implementation and testing time) to external AIOTES developers a list of functional requirements has been elaborated (Figure 1).

### 2.1 AIOTES Development tool kit functionalities

#### 2.1.1 Support consumption

It provides resources and documentation, in order to facilitate the developer in developing applications within the AIOTES (wiki, tutorials, code samples, training, live demos, discussion forum). For more details, refer to Section 4.1.5

#### 2.1.2 Implementation

It refers to a software application that provides comprehensive facilities to computer programmers for software development. Close to the functionalities of an Integrated Development Environment (IDE) it will support source code edition, compilation or interpreter, build automation tools, and a debugger. For more details, refer to Section 4.1.4.

#### 2.1.3 Data processing

Data processing support tools allow the operator to (a) perform AIOTES data analytics methods or and visualise the results on existing (open) databases but also on customised data or (b) develop a new (or improved version of) an analytics method. For more details, refer to Sections 4.1.2 and 4.1.3.

#### 2.1.4 IoT infrastructure management

Such functionalities refer to the ability of registering new devices or users, subscribing to events/topics/webservices, discovering new devices or services, and searching and filtering AIOTES components.



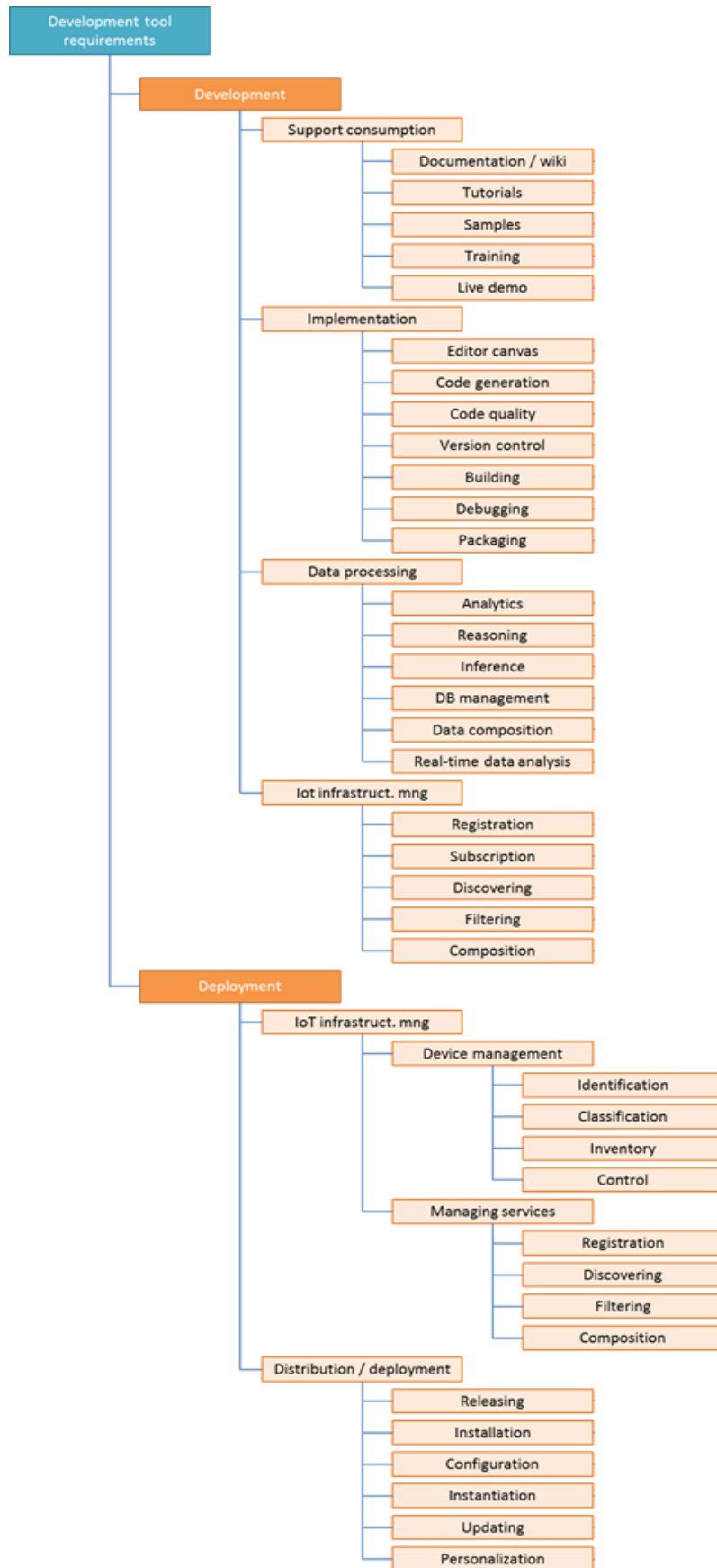


Figure 1: Development and Deployment tools functionalities

## 2.2 AIOTES Deployment toolkit functionalities

Like the development functionalities, the deployment tools should address the following aspects.

### 2.2.1 IoT infrastructure management

On one hand there must be tools to manage devices (identification, classification, inventory, control, maintenance, calibration), and on the other hand dedicated methods to manage services (registration, discovery, filtering composition). For more details, refer to Section 5.1.1.

### 2.2.2 Distribution/deployment

It must contain the following functionalities: releasing, installation, configuration, installation, updating. For more details, refer to Section 5.1.2.

## 3 Use cases

### 3.1 Developer profiles

The software development and deployment tools are offered to the community with the aim of facilitating the inclusion of new (AHA) services into the AIOTES ecosystem. The audience and contributors will have different technical backgrounds and experiences with the technologies embedded into AIOTES. We have identified the following developer profiles (see Figure 2):



Figure 2. Profiles of developers that are foreseen to use Activage development and deployment tools

- “Non-technical” developer

A “non-technical” developer corresponds to a person with no software development skills but with experience and expertise on (a) applications and/or services (e.g. a set of primary preventive interventions) and/or ux-design and/or living labs professionals curious to explore how such applications or services could be embedded (empowered) within AIOTES.

- Junior software developer

A junior developer is a person with basic development skills and a limited experience. This profile might not be familiar with all technologies used in AIOTES (e.g. web, mobile, desktop, backend, front, webservices) but has the background to learn them. He/she would be asked to implement AIOTES-based Proof-of-Concepts (POCs).

- Senior software developer

A senior developer understands that everything in his field involves trade-off, and will look for what that is for design patterns, libraries, frameworks, and processes. He/she understands that his/her job is to provide solutions to problems, not writing code. Related to ACTIVAGE AIOTES platform, a senior developer will first evaluate the strengths and weaknesses of the ecosystem prior to intensive developments.

– ACTIVAGE DS service developer

This person is an Activage Deployment Site developer and, in the course of the ACTIVAGE project, had to develop software components communicating with the AIOTES platform. This person has therefore a knowledge and some experience with AIOTES modules.

– ACTIVAGE AIOTES developer

AIOTES has emerged from seven European IoT platforms (FIWARE, IoTivity, OpenIoT, SENIORSOME, SensiNact, universAAL). An Activage IoT developer is a person who contributed to the development of one of these seven IoT platforms and contributed to AIOTES development. It is, therefore, someone who knows in detail the architecture and functioning of AIOTES.

AIOTES development and deployments tools are mainly targeting external Junior and Senior developers. They will also be used by ACTIVAGE developers to fasten upcoming Activage developments. Whenever possible, specific deployment tools will be offered to “non-technical” persons (e.g. code-generation plugin of Protégé).

### 3.2 Indicative use cases for development

Figure 3 illustrates the different steps that a developer would/could have to go through when implementing a new application using AIOTES. Therefore, development tools should cover all the listed functionalities.

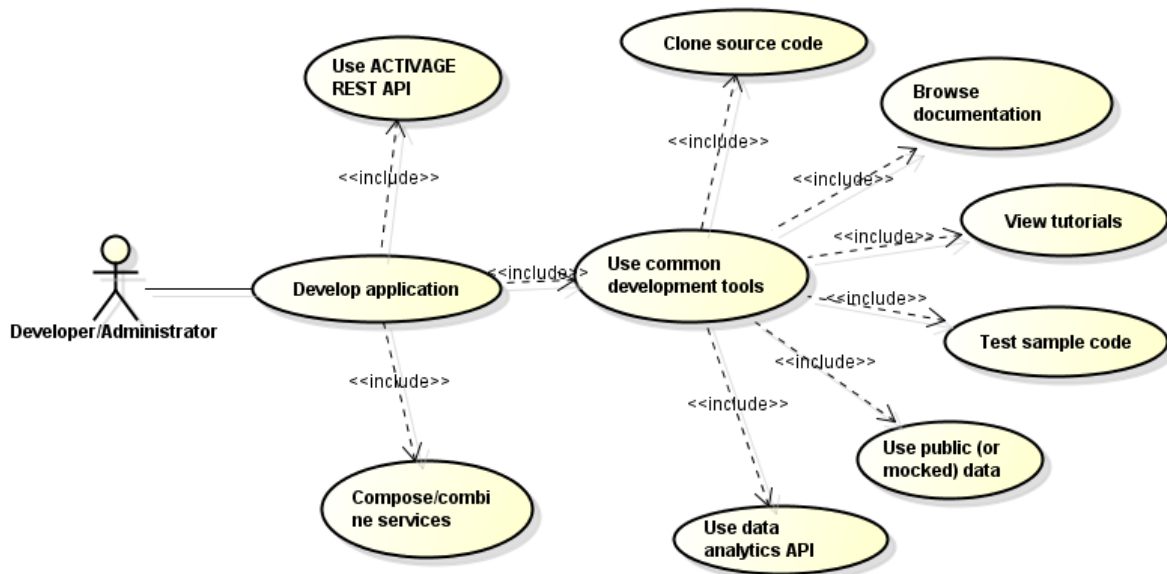


Figure 3: Use case diagram of the development tools functionalities.

The interconnection between the ACTIVAGE development tools and the other ACTIVAGE components is more explicitly depicted in the sequence diagram of figure Figure 4.

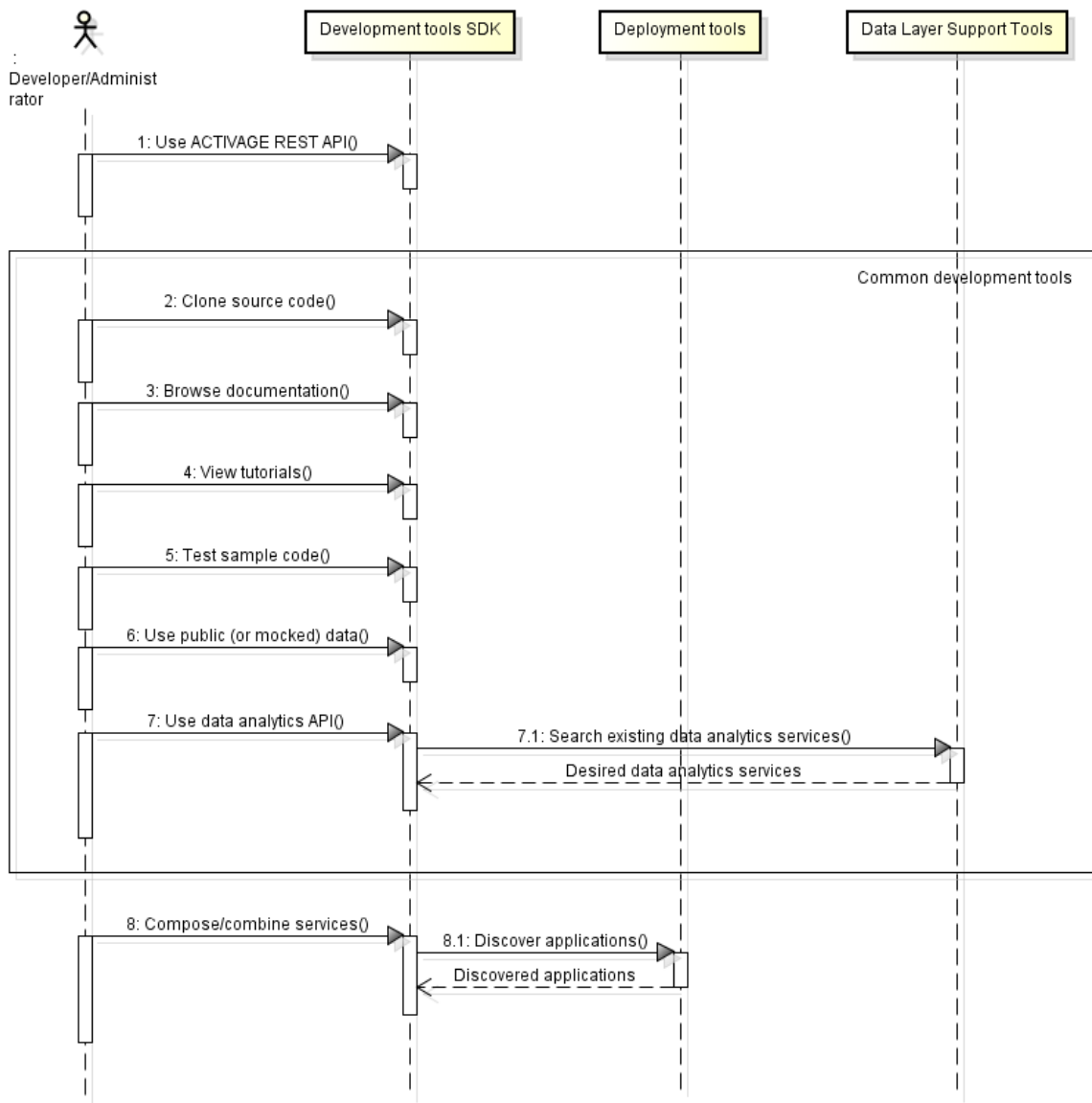


Figure 4: Sequence diagram of the development tools functionalities.

### 3.2.1 Indicative use cases for non technical developers

In the analysis of the development tool kit requirements, a particular attention has been on non technical developers. As a consequence, we recently proposed to incorporate, ClickDigital as a development tool.

In order to enable, empower and ease the usage of the AIOTES interoperability layer in the Activage project context by the different technical developers of the pilots’ sites, but also beyond the project duration as a part of the further exploitation and sustainability strategy, a very important already identified barriers, which is the programming complexity & heterogeneity, should be taken in consideration. Therefore, the main vision to enable the widespread and to increase the acceptance of AIOTES is the creation, part of the AIOTES APIs, of an “Enabler API” that ease the integration of platforms, the quick development of Smart interoperable IoT Services, but also the usage of further APIs and services from other platforms. In this context, ClickDigital will bring a valuable added value to the project. The

Fraunhofer IGD has initially developed “ClickDigital”, a Visual and Pluggable User Friendly Integration development environment for IoT platforms to ease the creation of Smart and digital services for smart living, smart city, mobility and eHealth like visually programming a fall detection and notification services, visually adding new hardware sensor to the platform... It allows you to quickly prepare smart digital solutions and offer the resulting application Dashboard in a “use only” mode to your clients. In fact, the aims behind the ClickDigital as a pluggable visual IoT IDE for different IoT platforms is

- To decrease the learning curve/complexity of creating Apps for heterogenous IoT platforms
- To offer a new App creation experience for the developers/consumers of IoT solutions
- To optimize the path/time to the market
- To enable the IT departments to develop, optimize the cost and enhance the usage of IoT solutions based on multitude used IoT platform through the AIOTES framework.



Figure 5: ClickDigital Logo and elevator pitch

Clickdigital has recently achieved a very valuable conclusion in relation with quantifying its added value from a Money/Time value perspective same from user experience ones. For that reason, it was decided not to earlier integrate it in D5.1: Integration plan and operational framework till validating its added value. As tangible proves about its impact on Activage and the future sustainability of the related ecosystem as been bought, it has been recently decided to include the ClickDigital Framework in Activage and to created a specific ClickDigital API for AIOTES.

In this context, the following use cases will be taken in consideration for further development and adaptation to the single ACTIVAGE IoT platforms but also for the AIOTES.

### 3.2.1.1 AIOTES ClickDigital common use cases

While using ClickDigital, and as highlighted in the bellow figure, the developer as the main addresses user should be able to address the following common usecases:

- Execute the IDE
- Connect it to AIOTES or a compatible IoT Platform through it
- Use the Drag and drop paradigm for UI – card widgets to manage the diffrents available Widgets (Widget for devices mgmt., vidualization, monitoring, rules creation...)
- Add Widget
- Delete Widget
- Adjust Themes settings
- Adjust text size
- Resize with automatical rearranging of content

- Developers and endusers Log-in to AIOTES framework
- Change user
- Add Screens\Desktops\Tabs
- Export App for a use-only mode
- Create and share projects
- Create, configure, share and configure IoT Dashboards
- Configure Widgets

Once exported the dashboard to a “use only mode”, the enduser user will be able to login to the dashboard and exploit the already prepared functionalities (cf. Figure 7 on next page).

### 3.2.1.2 AIOTES ClickDigital devices related use cases

From a devices perspective, the developer, and while using ClickDigital IDE plugged to AIOTES and the targeted IoT Platform, should be able, through the usage of widgets and by creation of dashboard widgets, to perform the following tasks:

- View existing devices
- Find a device
- Add a device
- Monitor\Visualize a device status (Functional visualization)
- Control device
- Control a group of devices

The tasks are provided mainly through the Interoperability framework from AIOTES. From a usage perspective, the user will be only allowed to control the explored and available devices configured by the system developer.

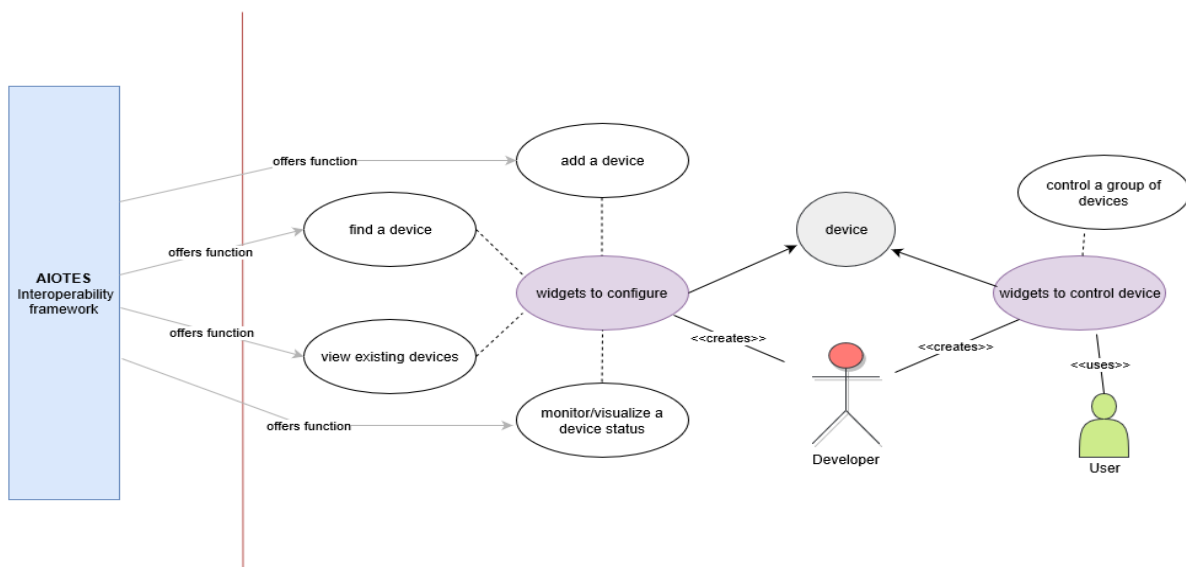


Figure 6: Devises related case diagram of the ClickDigital IDE visual tool.

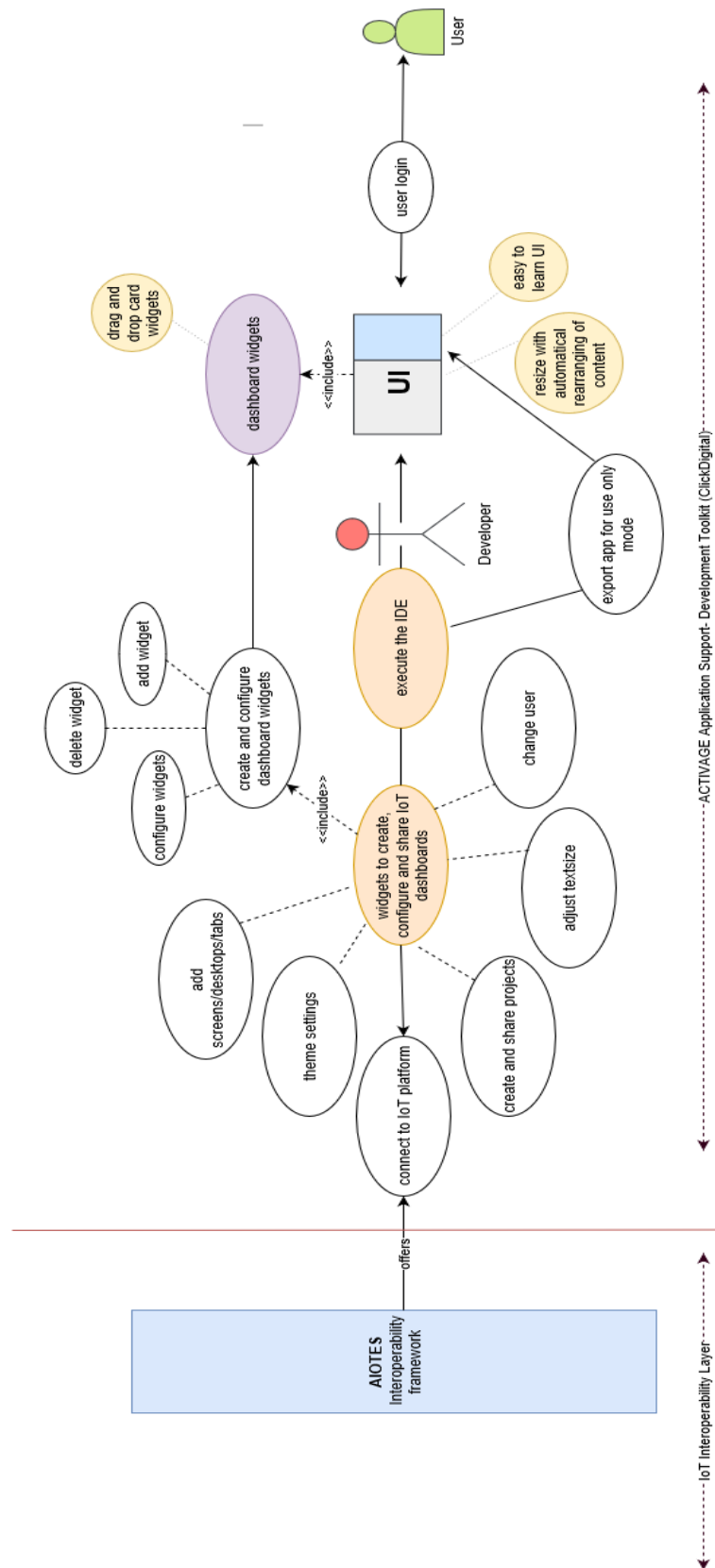


Figure 7: AIOVES ClickDigital API use cases



### 3.2.1.3 AIOTES ClickDigital rules related use cases

As while managing devices, ClickDigital Widgets enable the developers to visually and smoothly create different rules logic based on AIOTES connected to one or more IoT platforms, mainly

- Create a rule
- Edit an existing rule
- Find a rule
- Manage Rules (Activate, deactivate... )
- Rules notification
- Visualize rules

The user of the created Dashboard will be also allowed to activate or deactivate rules,

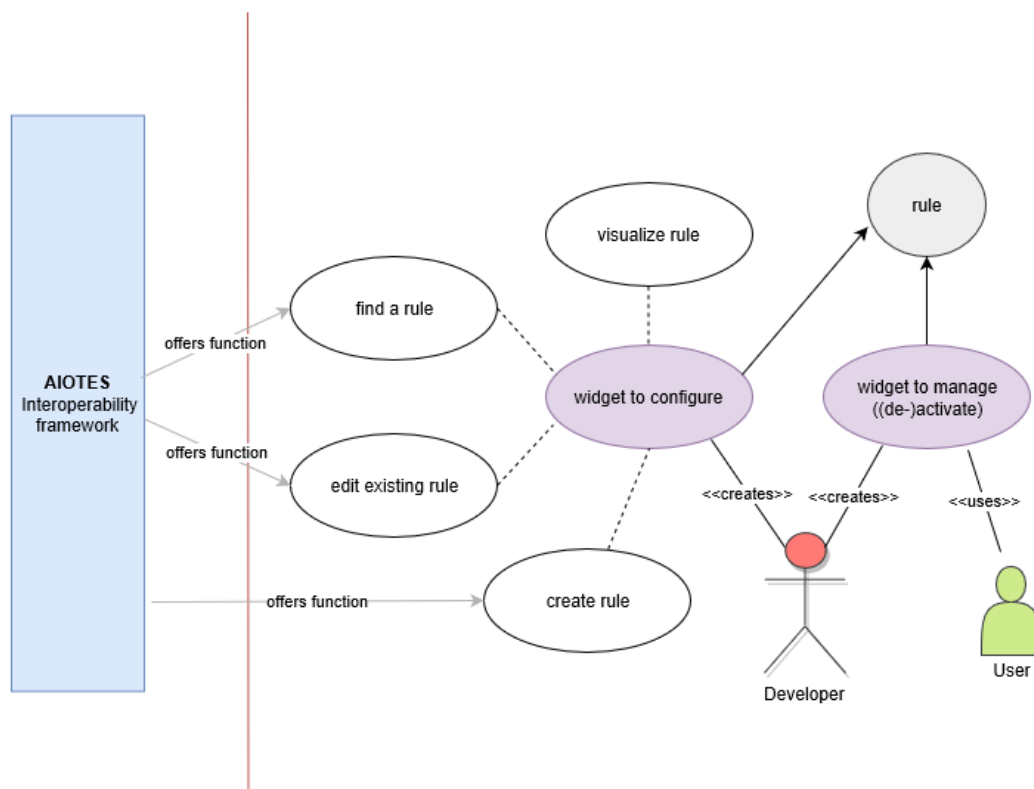


Figure 8: Rules related case diagram of the ClickDigital IDE visual tool.

### 3.2.1.4 AIOTES ClickDigital visualization related use cases

Finally, the bellow figure illustrates the use cases a developer and the end users can be perform from a visualization perspective based on the AIOTES framework, mainly:

- Data workflow visualization-Visualize data from a device (historical visualization)
- Data workflow real time monitoring
- Add a device
- Monitor\Visualize a device status (Functional visualization)

- Visualize health status of a device
- Monitor a group of devices

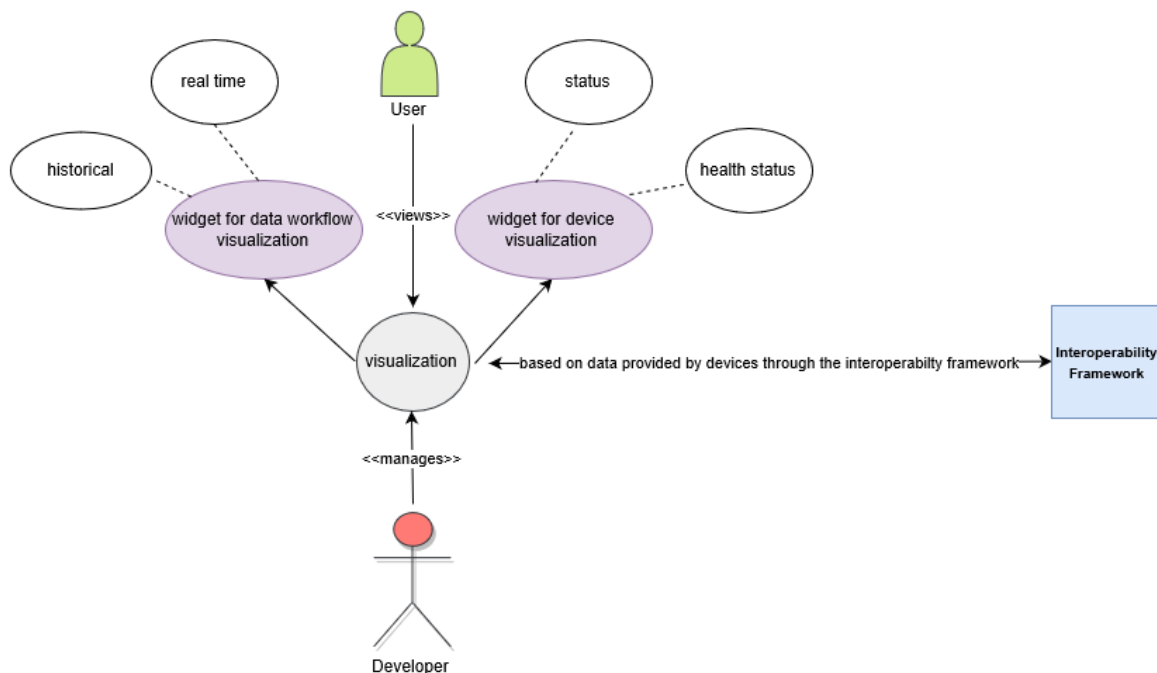


Figure 9: Visualization related case diagram of the ClickDigital IDE visual tool.

### 3.3 Indicative use cases for deployment

Figure 10 illustrates the different steps that a developer would/could have to go through when implementing a new application using AIOTES. Therefore, deployment tools should cover all the listed functionalities.

For simple systems, installation involves establishing some form of command, shortcut, script or service for executing the software (manually or automatically). For complex systems it may involve configuration of the system – possibly by asking the end-user questions about its intended use, or directly asking them how they would like it to be configured – and/or making all the required subsystems ready to use.

The deployer can use the semantic discovery tools to search for existing components registered by the community of IoT developers, in order to deploy them to the actual deployment site. Configuration and maintenance functionalities are also provided, for the proper installation and operation of the deployed application.

Activation is the activity of starting up the executable component of software for the first time.

Deactivation is the inverse of activation, and refers to shutting down any already-executing components of a system. Deactivation is often required to perform other deployment activities, e.g., a software system may need to be deactivated before an update can be performed.

The update process replaces an earlier version of all or part of a software system with a newer release. It commonly consists of deactivation followed by installation.

Version tracking systems help the user find and install updates to software systems. For more details, refer to Section 5.

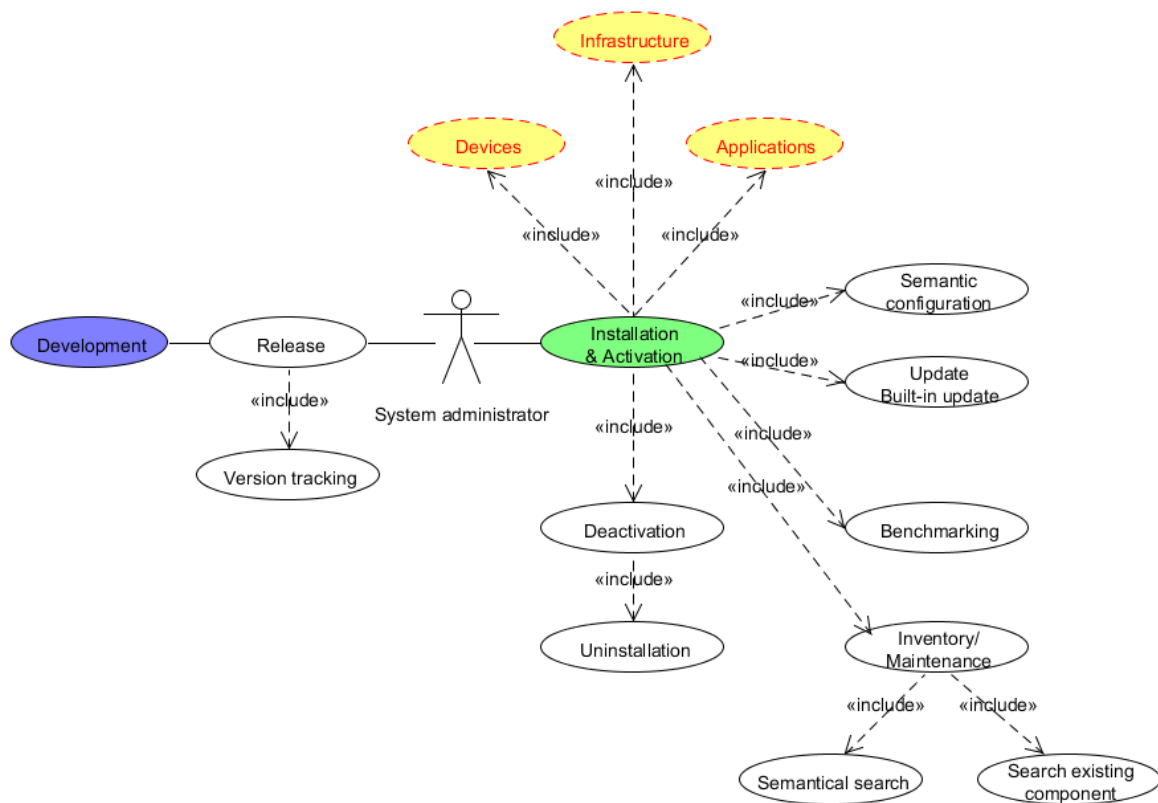


Figure 10: Use case diagram of the deployment tools functionalities.

# 4 Development tools

## 4.1 Architecture

The ACTIVAGE development tools offer means to facilitate the design, the implementation and test of new AHA IoT applications. They are closely linked with the application deployment tools described in Section 5 and allow the composition of existing applications and tools, in order to easily generate new applications. The ACTIVAGE development tools are part of the ACTIVAGE application tools, which operate at the highest levels of the overall project architecture. The positioning of the development tools within the ACTIVAGE architecture is depicted in Figure 11.

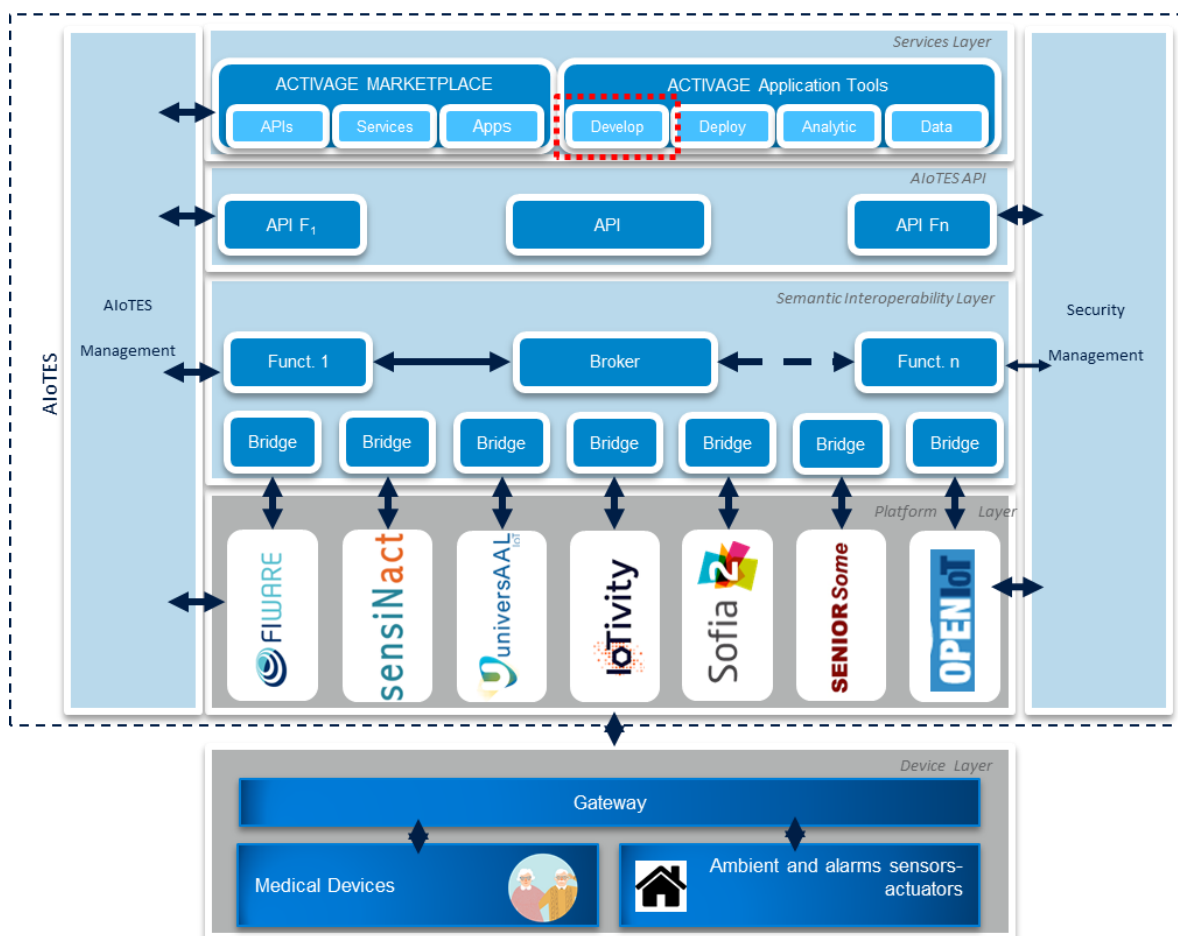


Figure 11: Positioning of the ACTIVAGE development tools within the overall ACTIVAGE architecture.

The purpose of the ACTIVAGE development tools component is to offer the appropriate development infrastructure which facilitates the creation of new IoT applications by technical developers, through the (re-)use of existing IoT applications already registered at the ACTIVAGE application ecosystem.

The development tools offer functionalities that can easily be reused by developers, in order to be integrated in a new application. Such functionalities include specification of IoT sensors to use, security mechanisms, integration with the data lake and data analytics applications,

etc. The ACTIVAGE development tools offer means for facilitating the use of existing applications by other developers, such as making use of available source code samples, browsing documentation, viewing available tutorials, testing sample code and using public or mocked data. The development tools also offer a link to the ACTIVAGE data analytics API, in order to include data analytics services in new applications, through easy-to-use tools.

Development tools allow the developer to use the ACTIVAGE REST API, in order to communicate with other ACTIVAGE components, such as the interoperability layer and the Data Lake. Development tools also offer a tight link with the ACTIVAGE deployment tools, described in Section 5, and their semantic discovery features, in order to support the composition and combination of already existing applications and tools registered at the ACTIVAGE ecosystem. In this way, a developer, even with limited technical knowledge, will be able to reuse existing tools created by other developers, and combine them into larger applications. For instance, a developer could search for existing tools providing security or logging mechanisms, in order to use them off-the-shelf within a new application.

The conceptual architecture of the ACTIVAGE development tools component and its connection to the other ACTIVAGE components is illustrated in Figure 12. The developer's toolkit communicates with the deployment tools component, via the latter's web API, in order to allow the developer to discover existing applications and combine existing tools as needed, in larger applications. The development tools also communicate with the Data Layer Support Tools, in order to allow the developer to use the Data Lake and Data Analytics infrastructure and functionalities for the development of applications. Finally, the development tools communicate with the Semantic Interoperability Layer (SIL), in order to have access to the semantics of devices and services registered to the AIOTES, and the semantics of the collected data. The SIL does not contain itself any data collected by devices and services; these are contained in the Data Lake and in the individual IoT platforms employed. Instead, the SIL contains the main ACTIVAGE ontology, which describes the *semantics* of devices, services and collected data, which are essential for the development of platform-agnostic applications. The functionalities offered by the development tools are wrapped in a Software Development Kit (SDK), offered as a set of Web services, ready to be used by developers.

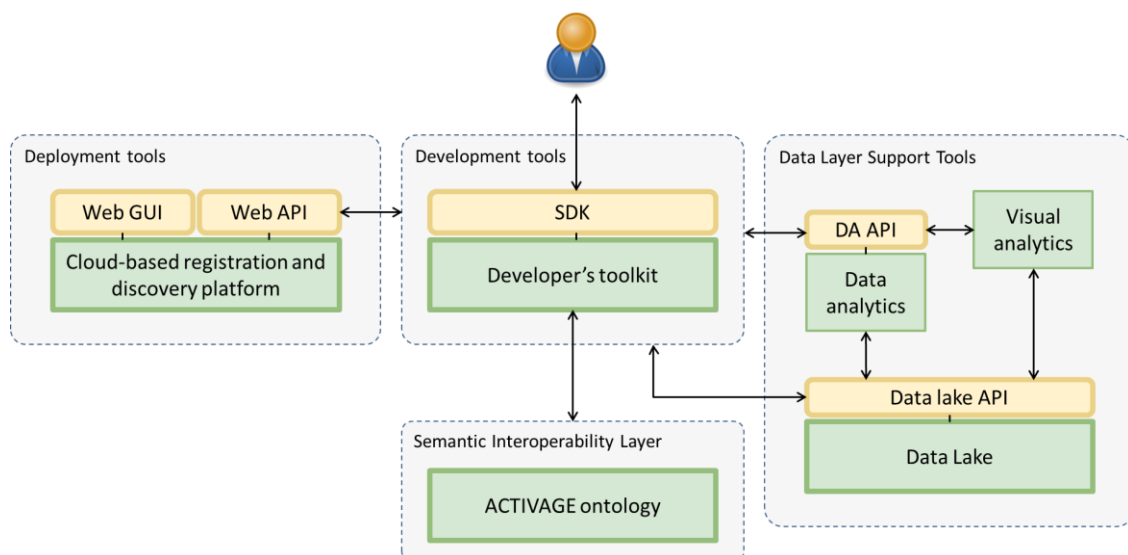


Figure 12: Architecture of the ACTIVAGE development tools component and its connection to the other ACTIVAGE components.

The requirements presented in Section 2 outline the functionalities that need to be covered by a kit of development tools. Individual IoT platforms offer their own tools in order to

facilitate developers in developing new applications. The AIOTES development tools need to be one level higher than these platform-specific tools, and offer functionalities that facilitate the use of AIOTES components, such as the Semantic Interoperability Layer or the Data Analytics, by developers. Thus, the focus of the ACTIVAGE development tools is on tools that are related to the AIOTES components, rather than on covering all aspects of software development, such as text editing and debugging, which are already well addressed by third-party software.

An overview of the ACTIVAGE development tools can be seen in Figure 13.

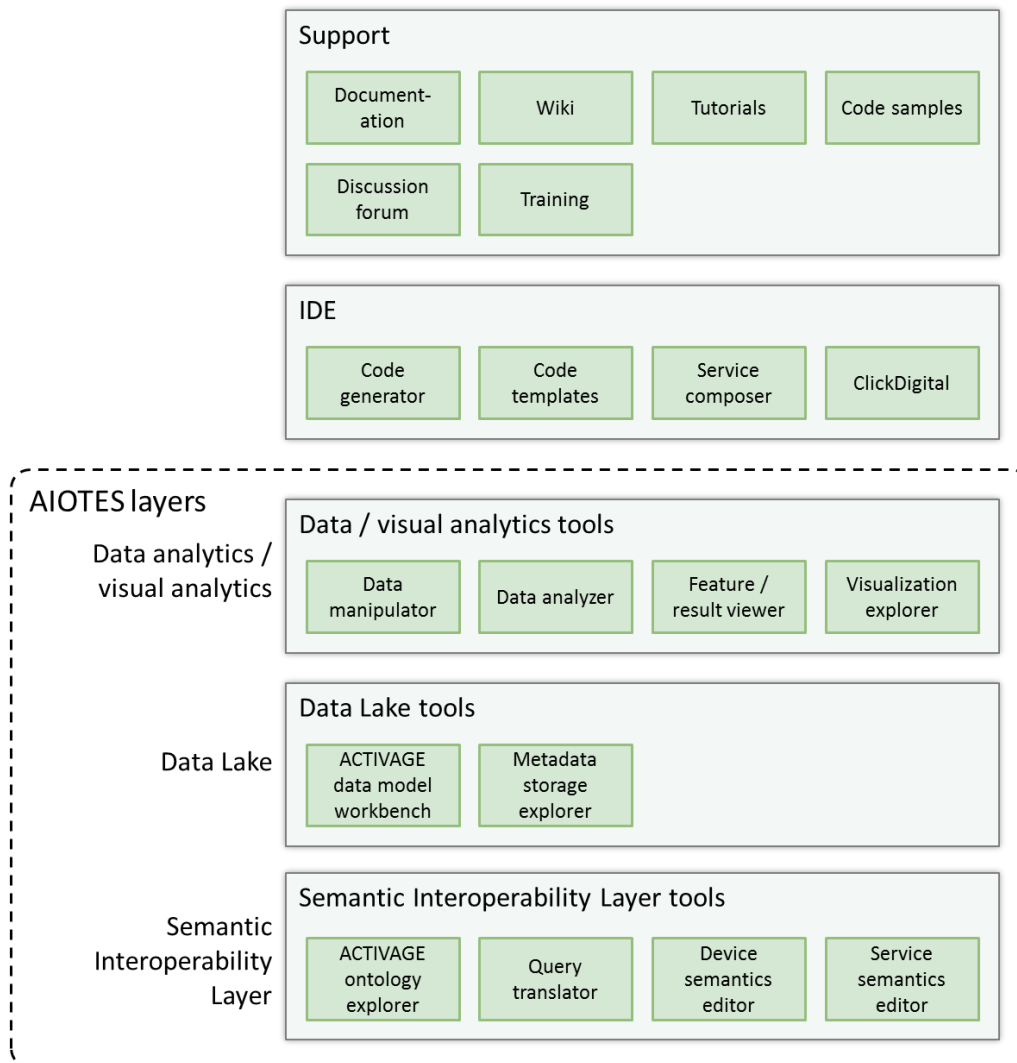


Figure 13: ACTIVAGE development tools.

The development tools are divided in the following categories:

- **Support:** Tools for providing documentation and instructions about using the AIOTES development tools.
- **Integrated Development Environment (IDE):** Tools for facilitating the creation of new applications.
- **Data / visual analytics tools:** Tools for facilitating the introduction of data analytics and visual analytics in an application.
- **Data Lake tools:** Tools for facilitating access to the data available through the Data Lake.

- **Semantic Interoperability Layer tools:** Tools for facilitating access to the Semantic Interoperability Layer ontologies.

The three categories at the bottom of Figure 13 correspond to architectural layers of the AIOTES, namely the Semantic Interoperability Layer, the Data Lake, the data analytics and the visual analytics components.

In the following sections, the ACTIVAGE development tools of each category are presented in detail, starting from the bottom layer, that corresponding to the Semantic Interoperability Layer, and moving upwards towards supporting material.

### 4.1.1 Semantic Interoperability Layer tools

The tools of this category aim at providing utilities for accessing the central ACTIVAGE ontology and functionalities of the Semantic Interoperability Layer (SIL). The SIL development tools, depicted in Figure 14, are the following:

- ACTIVAGE ontology explorer
- Query translator
- Device semantics editor
- Service semantics editor

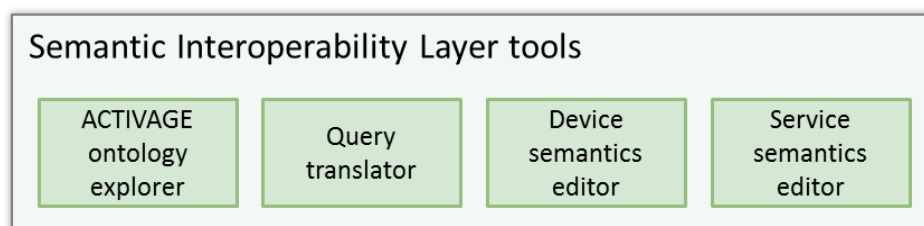


Figure 14: The Semantic Interoperability Layer (SIL) tools.

Each of the above tools is described in more detail below.

#### 4.1.1.1 ACTIVAGE ontology explorer

The ACTIVAGE ontology explorer provides access to the ACTIVAGE ontology that is the central part of the Semantic Interoperability layer. The developer can view the ontology schema, through entity-relation diagrams. By selecting specific entities or relations in the diagrams, the user can view details about the entities. The ontology schema can also be viewed in standard ontology description formats, such as OWL, and exported to files, which may be of use to the developer. The ACTIVAGE ontology explorer is useful to developers, since it allows them to see which types of devices, services, attributes, etc., which form the basic components of an application, are available in the AIOTES. In order for the ontology explorer to operate, it is directly connected to the Semantic Interoperability Layer of the AIOTES architecture. The main functionalities of the ACTIVAGE ontology explorer and its connection to the SIL are depicted in Figure 15.

#### Usage

The ACTIVAGE ontology explorer provides a Web-based Graphical User Interface, through which it presents the schema of the ACTIVAGE ontology, in an entity-relation diagram. The developer is able to navigate (pan, zoom) in the visualized ontology. By selecting an entity or relationship, more information regarding this entity/relation will be presented in a dedicated

panel (e.g. description). By selecting one or more entities/relations, or the entire ontology, the developer is also able to extract the corresponding textual representation, in standard ontology description formats (e.g. OWL).

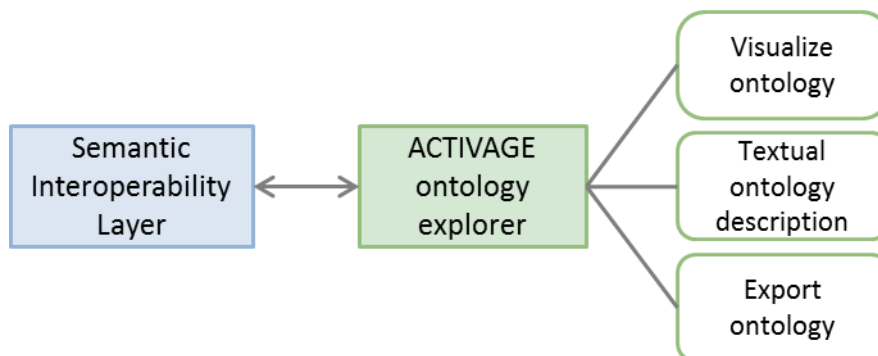


Figure 15: Functionalities and communication of the ACTIVAGE ontology explorer.

### 4.1.1.2 Query translator

The query translator is a utility exposing to the developer a central functionality of the Semantic Interoperability Layer, i.e. translating queries formulated for the ACTIVAGE ontology, into IoT platform-specific queries. Through an easy-to-use interface, the developer can write a data retrieval query, addressing it to the ACTIVAGE data model, and translate it to the schema of any of the 7 IoT platforms available in ACTIVAGE. The platform-specific queries may be useful to the developer in developing platform-specific applications. The inverse translation, from a platform-specific query to a query for the ACTIVAGE data model is also possible. In order for the query translator to operate, it is directly connected to the Semantic Interoperability Layer of the AIOTES architecture. The main functionalities of the query translator and its connection to the SIL are depicted in Figure 16.

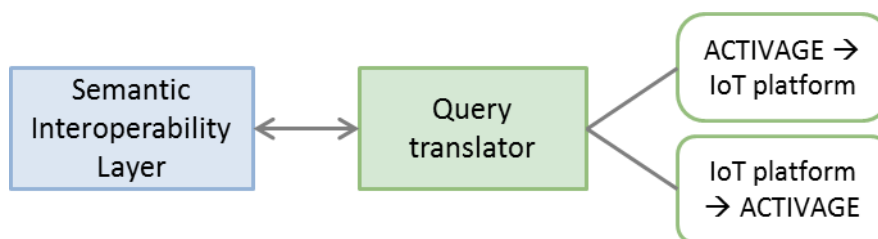


Figure 16: Functionalities and communication of the query translator.

#### Usage

The query translator offers a text editing window, through which the developer can write an input query, as well as drop-down menus for the selection of a target IoT platform. By clicking on a translation button, the input query is translated into the platform-specific naming conventions and database, and the result is presented to the developer in another text area. The developer is thus able to copy the produced query text, in order to use it within a platform-specific application.

### 4.1.1.3 Device semantics editor

The device semantics editor allows the developer to specify the semantics associated with the operation of a specific device, so that it can be registered to the ACTIVAGE ontology. Being able to edit the ontologies of the SIL is important in order to ensure that the ACTIVAGE system can be extended as new devices are being used. The device semantics editor is implemented as a form through which the developer can specify the semantics of



the device’s functionality, its parameters and outputs. The device semantics editor is connected to the SIL, in order for the registered device to be added to the overall ontology and be available to applications. However, care will be taken (e.g. role-based access rights) in order to ensure that the semantics of existing devices currently used by applications are not altered, and thus avoid application malfunctioning. The purpose of the device semantics editor is to allow the registration of new devices in the SIL. The connection of the device semantics editor to the SIL is depicted in Figure 17.

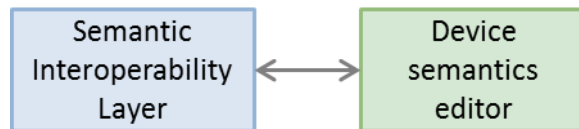


Figure 17: Connection of the device semantics editor to the SIL.

**Usage**

The device semantics editor, through its graphical user interface, presents the developer with a list of all registered devices, from which the developer can select one, in order to edit its semantics (the tool can also be linked to other tools, such as the ontology explorer, so that the developer can select a device through the ontology explorer). By selecting a device, a form is presented to the developer, where details about the device semantics can be entered and submitted to the ACTIVAGE ontology. The developer can also create a new device type and add it to the ACTIVAGE ontology, by selecting a “New device” option in the tool’s GUI. The details for the new device are provided through a similar form as the one used for editing.

4.1.1.4 Service semantics editor

Similar to the device semantics editor, the service semantics editor allows the developer to specify the semantics associated with the operation of a specific service/application, so that it can be registered to the ACTIVAGE ontology and be discoverable by other developers. The service semantics editor is implemented as a form through which the developer can specify the semantics of the device’s functionality, its inputs and outputs. The service semantics editor is connected to the SIL, in order for the registered service to be added to the overall ontology and be available to developers and deployers. The connection of the service semantics editor to the SIL is depicted in Figure 18.

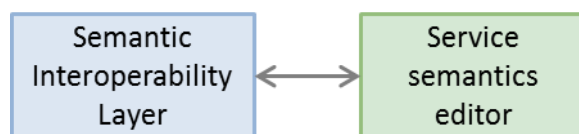


Figure 18: Connection of the device semantics editor to the SIL.

**Usage**

The service semantics editor, through its graphical user interface, presents the developer with a list of all registered services, from which the developer can select one, in order to edit its semantics (the tool can also be linked to other tools, such as the ontology explorer (Section 4.1.1.1) so that the developer can select a service through the ontology explorer). By selecting a service, a form is presented to the developer, where details about the service semantics can be entered and submitted to the ACTIVAGE ontology. The developer can also create a new service and add it to the ACTIVAGE ontology, by selecting a “New service” option in the tool’s GUI. The details for the new service are provided through a similar form as the one used for editing.

## 4.1.2 Data Lake tools

The Data Lake development tools provide utilities for accessing the Data Lake component of the AIOTES architecture. The Data Lake provides an entry point for the developer to access the data available in the ACTIVAGE deployment sites, as well as to access the metadata used by data analytics methods. As described in D4.2, “Data Layer Support Tools”, the Data Lake, architecturally, lies in the data layer, above the SIL. The Data Lake provides data storage for IoT platforms not having their own, stores analytics metadata, and directs queries towards the SIL, integrating the results. The Data Lake development tools correspond to these main Data Lake functionalities and are the following, as depicted in Figure 19:

- ACTIVAGE data model workbench
- Metadata storage explorer

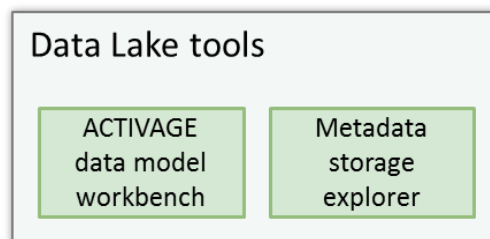


Figure 19: The Data Lake modules.

Each tool is described in the following sections.

### 4.1.2.1 ACTIVAGE data model workbench

The ACTIVAGE data model workbench is an environment through which the developer can view the structure of the ACTIVAGE data model and the data available in the distributed databases of the IoT platforms. The environment is similar to common database management workbenches, such as MySQL workbench or pgAdmin. It allows the developer to see the structure of the ACTIVAGE data model, as if it is a database, with its tables and schemas. By selecting an entity (table), e.g. “temperature\_sensors”, the developer can view the data available for this entity. The data are presented as if they were contained in a single table, in a single database using the ACTIVAGE data model schema; however, they are in fact collected dynamically from the multiple diverse IoT platform databases, through automatic translations performed by the Semantic Interoperability Layer. The developer can formulate and submit queries to the ACTIVAGE schema, which are again automatically translated by the SIL, and retrieve collected results. This facilitates experimenting with queries, in order to achieve a desired outcome. The submitted queries can then be copied into the source code of developed applications, as needed. The retrieved data can also be exported as needed. In order for the ACTIVAGE data model workbench to perform, it is connected to the Data Lake component of the AIOTES architecture, as depicted in Figure 20.

#### Usage

The ACTIVAGE data model workbench offers a Graphical User Interface (GUI) for viewing the data available and executing queries. The GUI consists mainly of three components:

- A tree view of the ACTIVAGE schema, in the form of tables and table columns, from which the developer can select different tables to view data from.
- A text area, through which the developer can write queries and execute them.

- A table view, for presenting the contents of a data table, after the user has selected one from the tree view, or for presenting the results of submitted queries.

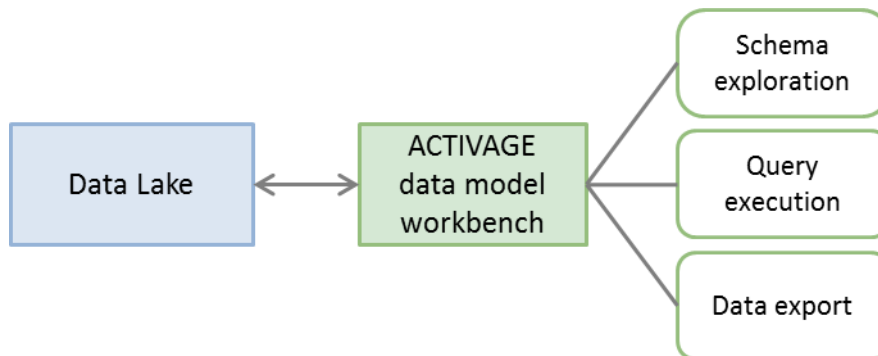


Figure 20: Functionalities and communication of the ACTIVAGE data model workbench.

The purpose of the workbench is to allow the developers to experiment with queries, in order to compose ones that meet their needs, before including them in their developed applications.

#### 4.1.2.2 Metadata storage explorer

The metadata storage explorer allows the developer to explore the metadata produced by data analytics methods and stored in the Data Lake. The interface is similar to the ACTIVAGE data model workbench, allowing the developer to view the available schema and perform queries. The retrieved metadata, such as features, thresholds, etc., can be exported for further use in applications, tests and debugging sessions. The functionalities of the metadata storage explorer and its connection to the Data Lake can be seen in Figure 21.

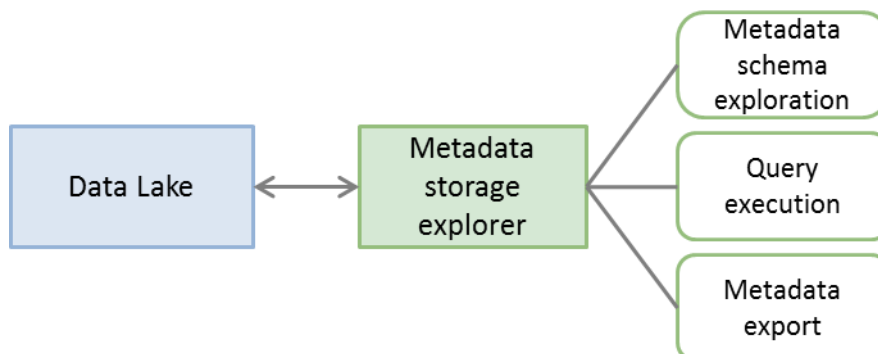


Figure 21: Functionalities and communication of the metadata storage explorer.

#### Usage

The Metadata storage explorer offers a GUI for viewing the metadata available and executing queries. The GUI is similar to the ACTIVAGE data model workbench (Section 4.1.2.1) and consists mainly of three components:

- A tree view of the metadata schema, in the form of tables and table columns, from which the developer can select different tables to view metadata from.
- A text area, through which the developer can write queries and execute them.
- A table view, for presenting the contents of a metadata table, after the user has selected one from the tree view, or for presenting the results of submitted queries.

The purpose of the workbench is to allow the developers to experiment with queries and see which kind of information is stored in the metadata, in order to finally use them during the development of data analytics or other applications.

### 4.1.3 Data / visual analytics tools

The data and visual analytics development tools allow the operator to perform data analytics methods and view results and visualizations on-the-fly, using custom data. The developer can thus experiment with different analytics and visualization methods and their parameters and see the results of his/her actions, before using them inside an application under development. They consist of the following tools, depicted in Figure 22:

- Data manipulator
- Data analyser
- Feature / result viewer
- Visualization explorer

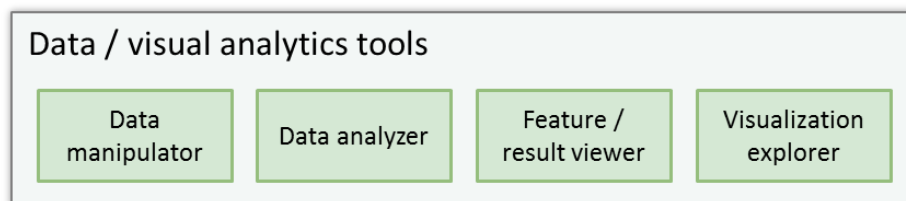


Figure 22: The data / visual analytics development tools.

The data analytics and visual analytics tools are based on the data analytics and visual analytics Web APIs, described in Deliverable D4.2 “Data Layer Support Tools”, and offer graphical user interfaces for easy integration of analytics services in developed applications.

#### 4.1.3.1 Data manipulator

The data manipulator offers functionalities for pre-processing data, before further analysis is performed. Data can be inserted either manually or through a query to the Data Lake. The data are viewed in the form of a spreadsheet, from where the developer can select attributes, filter records, perform transformations, etc. The source code corresponding to the pre-processing actions performed by the developer (Data Lake queries, transformations, etc.) can be exported, in order to be used inside developed applications. The data manipulator is connected to the Data Lake, in order to retrieve the available data, as depicted in Figure 23.

##### Usage

The Data manipulator offers a Web-based GUI, with the following main components:

- A data insertion panel, through which the developer can specify the input data to be manipulated. The data can be inserted in one of two ways:
  - Through a “Data upload” operation, where the developer is able to select and upload local data files, e.g. in CSV or JSON formats.
  - By writing a query to the Data Lake in a dedicated text area and executing it. The data retrieved are the ones used for further manipulation (The ACTIVAGE data model workbench (Section 4.1.2.1) can be used at this point to assist the developer in composing a query that meets his/her needs).

- A table view of the inserted data. Each row corresponds to a record, while the columns correspond to the different attributes available for each record. Through the table view, the developer is able to perform the following functionalities:
  - Filter records, by inserting filters for each attribute (column). The filters can include selecting a specific attribute, or a range of attribute values.
  - Select/deselect subsets of attributes (columns).
  - Create new attributes, by transforming existing ones (e.g. multiplying by a value or adding two attributes).

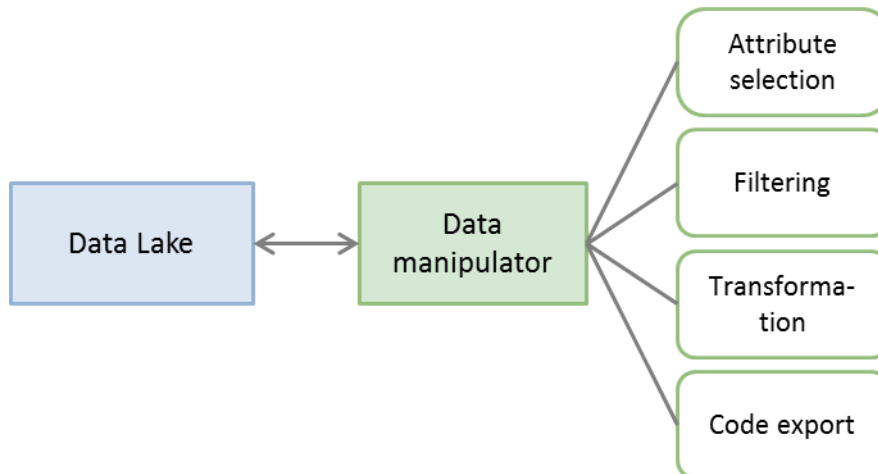


Figure 23: Functionalities and communication of the data manipulator development tool.

The final data, after these preprocessing operations, can be provided as input to the data analyser tool (Section 4.1.3.2), presented below. By selecting an “Analyse” option, the Data analyser tool is opened, with the preprocessed data pre-loaded as its input.

### 4.1.3.2 Data analyser

The data analyser tool provides an interface through which the developer can experiment with the analytics methods available in the Data Analytics component of the AIOTES. The data analyser takes as input a set of data (usually pre-processed by the data manipulator tool), and provides the developer with a GUI, through which he/she can select from the available analytics methods and adjust their parameters. The available analysis types are those supported by the Data Analytics component, i.e. feature extraction, clustering, anomaly detection, hypothesis testing, etc. The results of an analysis, e.g. extracted features or clusters, can be viewed using the feature / result viewer (see below). Through the data analyser, the developer can choose the main analysis entities, select among different analysis methods and configure them, by adjusting their parameters through visual controls (sliders, etc.). The results of the analysis can be instantly viewed in the result viewer (within the time limitations of each algorithm), which facilitates the developer in selecting the appropriate analysis method and its parameters, to use in a developed application. The source code corresponding to the selected methods and parameters can be exported, in order to be used during application development. The data analyser is connected to the Data Analytics component, in order to perform the analyses, as well as to the data manipulator tool (for reading its input) and the feature / result viewer tool (for showing its output), as depicted in Figure 24.

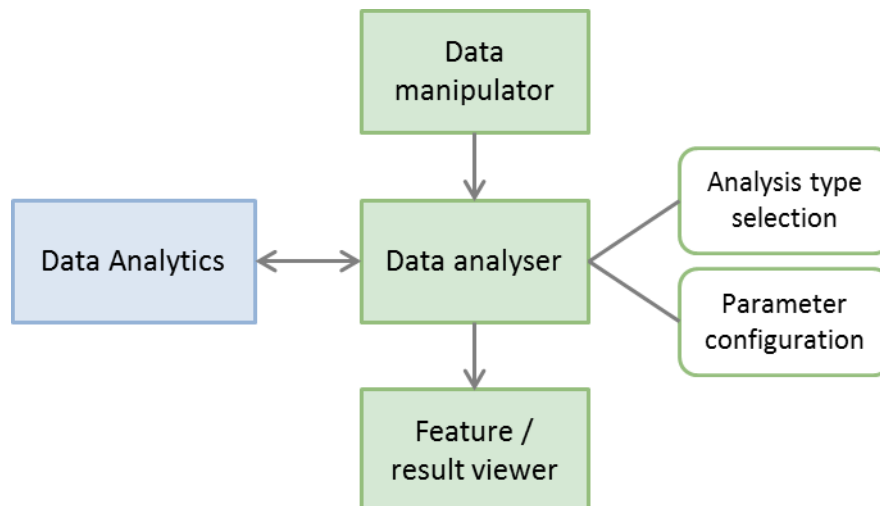


Figure 24: Functionalities and communication of the data analyser development tool.

## Usage

The Data analyser tool offers a Web-based GUI, with the following main components:

- A data selection panel, through which the developer can select data to analyse, in one of the following ways:
  - Direct data upload, where the developer can upload a local data file, e.g. in CSV or JSON format.
  - Data Lake query, where the developer can write and execute a query towards the Data Lake and perform the analysis on the retrieved data.
  - In case the user has pre-processed data through the Data manipulator tool (Section 4.1.3.1), the pre-processed data are already pre-loaded in the data selection area.
- An analysis type selection drop-down menu, through which the developer can select one of the available analysis types (e.g. feature extraction, dimensionality reduction, anomaly detection, clustering, etc.) to perform on the data. More information about the analysis types supported can be found in deliverable D4.2 “Data Layer Support Tools”.
- A panel through which attributes of the loaded data can be selected for the specific type of analysis (e.g. specifying that anomaly detection will be performed on the “blood pressure” data attribute).
- A panel for configuring the parameters of the specific analysis type, e.g. thresholds for anomaly detection or the number of clusters used in clustering. The parameters can be selected through text boxes or sliders.
- A table view of the analysis results, e.g. the list of record ids, with an added “anomaly score” column, indicating the “outlierness” of each record. The analysis results can be downloaded by the developer, or linked to the Feature/result viewer tool (Section 4.1.3.3) presented in the next section, for visualization.
- A Data Analytics API call viewer, which is a text area showing, at any time, the calls made to the Data Analytics Web API, each time the developer performs an analysis or modifies a parameter. The developer can copy the presented calls, in order to use them inside a developed application.

### 4.1.3.3 Feature / result viewer

The feature / result viewer development tool is used to present the results of an analysis in an easy-to-perceive way. The analysis entities chosen through the data analyser tool, are presented both in a spreadsheet-type form and using a few visualization methods. The spreadsheet contains the analysis entities in their raw form, or the features extracted from them. The visualization depicts each entity as a point on the screen, whose coordinates are determined by its attributes or features. Any other analysis results, such as clustering labels or anomaly detection scores, are visualized by using visual attributes such as color or size, either in the spreadsheet or in the visualization view. These types of presentation, in combination with the parameter controls of the data analyser tool, allow the developer to quickly see the results of different extracted features, different clustering parameters, different anomaly detection thresholds, etc. on the analysed data, before they are used in an application. In order for the feature / result viewer to perform, it is connected to the data analyser tool, as well as to the Visual Analytics component of the AIOTES, as depicted in Figure 25.

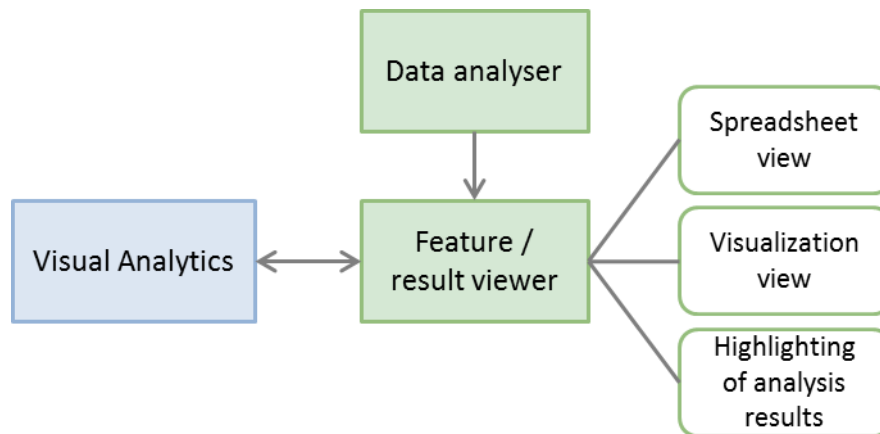


Figure 25: Functionalities and communication of the feature / result viewer development tool.

#### Usage

The Feature / result viewer tool offers a Web-based GUI, through which the developer can visualize the analysis results, as they are produced by the Data analyser tool. The main panel of the viewer is a visualization area, where a few types of visualizations (scatterplots or line plots) are used to visualize the analysis results. Each record of the selected data for analysis is represented by a visual object (e.g. a point), whose visual attributes (position, color, size, etc.) are mapped to selected attributes of either the original data or the analysis results. For instance, blood pressure data may be visualized by a line plot, having timestamps as the horizontal axis, blood pressure data as its vertical axis, while the color of the corresponding points depends on the anomaly score, as computed by an anomaly detection algorithm. The purpose of the feature / result viewer is to allow a quick overview of the analysis results, being directly linked to the parameter tuning functionality of the data analyser tool (Section 4.1.3.2), in order to facilitate the developer in analysis and parameter selection.

### 4.1.3.4 Visualization explorer

The visualization explorer tool facilitates the developer in selecting the most appropriate visualization type for a set of data, in order to use it within an application or dashboard. Through the visualization explorer's GUI, the developer can select among the visualization types available in the Visual Analytics component of the AIOTES. The developer can test the

selected visualization using data provided either directly or using a query to the Data Lake. The developer can select which data attributes are mapped to each visualization type's visual attributes. The source code used to generate the selected visualization can be exported, in order to be used within an application under development. In order for the visualization explorer to operate, it is connected to the Data Lake, for data retrieval, and to the Visual Analytics component, for using the available visual analytics methods, as depicted in Figure 26.

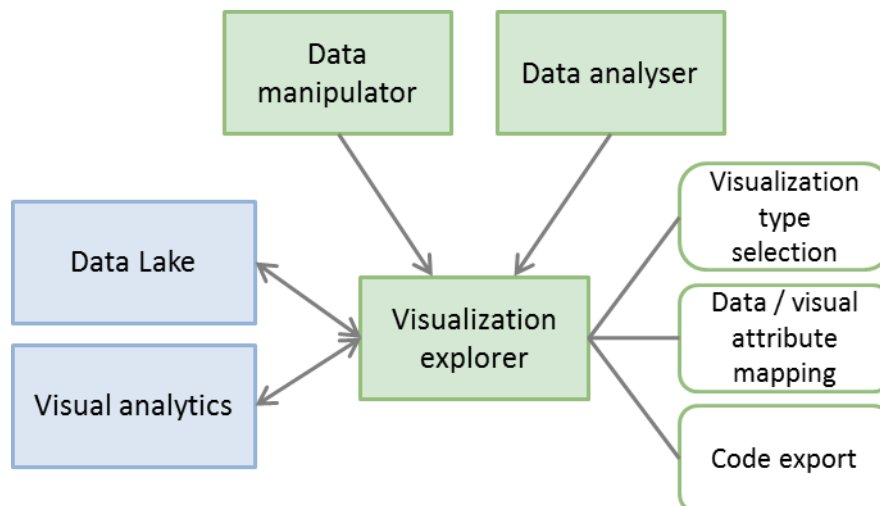


Figure 26: Functionalities and communication of the visualization explorer development tool.

## Usage

The visualization explorer offers a Web-based Graphical User Interface, through which the developer can load data and select among different visualizations to use. Its main components include the following:

- A data selection panel, similar to (or the same as) the one used in the data manipulator tool (Section 4.1.3.1), from which the user can select the data to visualize, in one of the following ways:
  - Direct data upload, where the developer can upload a local data file, e.g. in CSV or JSON format.
  - Data Lake query, where the developer can write and execute a query towards the Data Lake and perform the analysis on the retrieved data.
  - In case the user has pre-processed or analysed data through the Data manipulator (Section 4.1.3.1) or Data analyser (Section 4.1.3.2) tools, the pre-processed or pre-analyzed data are already loaded in the data selection area.
- A visualization type selection drop-down menu, from which the developer can select among the different visualization types supported (bar charts, line charts, scatterplots, graph-based visualizations, etc.). More information about the visualization types supported can be found in deliverable D4.2 “Data Layer Support Tools”.
- A visualization panel, where the data visualization is displayed. After the user selects a visualization type to use, various visualization attributes, specific to the visualization type selected (e.g. position, color, size), can be mapped to attributes of the loaded data (e.g. blood pressure levels).
- A code extraction text area, where the background visualization Web API calls made to produce the presented visualization, or the HTML/JavaScript/code used to produce it,



are displayed, so that the developer can copy-paste them inside the source code of an application under development.

#### 4.1.4 Integrated Development Environment (IDE)

The Integrated Development Environment (IDE) contains tools that facilitate the creation of new applications by developers. The ACTIVAGE IDE contains the following tools:

- Code generator
- Code templates
- Service composer
- ClickDigital IDE

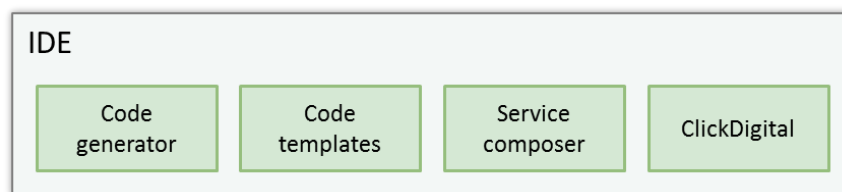


Figure 27: The Integrated Development Environment (IDE) components.

##### 4.1.4.1 Code generator

The code generator development tool provides generated code for manipulating devices and sub-services, within a larger application. The code generator allows the developer to select among available components and services using a graphical user interface. After selecting a component (or dragging it in a design area), the developer can further select among the functionalities available for this component, as they are registered in the ACTIVAGE ontology. The developer can generate source code for implementing these functions, compatible with any of the individual IoT platforms registered in ACTIVAGE. The translation of the component's characteristics to the naming conventions of each platform is handled by the Semantic Interoperability Layer, in the background. The functionalities and communication of the code generator development tool are depicted in Figure 28.

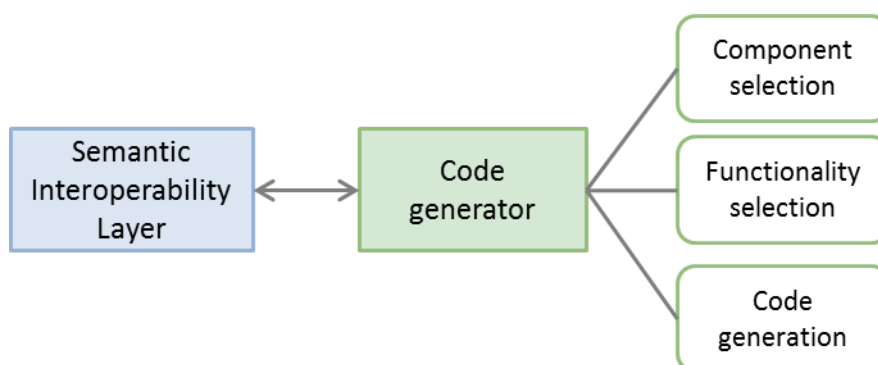


Figure 28: Functionalities and communication of the code generator development tool.

##### Usage

The code generator tool offers a Web-based Graphical User Interface, which acts as a mini Integrated Development Environment (IDE). The code generator, through a list view, allows the developer to select devices and services registered in the AIOTES. After a device or service is selected, the functionalities associated with this device/service are presented to

the developer. For instance, the developer can select a temperature sensor, which has a “get status” and a “collect data” functionality, or a lamp, which has a “get status”, a “switch on” and a “switch off” functionality. The developer can select the desired functionality from a menu and generate the source code needed to perform this functionality for the specific device/service. The source code is displayed in a dedicated text area and the developer is able to select the target platform for which to generate the code.

#### 4.1.4.2 Source code templates

Source code templates provide starting points for the development of applications. They provide minimal yet buildable applications, which can then be modified by the developer. They ease development by freeing the developer of all the details needed to be considered for a minimal application to run. Source code templates are provided for developing applications for various platforms, including the following:

- **Server / client applications**, for implementing IoT data management services
- **Mobile applications**, for implementing applications meant to be run in mobile devices
- **Web applications**, for implementing Web services or visual interfaces (dashboards) for management of IoT devices and data.

The ACTIVAGE source code templates also facilitate the developer in creating a new application for the individual IoT platforms registered in the ACTIVAGE federation. For each of the above application platforms and types, code templates and sample applications exist for each of the underlying platforms, in order to get the developer started quickly. Semantic mappings from the Semantic Interoperability Layer can be used to provide these templates with data-related components, which are automatically translated to the database schemas of the individual platforms. The functionalities and communication of the source code templates module are depicted in Figure 29.

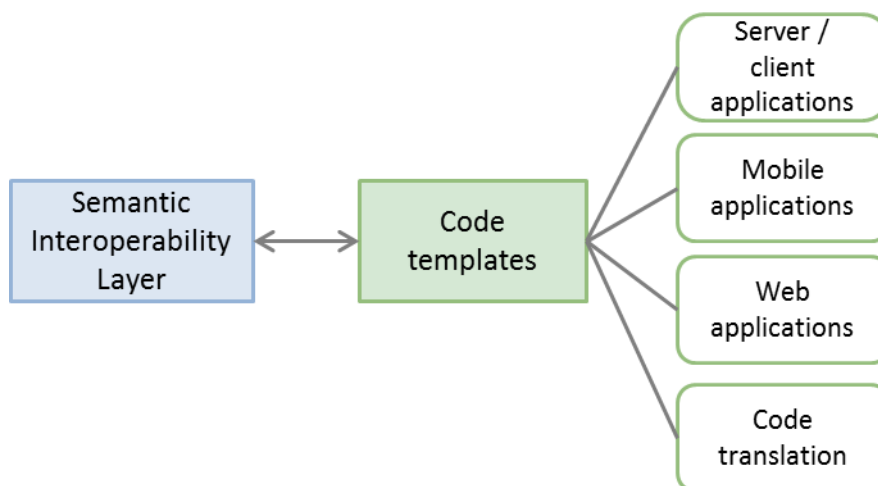


Figure 29: Functionalities and communication of the code templates development tool.

#### Usage

The source code templates tool contains code templates for several application types or functionalities, which the developer can use while creating applications either for the AIOTES or for the underlying IoT platforms. The code templates tool presents the developer with a list of all available templates, e.g. templates for server/client applications, for mobile applications, for web applications, for collecting data, for controlling devices, etc. After the developer selects a desired template, the code templates tool allows him/her to select the target platform, where the code will be used. After the platform is also selected, the template

code is displayed in the main text area, where the developer can insert actual values and names in the template's placeholders, in order to instantiate the template for a specific application.

### 4.1.4.3 Service composer

The service component development tool facilitates the developer in combining existing services, tools and applications, in order to compose larger applications. The developer can search for existing services by using the semantic service discovery tool, described above. After the developer finds appropriate services, he/she can combine them into larger workflows. The service composer provides two types of interface to support the composition functionality: a textual interface, where the developer can describe how the services are combined in an XML-style format, and a visual interface, where the developer can describe how the services are combined by visually connecting building blocks. The two interfaces are linked to each other, so that the developer can work with the most appropriate one at any moment. The service composition involves the determination of each service's inputs and outputs and the connection of one's outputs to another's inputs. A characteristic example is combining data analytics methods, in order to perform a complex analysis: the raw data may pass through a data filtering service, before they enter a feature extraction procedure. The extracted features may then be used as input in an anomaly detection method, in order to find out entities not behaving in the expected manner. The developer can subsequently generate the source code corresponding to the desired combination, in order to import it into an application under development. The service composer tool implementation will be based on similar service composer tools used in currently running or previous European Projects, such as IN LIFE<sup>2</sup> and Cloud4All<sup>3</sup>. However, new functionalities will be needed, in order to cover the services and functionalities of ACTIVAGE. In order for the service composer to perform, it is connected to the Semantic Interoperability Layer, as depicted in Figure 30, in order to have access to each service's semantics for its functionalities, inputs and outputs, so as for the combination to become possible.

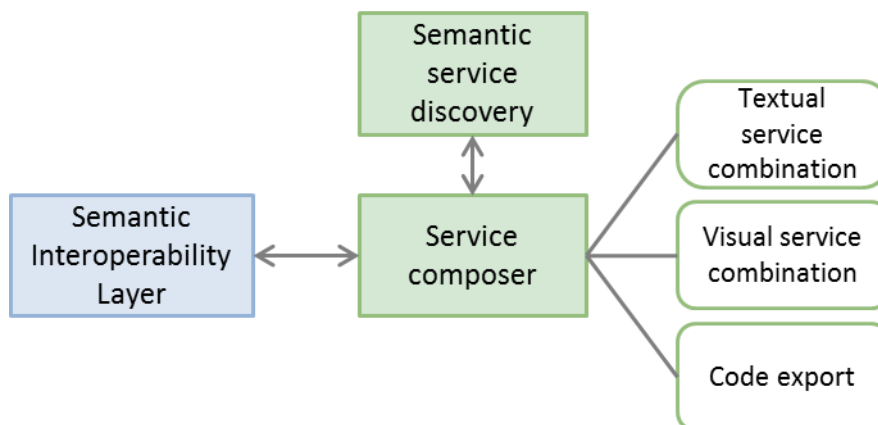


Figure 30: Functionalities and communication of the service composer development tool.

#### Usage

The service composer tool offers a Web-based Graphical User Interface that allows the developer to compose services. The main components of the interface are the following:

<sup>2</sup> <http://www.inlife-project.eu/>

<sup>3</sup> <http://www.prosperity4all.eu/related-projects/stay-connected/>

- A service discovery panel, displayed as a form, through which the developer can insert the characteristics of a needed service, or use a similar service as an example, in order to find services with similar semantics, that fit to the developer's needs. The discovered services are presented in a list, allowing the developer to select each of them in order to view its details.
- The composition area, where the selected services can be inserted and connected to each other. The composition area can be displayed in one of two forms:
  - A text area, where an XML-like description of the services and their connections is edited. The developer can insert the description of a service, in the composition-specific syntax, by selecting the service from the discovery panel. The developer can also specify how the services are connected to each other, by manually inserting specific directives.
  - A visual interface, where the service composition is displayed as a block diagram. The developer can insert a block, representing a service, by selecting a service from the discovery panel. The inserted service is presented as a block, having its inputs and outputs as points on its border that can be linked to the inputs and outputs of another service.
- The code export area, for exporting the service composition (described visually or textually) as a new application or as source code that can be included in a new application. This exported source code contains the specific low-level API calls and logic that needs to be performed in order for the service composition to work.

#### 4.1.4.4 ClickDigital user friendly IDE for IoT platforms

ClickDigital is a Visual and Pluggable User Friendly Integrated Development Environment for IoT platforms. It allows you to quickly prepare smart digital solutions and offer the resulting App/Dashboard (in use only mode) to your clients. ClickDigital aims mainly:

To decrease the learning curve/complexity of creating Apps for heterogenous IoT platforms

- To offer a new App creation experience for the developers/consumers of IoT solutions
- To optimize the path/time to the market
- To enable the IT departments to develop, optimize the cost and enhance the usage of IoT solutions

Main extents behind ClickDigital are:

- To Increase and accelerate the frequency of use and creation of IoT solutions
- To Increase the use of IoT solutions
- To Enable the for "IoT for all" strategic vision

ClickDigital is being addressed mainly to:

- IoT/Digital solutions & applications developers: The IoT/Digital solutions & application developers present a first level of our target users group, mainly in order to:
  - Optimize the needed time to learn an IoT platform's philosophy and structure to be able to start developing compatible applications.
  - Optimize the programming complexity and reduce the blocked status
  - Perform the time usage to extend and empower the created system quality
  - Reduce time to market

- The IT departments of different services providers companies in relation with the IoT verticals: The IT departments present the strategic target group behind ClickDigital as it will enable the correspondent IoT service providers to:
  - Act independently from the developers
  - Have more space for quick prototyping of ideas to be stress tested towards the market
  - Shorten the path to market
  - Enable the quick time reaction toward smart solutions adjustment, modification and extension
  - Empower the user centric approach through the direct contact between the IT departments and the respective services providers.

In the Activage context, and in order to enable/scale up the usage of IoT solutions, ClickDigital would fit the big picture, mainly based on the following characteristics:

- Just a “Pluggable” Visual IDE to IoT platforms to create your own IoT dashboard and present it in different modus to your clients
- Widget based
  - Cover the spectrum of needed use cases for a successful IoT application, mainly
    - Management
    - Visualization
    - Control
    - Logic creation

Therefore, this tool will enable the Visual programming of IoT solutions based on different IoT platform and AIOTES also, it will cover also tje mainly following domains:

- Smart Living
- Building management
- eHealth
- Energy economy

...

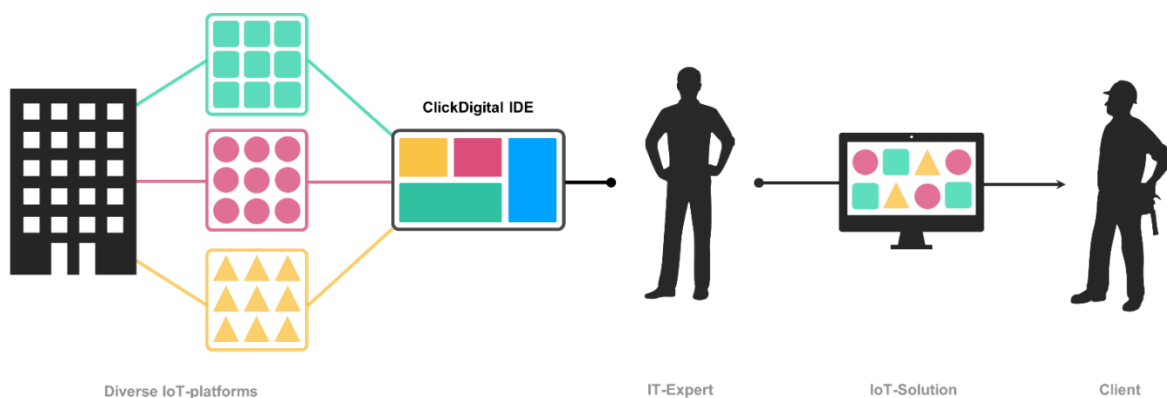


Figure 31: ClickDigital IDE for programming IoT solutions on top of several IoT platforms

The ClickDigital IDE communicates with the Semantic Interoperability Layer, in order to have access to the available components and services that can be used for visual dashboard creation. The functionalities and communication of the ClickDigital IDE are presented in Figure 32.

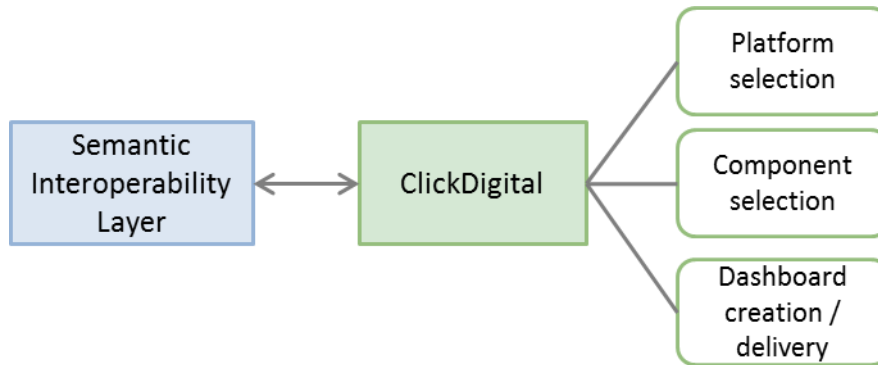


Figure 32: Functionalities and communication of the ClickDigital IDE.

### Usage

As shown in Figure 33, the 3 main added values of ClickDigital are to:

- Plug,
- Create, and
- Deliver

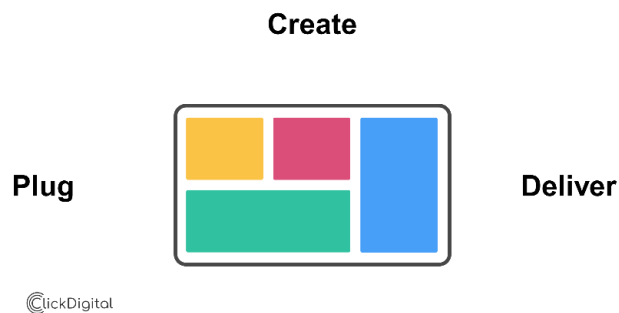
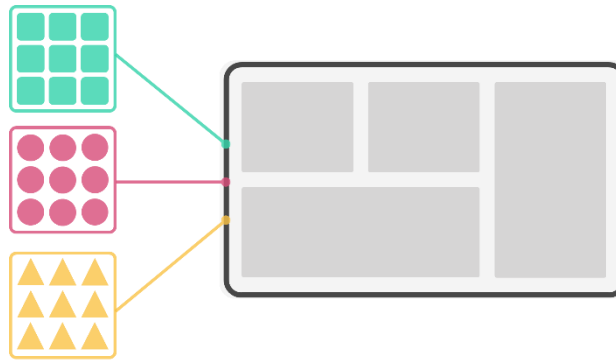


Figure 33: Added values of ClickDigital IDE.

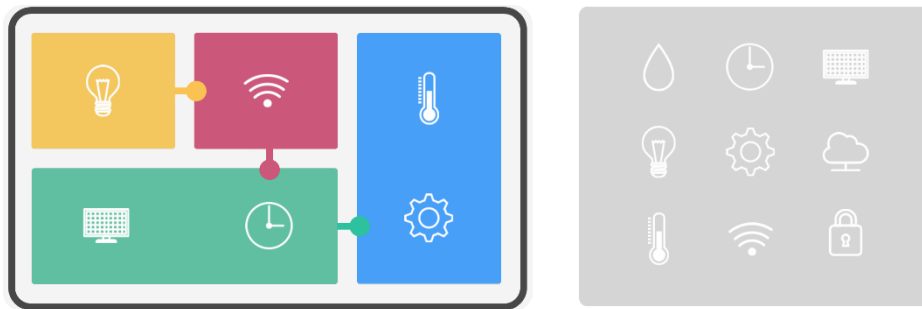
The Plug capabilities aims to offer the possibilities to the ClickDigital experts to plug it to different IoT Platforms based on the partners/clients requirements (Figure 34).



©lickDigital

Figure 34: Plug capabilities of the ClickDigital IDE.

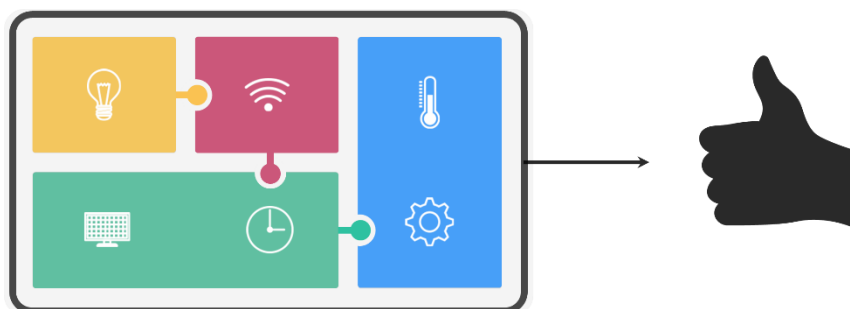
Once plugged, ClickDigital, through its marketplace-based Widget, allow the users to select, drag and drop different widgets to the main dashboard screen, this will enable them to Vizually plug new devices, control them, vizualize data and create smart rules.



©lickDigital

Figure 35: Create capabilities of the ClickDigital IDE.

Once created, ClickDigital offers the possibility to deliver the created application/dashboard in a use-only mode to the target users.



©lickDigital

Figure 36: The deliver capabilities of the ClickDigital IDE.

## 4.1.5 Support

The support modules provide resources and documentation, in order to facilitate the developer in developing applications within the AIOTES infrastructure and resolving issues. They offer support for the development of AIOTES applications, in a similar manner as the support tools of T5.3 offer support for the use of AIOTES applications. The support modules include the following, as also depicted in Figure 37:

- **Documentation:** Detailed documentation of all available ACTIVAGE Web APIs, and development tools, with instructions for usage, description of inputs and outputs, etc.
- **Wiki:** Wiki pages, editable by the developers, providing information about issues that commonly arise while developing IoT applications in the ACTIVAGE ecosystem.
- **Tutorials:** Step-by-step guides for common tasks, which facilitate new developers in specific application types.
- **Code samples:** Example source code snippets or complete applications, for performing common tasks, which can be readily used and modified by developers.
- **Discussion forum:** A dedicated forum, where developers can communicate with each other, ask questions and provide answers to arising issues.
- **Training:** The training module includes webinars, live demos, etc., useful for training developers in using the AIOTES components and development tools.

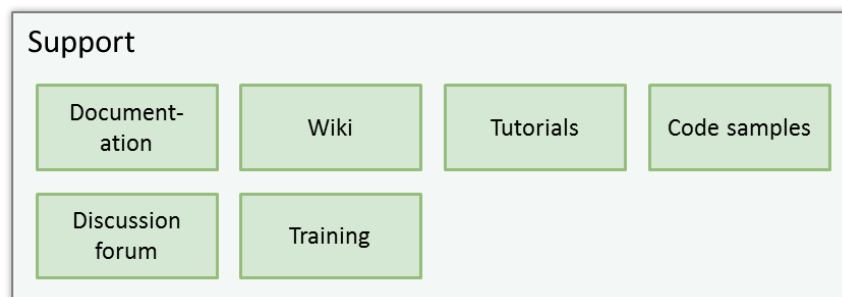


Figure 37: The support development tools.

### Usage

The support tools are available to the developer through a Web-based help center. Hyperlinks to the primary categories of available materia, i.e. documentation, wiki, tutorials, code samples, discussion forum and training, are provided through the support tools home screen. The support tools offer links to other parts of the support tools, as well as examples of usage for all parts of the AIOTES. They also offer links to the platform-specific documentation websites.

## 4.1.6 Mapping between development tools requirements and modules

The development tools, as described in the sections above, cover the development tools requirements outlined in Section 2. Figure 38 provides the mapping between requirements and modules. Orange boxes denote requirements, while green boxes denote the corresponding ACTIVAGE development tools. Part of the requirements is mapped to deployment tools (grey boxes), which are described in Section 5.1.



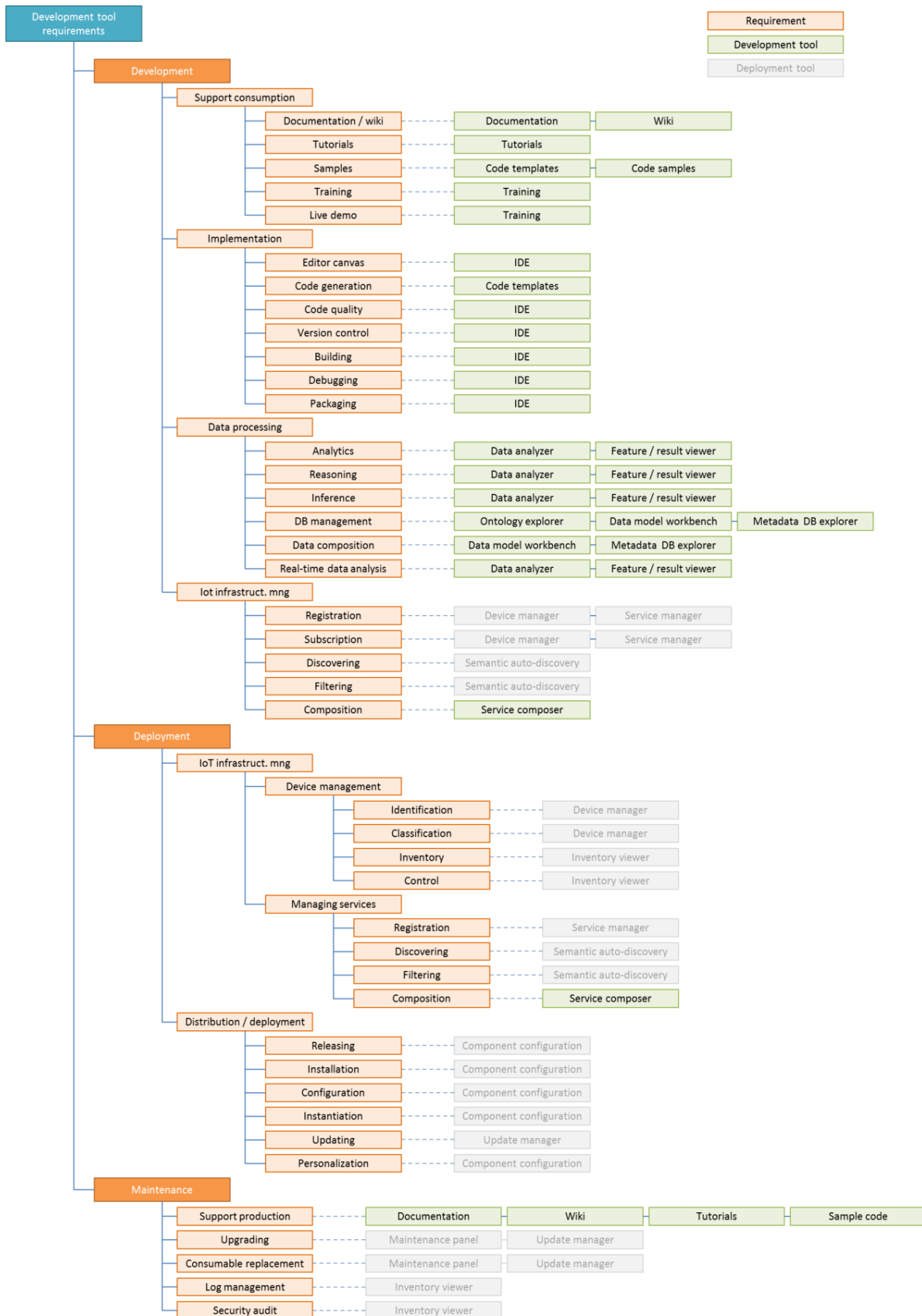


Figure 38: Mapping between requirements (orange) to the ACTIVAGE development tools (green).

## 4.2 Available development tools supported by the ACTIVAGE IoT platforms

### 4.2.1 universAAL

The universAAL developer tools were developed to simplify and assist the developer in developing smart services and components for the universAAL platform. This section will provide an overview about the available tools that will be used by end users and deployers to install, configure and personalize universAAL based applications.

The universAAL platform is developed in Java, using Maven as basis for project management, and GIT for source versioning control. All of its code is publicly available in github at: <https://github.com/universAAL>. Taking also advantage of its project tools (such as issue management), and documentation (such as wikis and github pages).

#### 4.2.1.1 Support Tools

Table 1: universAAL support tools

<b>Documentation</b>	<a href="https://github.com/universAAL/platform/wiki">https://github.com/universAAL/platform/wiki</a>
<b>API Doc</b>	<a href="http://universaal.github.io/platform/apidocs/index.html">http://universaal.github.io/platform/apidocs/index.html</a> <a href="https://github.com/universAAL/remotes/wiki/REST-API">https://github.com/universAAL/remotes/wiki/REST-API</a>
<b>Source Code</b>	<a href="https://github.com/universAAL">https://github.com/universAAL</a>
<b>Tutorial and Sample code</b>	<a href="https://github.com/universAAL/platform/wiki#core-tutorials">https://github.com/universAAL/platform/wiki#core-tutorials</a> <a href="https://github.com/universAAL/platform/wiki&gt;Hello-World">https://github.com/universAAL/platform/wiki&gt;Hello-World</a> <a href="https://github.com/universAAL/samples">https://github.com/universAAL/samples</a>

#### 4.2.1.1.1 universAAL documentation and WIKI

universAAL IOT is an open source platform that enables seamless interoperability of devices, services and applications on an unprecedented scale. The platform provides the framework for communication, connectivity and compatibility between otherwise disparate products, services and devices.

The universAALwiki (<https://github.com/universAAL/platform/wiki>) provides information about:

- Introduction
- Core Tutorials
- Advanced Topics & Managers
- Community

These sections are described in detail below.

#### Introduction

This section contains introductory information to developers that have just started using universAAL. Specifically it provides a detailed explanation of what universAAL is and describes briefly some of the basic concepts of the platform. Moreover, it provides

information related to the middleware of the platform and its layers and explains the ontological model used by the platform. Finally, it provides details related to the development environment setup and examples of how to create simple universAAL-based applications

### Core Tutorials

The Core Tutorials section provides an in-detail explanation of the very basic concepts of universAAL. In particular, it contains the following sub-sections:

- Advanced topics related to ontologies and an overview about the implementation of OWL in universAAL. Example of how to create a new ontology.
- Introduction to the Context Bus and how it works. Information about context publishers, context events and context subscribers, and examples of restrictions and context event patterns.
- Introduction to the Service Bus and details related to its ontological model. Explanation of the API and how to create service profiles and service requests. Information about the matchmaking system used by universAAL and examples.
- Introduction to the UI Bus and presentation of its components and concepts. In detail example of how to create a universAAL-based UI.
- Example that demonstrates all the aforementioned topics of this section

### Advanced Topics and Managers

Managers are parts of the platform necessary for its proper operation, or provide relevant basic services or events for other applications. All universAAL managers are described and explained in details. Specifically, the managers described are:

- Remote Gateway - A Gateway Manager that connects nodes in different networks so they belong to the same uSpace, or connects different independent uSpaces to a centralized uSpace in a server. This enables the deployment of cloud-based solutions by introducing multi-tenancy support.
- Remote API - The Remote API allows to access an instance of universAAL running in a server, by calling the basic functionality of the buses through a HTTP-accessible API.
- REST API - allows to access an instance of universAAL running in a server, by calling the basic functionality of the buses through a fully RESTful API.
- Context History Entrepot - the Manager in charge of the persistent storage of Context information and history..
- Situation Reasoner - a generic purpose reasoner (gets some basic context information and elaborates new information out of it). Applications can use it to set up their own reasoning rules.
- Profiling and Space Servers - the Profiling Server is a Manager that helps applications deal with profile-related information stored in the Context History Entrepot, including users, their information and their profiles. The Space Server is the same but deals with information about the uSpace and its environment rather than the user.
- Space Orchestrator - provides a scripting language to interact with the buses.
- This section also provides information related to security and data protection, development tools and container functionalities, e.g. installing and running the platform in different container.

## Community

Provides access several repositories that contain the code of universAAL for contribution, code examples and other created universAAL-based applications. Moreover, there is a youtube channel ([https://www.youtube.com/channel/UCIKF3tT\\_P4dz3\\_DmcpFjNoQ](https://www.youtube.com/channel/UCIKF3tT_P4dz3_DmcpFjNoQ)) that contains video tutorials related to universAAL core topics and the creation of universAAL-based applications.

### 4.2.1.1.2 universAAL API doc and swagger

The main API for universAAL is a Java based mechanism that allows the creation of universAAL-based applications. The provided API doc describes in detail all the packages and Java classes of the API (<http://universaal.github.io/platform/apidocs/index.html>).

universAAL also supports a REST API mechanism that allows to access an instance of universAAL running in a server, by calling the basic functionality of the buses through a fully RESTful API. In the related wiki page (<https://github.com/universAAL/remote/wiki/REST-API>) details related to the installation and configuration of the REST API Manager are provided.

The REST API works by handling the basic uAAL-based resources: Spaces representing tenants can hold Context Subscribers, Context Publishers, Service Callees and Service Callers. Subscribers and Callees can receive Context Events and Service Calls respectively, and send them to the client through a callback. Publishers and Callers can be used to post Context Events and Service Requests to the server.

Table 2: universAAL API

URL <sup>4</sup>	METHOD	INPUT	OUTPUT
{url}/uaal	GET		{url}/uaal/spaces
{url}/uaal/spaces	GET		
	POST	Space (json/xml)	Space (json/xml list)
{url}/uaal/spaces/{myspace}	GET		{url}/uaal/spaces/{myspace}/context {url}/uaal/spaces/{myspace}/service
	PUT	Space (json/xml)	
	DELETE		
{url}/uaal/spaces/{myspace}/context	GET		{url}/uaal/spaces/{myspace}/context/publishers {url}/uaal/spaces/{myspace}/context/subscribers
{url}/uaal/spaces/{myspace}/context/publishers	GET		Publisher (json/xml list)
	POST	Publisher (json/xml)	
{url}/uaal/spaces/{myspace}/context/pub	GET		Publisher (json/xml)

<sup>4</sup> The {url} currently defaults to the host address, port 9000.

publishers/{mypub}	POST	ContextEvent (text/plain Turtle)	
	PUT	Publisher (json/xml)	
	DELETE		
{url}/uaal/spaces/{myspace}/context/subscribers	GET		Subscriber (json/xml list)
	POST	Subscriber (json/xml)	
{url}/uaal/spaces/{myspace}/context/subscribers/{mysub}	GET		Subscriber (json/xml)
	PUT	Subscriber (json/xml)	
	DELETE		
{url}/uaal/spaces/{myspace}/service	GET		{url}/uaal/spaces/{myspace}/service/callers {url}/uaal/spaces/{myspace}/service/callees
{url}/uaal/spaces/{myspace}/service/callers	GET		Caller (json/xml list)
	POST	Caller (json/xml)	
{url}/uaal/spaces/{myspace}/service/callers/{mycer}	GET		Caller (json/xml)
	POST	ServiceRequest (text/plain Turtle)	ServiceResponse (text/plain Turtle)
	PUT	Caller (json/xml)	
	DELETE		
{url}/uaal/spaces/{myspace}/service/callees	GET		Callee (json/xml list)
	POST	Callee (json/xml)	
{url}/uaal/spaces/{myspace}/service/callees/{mycee}	GET		Callee (json/xml)
	POST	ServiceResponse (text/plain Turtle)	
	PUT	Callee (json/xml)	
	DELETE		

Examples of the needed body requests and how to use the provided the REST-API are provided also in the wiki page of universAAL.

### 4.2.1.1.3 universAAL source code public access

UniversAAL is an open source platform (available under Apache License 2.0) and everyone is free to use, copy, modify and redistribute the platform sources (within the limits of the license). Master repositories for the source code, as well as wikis (main source of documentation), and issue management are all currently hosted in GitHub Repositories (<https://github.com/universAAL>).

### 4.2.1.1.4 universAAL tutorial and sample source code

universAAL provides several development tools. In particular, there is an eclipse plugin that help developers use universAAL (<https://github.com/universAAL/tools.eclipse-plugins>) and a runtime plugin (<https://github.com/universAAL/tools.runtime>). Also, an Apache Karaf distribution ready configured to run universAAL platform and applications (<https://github.com/universAAL/distro.karaf>) and a Pax Runner configuration for universAAL are provided (<https://github.com/universAAL/distro.pax>).

universAAL's wiki contains detailed information about how to begin creating your own applications. It starts with a typical "Hello World" example providing also information about how to run it (<https://github.com/universAAL/platform/wiki/Hello-World>).

Moreover, it provides (<https://github.com/universAAL/platform/wiki/Running-the-lighting-sample>) a more realistic example that is related to the manipulation of lighting bulbs and provides (<https://github.com/universAAL/platform/wiki/Lighting-Sample-Walkthrough>) also an analytical walkthrough of the code.

A youtube channel ([https://www.youtube.com/channel/UCIKF3tT\\_P4dz3\\_DmcpFjNoQ](https://www.youtube.com/channel/UCIKF3tT_P4dz3_DmcpFjNoQ)) contains video tutorials related to universAAL core topics and the creation of universAAL-based applications.

Finally, there is also a repository (<https://github.com/universAAL/samples>) that contains several complete samples, partially with GUI, as example for the development of new applications.

## 4.2.1.2 Useful tools in the context of ACTIVAGE

The universAAL IoT platform offers a full suite of tools which help developers in the whole development life cycle, from conception to testing. Within this suite there are some tools which have some interest in the context of ACTIVAGE development tools, and their conceptualization.

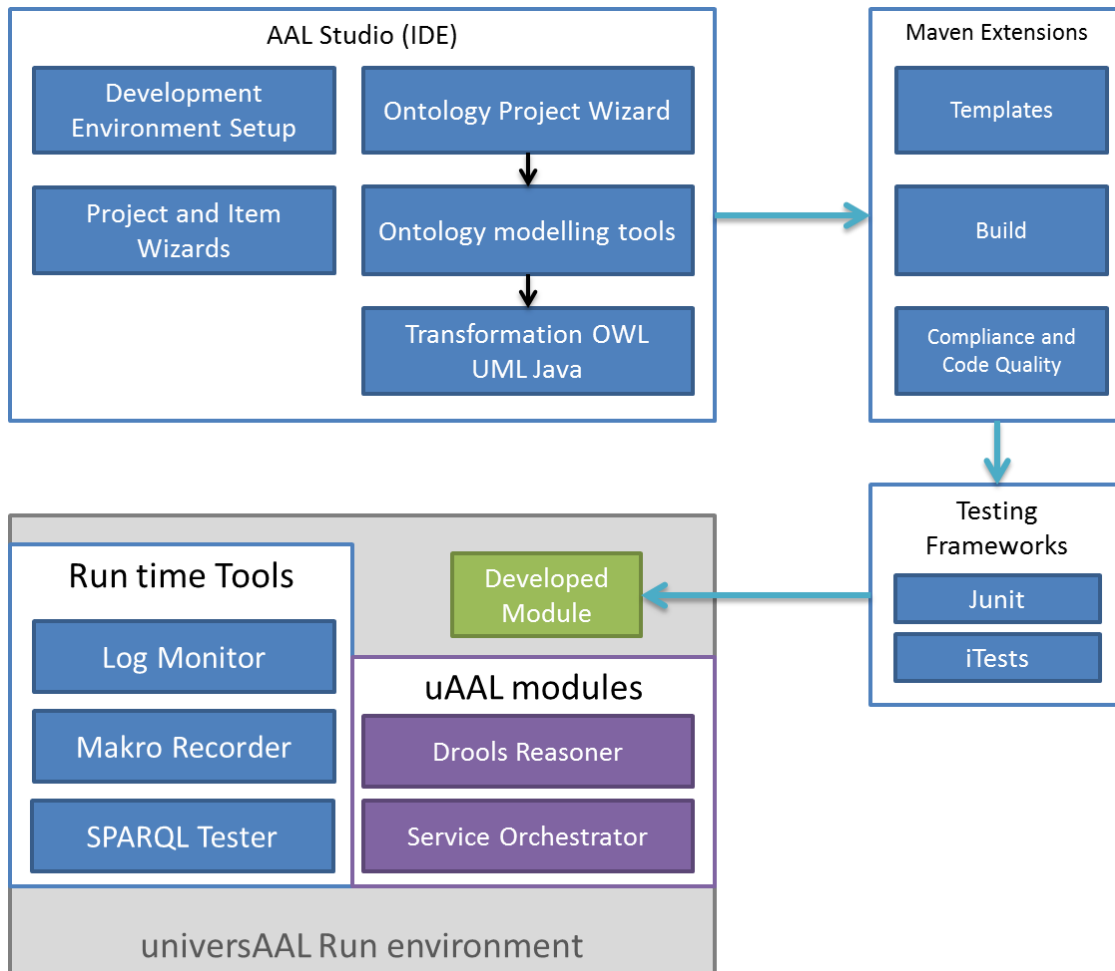


Figure 39 universAAL IoT tool set (Blue), modules (Violet), and workflow which can be interesting to ACTIVAGE.

Figure 39 Shows the most interesting tools for conceptualizing ACTIVAGE tools, as well as their context (IDE, run environment, or neither), and the role they play in the general development workflow for a specific module (in green).

IDE tools (4.2.1.2.1) are useful to create source code, which is then compiled using Maven (4.2.1.2.2), this allows developers to use IDEs other than the one proposed. Within the build process, the developer can choose to create tests using testing frameworks (4.2.1.2.3), which have been extended to make it easier to develop said tests. Within the runtime environment there are 2 kinds of tools, proper tools (4.2.1.2.4) which are used to test, monitor and produce data of existing applications; and modules of the platform it self (4.2.1.2.5), which can be extended through specific files to create or extend applications.

#### 4.2.1.2.1 AAL Studio

The AAL Studio is a suite which provides an integrated development environment (IDE) based on Eclipse for building applications and components using the universAAL execution platform. The AAL Studio makes it easier to get started with the AAL application development, and will make some of the development tasks more efficient. Also, it gives easier access to the resources needed by the developer such as documentation and samples.

The functionality of the AAL Studio is provided by the individual plugins that are installed within it, including wizards for creating projects; build tools for simplifying building and launching of applications, and modeling and transformation tools for making the development more efficient. The AAL Studio tools created by universAAL are implemented as Eclipse plug-ins.

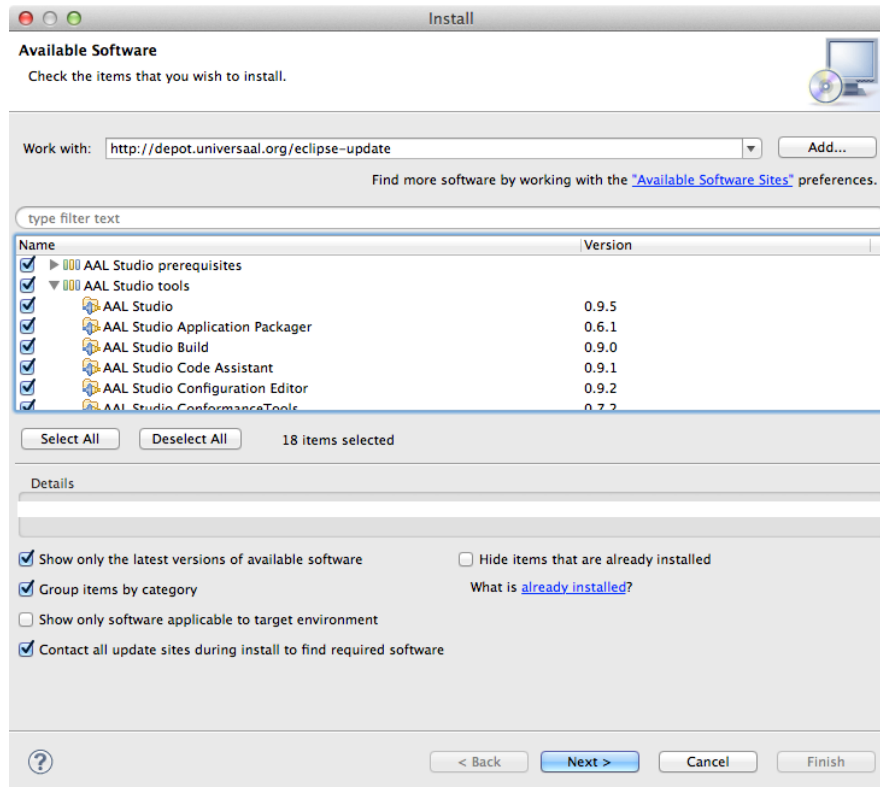


Figure 40 Installation process of AAL Studio suite in Eclipse

More info about the tool might be found here: <https://github.com/universAAL/tools.eclipse-plugins/wiki/AAL-Studio-overview-and-installation>

The AAL Studio is an IDE as described in Section 4.1.4, providing functions such as: component selection, functionality selection, code generation, Server-client Application source templates, and source translation.

The concept of SDK integrated in the IDE might be interesting for ACTIVAGE. Some of the plugins could be recycled into ACTIVAGE, but there are several problems with this. Not every ACTIVAGE project is a java project (ergo it would be difficult to force non-java developers to use Eclipse, a primarily Java IDE). Second, most of the plugins are very specific to universAAL development.

#### 4.2.1.2.1.1 Development Environment Setup

This tool aims at simplifying the setup of the development environment. It is shown automatically for every new Eclipse workspace and takes care of:

- settings for maven
- downloading source code
- importing source code into Eclipse



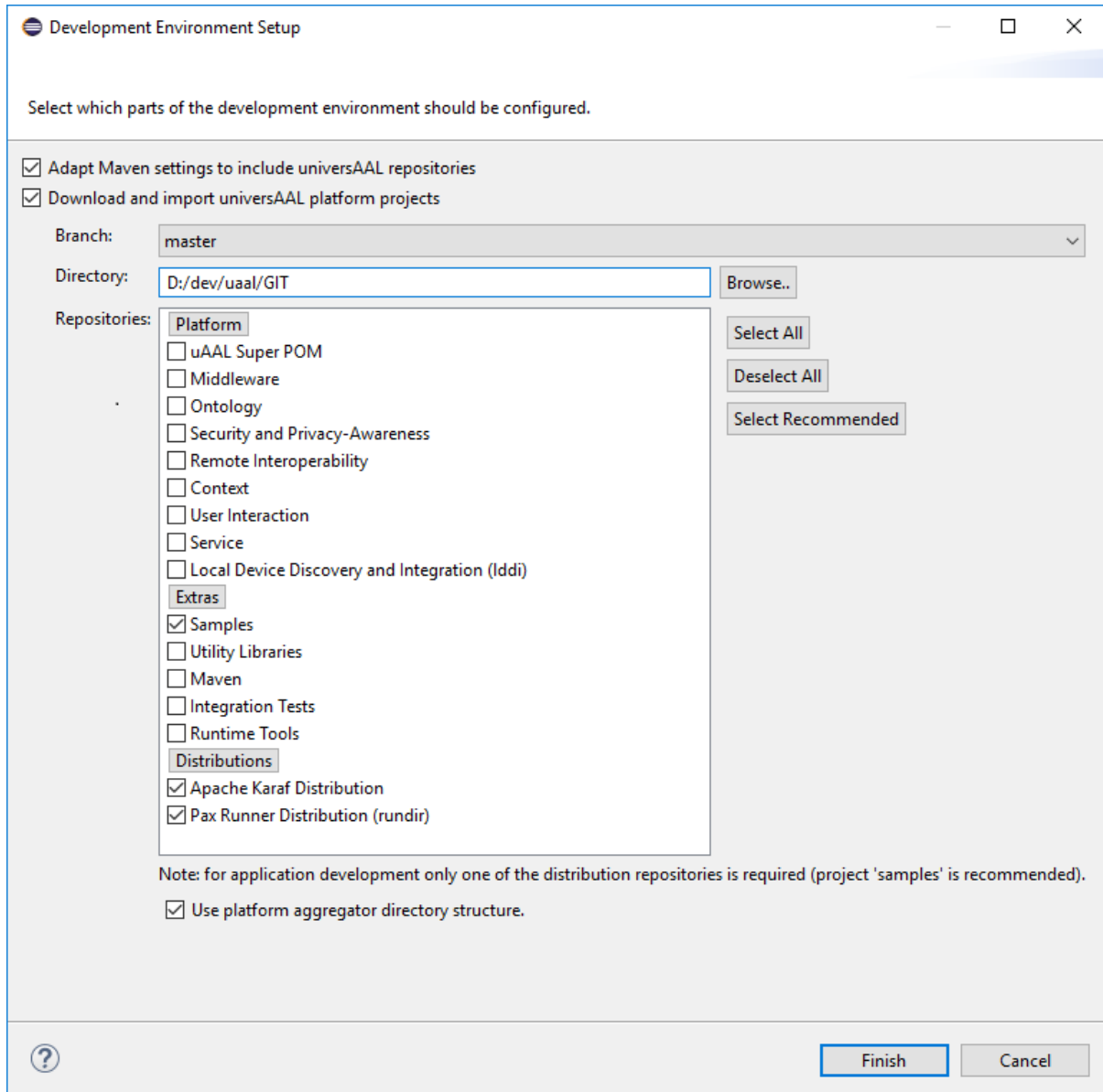


Figure 41 Source download interface

#### 4.2.1.2.1.2 Project and Item Wizards

This AAL Studio tool is intended to be used by developers of services and platform components. It makes it easy to create new universAAL-compliant projects by providing a skeleton project with all the files you need and initial content to make the project work in universAAL. The item wizards generate new files required or optional to a universAAL project with the proper API and protocol usage. It reduces the time of development since without this tool an unexperienced developer would need to review and copy samples. By using the wizards it is assured to have well-formatted files and project structures. As the developer’s experience grows the need for this module decreases, and most of it can be replaced by other tools such as the maven templates (see 4.2.1.2.2.3).

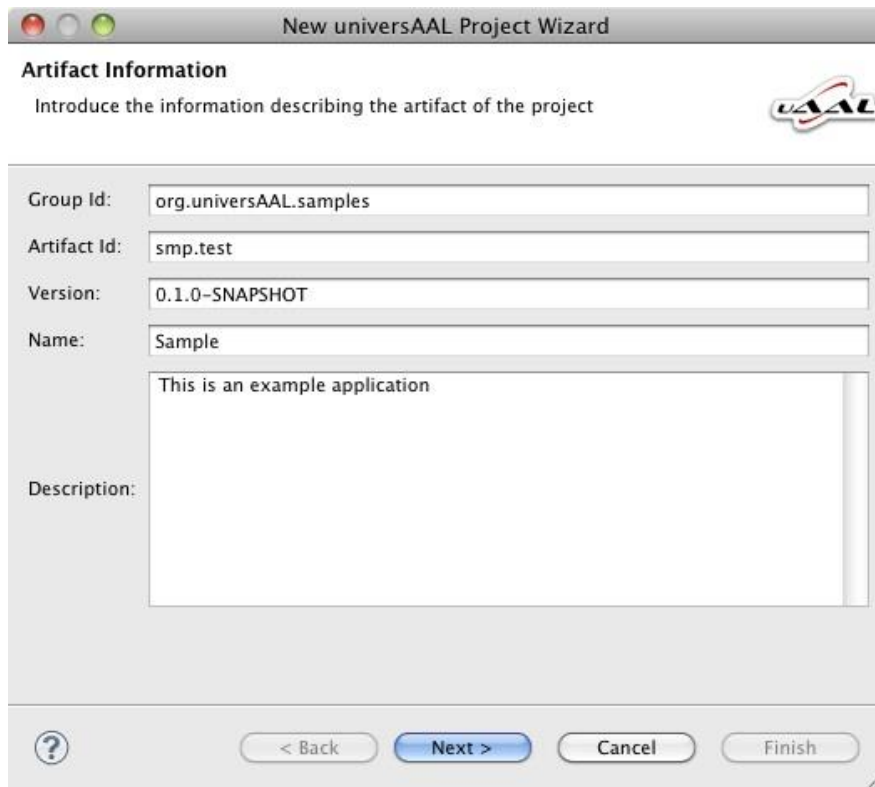


Figure 42 Basic information provided for the Project wizard

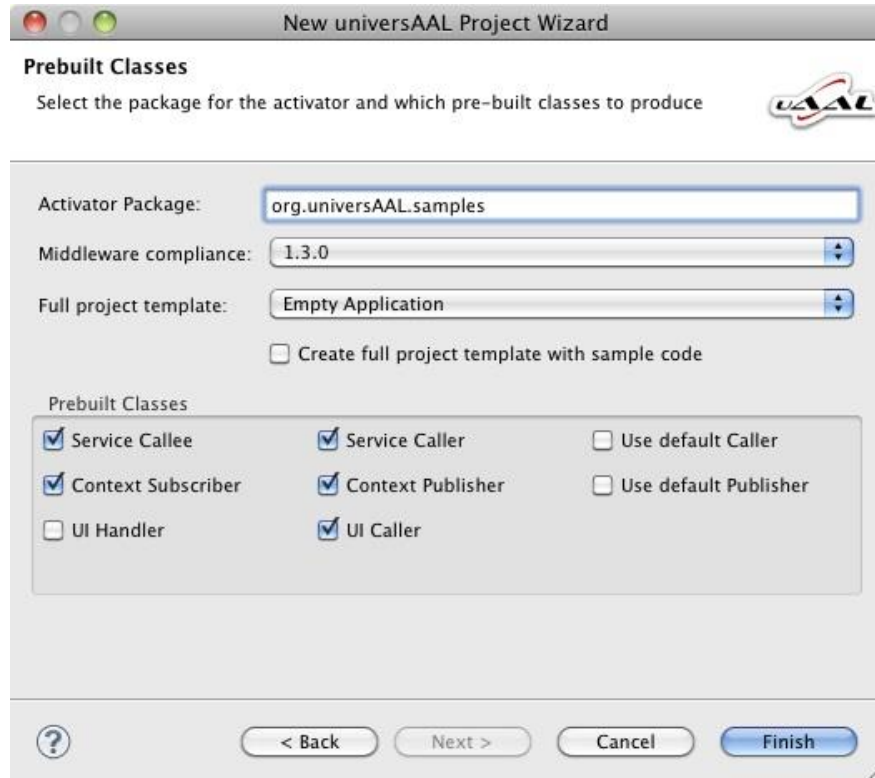


Figure 43 Customization of the project dependencies and components in the project wizard.

### 4.2.1.2.1.3 Ontology Project Wizard

The Ontology Project Wizard makes it easy for a developer to get started developing ontologies, by setting up a valid Eclipse project containing a UML model with the correct structure and template content typically used for ontologies in universAAL. The wizard lets the developer enter the name and package / name-space information for the ontology, and uses these values to set up both the initial content of the UML model and the Maven project file. This wizard is similar to the Project and Item Wizards in setting up a Maven project for universAAL, but the content created in the project is specifically for modelling of ontologies.

Figure 44 Basic information for the Ontology project wizard.

#### Ontology modelling tools

The Ontology Modelling Tool (OMT) provides a simple user interface that enables ontology developers to focus on the ontology concepts instead of the java representation of it. This tool provides the following benefits:

- Simplify the process of creating ontologies for use on universAAL
- Lower learning threshold
- Reduce effort required (time)
- Limit error-prone activities

- Reuse in universAAL and for other platforms (representations)

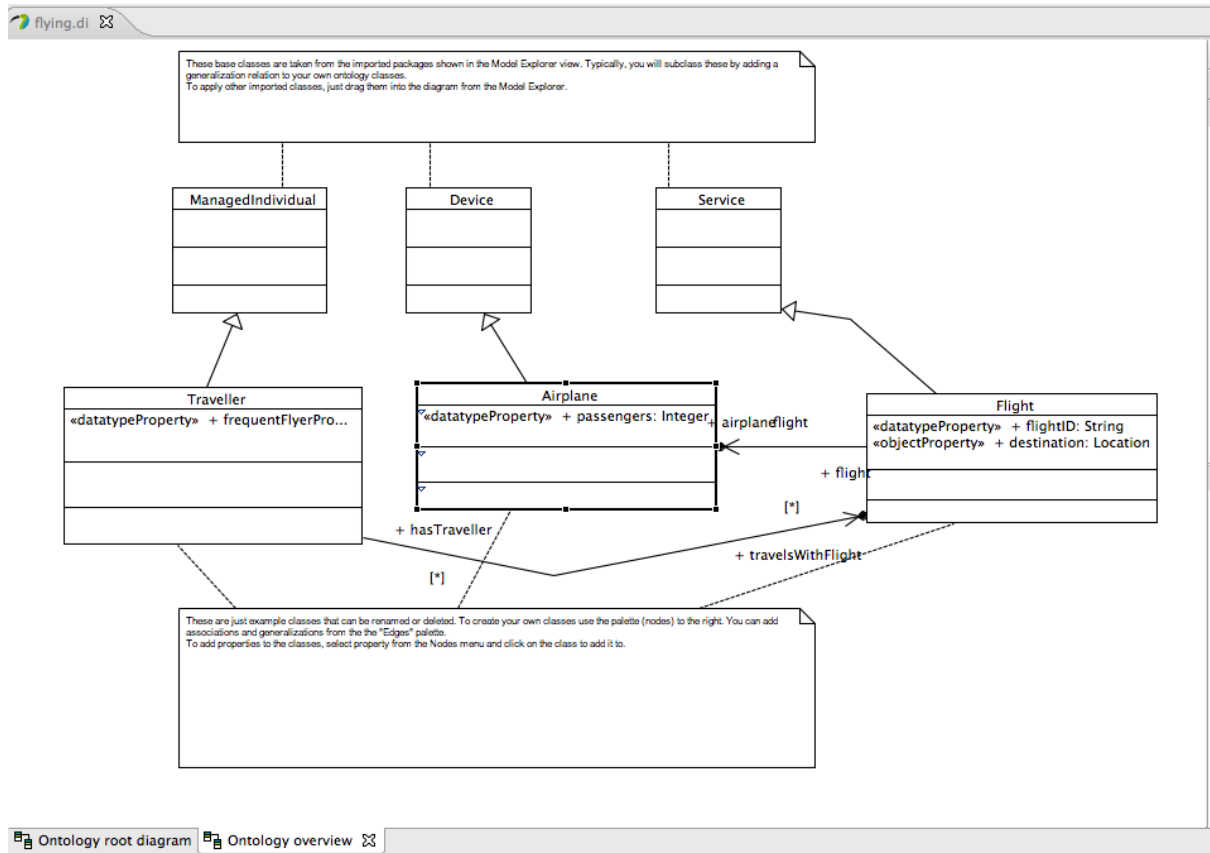


Figure 45 view of recently created project.

To add more concepts and properties, chose elements from the palette.

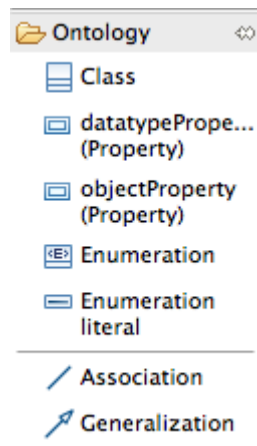


Figure 46 tooltips to create ontologies graphically.

Although the tool is used to model ontologies, it has many limitations, for example it cannot create complex restrictions, or import ontologies other than the default ones already included. These type of designs require the developer to review the generated code (see

next tool), and introduce them in the code. Because of these shortcomings the proposed AIOTES tool will be based on proper ontology editors such as Protégé.

A possible extension of this tool would be to allow it to generate UML diagrams which can later be exported to OWL independent of universAAL ontologies. The only advantage this has is the simplicity of the tool and it being available within the Eclipse IDE; but because of all the limitations it has it is better to reuse other ontology design tools.

#### 4.2.1.2.1.4 Transformation OWL UML Java

The role of the Model Transformation Tool is to be a common component for all model transformations in AAL Studio. Currently only one transformation is available on UML files, namely the Ontology to Java transformation. The purpose of this AAL Studio tool is to make it easy to generate implementation resources (such as Java source code) from models created in UML or EMF, and to give a good integration with the Eclipse development work.

Currently, the primary example of this the transformation that generates full Java source code and Maven POM files from UML ontology models created using the Ontology Project Wizard and modelled using our Ontology Modelling approach. The main benefit of using this tool is that it automates (part of) the implementation work. The alternative for the user would usually be to hand-code the implementation based on the model or some equivalent design.

The modelling and transformation approach also have the benefit that maintenance is simplified. E.g., if there are changes in the target platform (e.g. the Java representation of the ontologies), then correct code can quickly be re-generated once the transformations have been updated.

When the ontology modelling tool is replaced by a proficient ontology design tool, the output is always an OWL file. AIOTES will provide a tool which instead of transforming UML to java, it will transform directly OWL files to Java, superseding this tool. This not only has the benefit of using professional ontology design software, it can also be applied to existing ontologies. Additionally the Template based Ontology code generation tool (see 4.2.1.2.6.1) will be capable of not only generating universAAL code, but by switching the template system, generate code for other platforms, frameworks, or programming languages.

#### 4.2.1.2.2 Maven extensions

The universAAL platform is built thanks to the Maven project management system. The universAAL community has produced a series of tools to make more optimized use of this framework. As such there are 3 main categories of tools: Compliance, Build, and Templates. The functions that these tools offer are mainly in the realm of project management, but in the case of the templates they offer the source code templates functions very efficiently.

##### 4.2.1.2.2.1 Compliance and code quality

Since the beginning universAAL had very strict quality assurance rules, and many conventions. Thus there was a need for Platform developers to provide convenient automatic reports on these. Maven already provides many Compliance and code quality tools. But when it comes to the universAAL specific conventions a new set of tools needed to be developed.

A specific universAAL convention is the release policy, all universAAL components are synchronized to the same version before releasing, along with other checks these tools help smooth the release process. Although this tool is obviously very specific for platform development it still might be useful for any multi-module application using Maven.

#### 4.2.1.2.2.2 Build

These tools are used to setup each building environment for proper testing by particular testing framework (see Section 4.2.1.2.3); they also generate parallel files used for deploying, such as providing the start up sequence for running any particular module. These are known as composite files used in the pax runner (see 5.2.2.1.2 and 4.2.1.2.3).

These tools are quite generic, but only useful for software modules using Maven, and OSGi framework, in particular the Pax Runner OSGi provider. Thus it may not be useful for AITOTES. Yet the concept of a tool that automatically generates the required runtime configuration given a module might be interesting in AITOTES.

#### 4.2.1.2.2.3 Templates

The Maven framework provides many plugins, one of which, the archetype plugin, provides a robust and flexible framework for creating projects out of templates, as well as custom templates.

In particular universAAL provides a set of custom templates to create universAAL projects and runners. This is an alternative way to provide the function of source code template as offered by the AAL Studio (see 4.2.1.2.1).

#### 4.2.1.2.3 Testing frameworks

The universAAL platform is very keen on testing, particularly automatic testing, as such it offers specific APIs and frameworks for unit and integration testing which are automatically recognised by Maven as well as Eclipse.

The unit testing framework and API works on top of the Junit framework, has the capability of automatically building a run environment with the full stack of universAAL middleware for testing particular operations. There are different levels of the provided stack, which may be used according to if the developer is implementing tests for the middleware, or application; the later even setting up all the required ontologies for the test.

The integration testing framework and API allows the developer to set up a full OSGi stack, allowing also to introduce particular modules with the functions of testing a monitoring the results. It is much slower than the unit testing, but provides more trustworthy results as the run environment is almost exactly the same as the final deployed environment.

These frameworks provide the code quality function for the implementation phase. They are very universAAL specific; although the integration testing could be abstracted for OSGi applications. Yet the concept of a tool to easily set up testing could be very interesting to be included in AITOTES tool set.

#### 4.2.1.2.4 Runtime tools

universAAL provides a set of runtime tools. By definition these tools are meant to run in an universAAL Runtime environment, thus they are properly deployment tools; but in many cases the tools are also very useful for developers when testing. In this section we will overview these tools from the development perspective, for a full view of the tools please refer to the Section 5.2.2.1.2.

For more information about runtime tools find the full documentation in: <https://github.com/universAAL/tools.runtime/wiki>.

The concept of tools which run on the platform it self is already included in ACTIVAGE. These tools are specific for universAAL but some may be interesting to have in AITOTES. For example a visual log to easily identify what is going on in the framework; a makro recorder to compose services “manually”; or a query inspector to check the data in the system.

#### 4.2.1.2.4.1 Log Monitor

The log monitor is a graphical viewer of all the universAAL events; it helps debug applications by showing the internal logic behind the decisions for matchmaking services and events, as well as showing graphical representations of the exchanged messages between modules; showing them in a real life sequence message diagram.

This tool covers the functionality of data analyser, in fact it is analysing the metadata produced by the system itself and reporting on it.

This tool is specific for universAAL, and it may require a lot of effort to generalize for AIOTES. Yet it is possible to adapt it to the specific interfaces, such as the management interface of AIOTES, to adapt part of the functionality, such as the live view of events, semantic graphic visualizer and sequence diagram.

#### 4.2.1.2.4.2 Makro Recorder

The Makro recorder is useful to create sequence of events by recording the events and services generated in the devices in the deployed space, and being able to later replay them. The main purpose of the Makro recorder is to enable deployers and even end users, to set up rules and automatic responses. But for developers it may be useful to create sequences using a specific set of devices; and then generalize the result for any deployment.

This tool partially provides the functionality of service composition, as it complies for specific deployment and not for generic deployments.

Being a tool running in universAAL it may be possible to add this tool directly to the SDK of AIOTES through the universAAL bridge.

#### 4.2.1.2.4.3 Sparql tester

The sparql tester is a tool which enables developers, and deployers, to issue sparql queries to the data lake in the space.

This tool offers the function of Data model work bench, Metadata storage explorer and Data manipulator.

The tool is specific for the universAAL module implementing the sparql interface. Thus it is possible to directly use this tool through the universAAL AIOTES bridge, if the particular storage implements this interface. Yet the tool is extremely simple, and for desktop use only; thus a web interface providing the same (or more functionality) is probably a better option for AIOTES.

### 4.2.1.2.5 universAAL modules as tools

universAAL offers a series of modules which can be considered as tools since they allow to execute code from different sources, essentially providing alternative APIs and frameworks for universAAL.

Worth mentioning in this category are the Drools Reasoner<sup>5</sup>, and the Service Orchestrator<sup>6</sup>. Both provide the development function of service composition, the main difference is the way they do it, one uses Drools rule production engine (a java based if-then execution), and the second uses javascript (exporting universAAL API to javascript).

---

<sup>5</sup> <https://github.com/universAAL/context/wiki>

<sup>6</sup> <https://github.com/universAAL/service/wiki/Service-Orchestration>

Both tools can be imported to AIOTES directly through the use of the universAAL AIOTES bridge. Adding simple text service composition and rule based service definition are powerful methods to enable quick service development.

### 4.2.1.2.6 Planned Tools

#### 4.2.1.2.6.1 Template based Ontology code generation

A new tool development is planned ; this tool is not exclusively a universAAL tool, but an effort to expand the flexibility of the code-generation plugin of Protégé. The current implementation of this plug-in enables the generation of Java code, based on the OWL API<sup>7</sup> framework, from an OWL ontology. The code generation is done with static code, and simple text substitutions. It also has a maven flavour, to generate code directly from an OWL file in a maven lifecycle.

The tool to be implemented will use the Apache Velocity macro template system, which is robust, fast and easy to use and it will be based on the development of XML template coordination. It will allow pluggable template systems (either by attaching an OSGi bundle fragment to the code generation plugin bundle or by loading them from the internet). This way, communities will be able to create their own template systems and contribute them directly to the Protégé project. They can create template systems to create code specific for their platforms, create ontology based APIs, as well as generic needs. For example an SQL code generator template could transform any ontology to the series of SQL commands to create the appropriate data tables, views and populate it with the instances according to the ontology.

It will also support maven variant. This way, it will be possible to use the same code but launched from within a maven project, and include it in the build process.

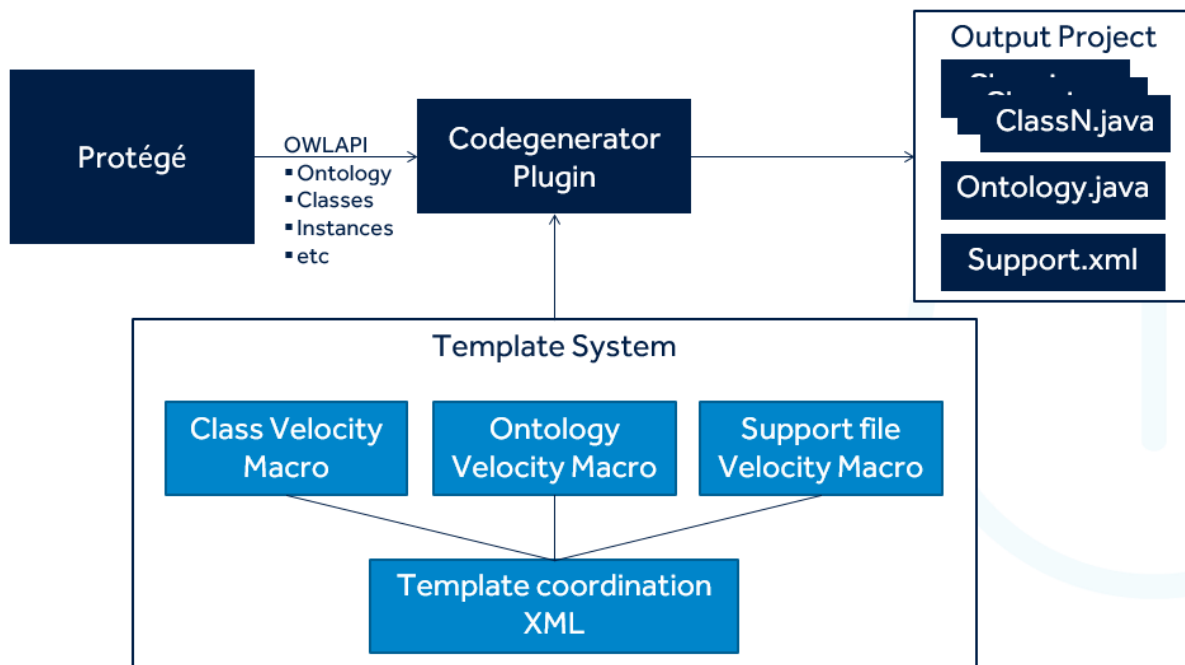


Figure 47 a code-generation Java generation example

<sup>7</sup> <https://owlcs.github.io/owlapi/>



As an extended use of the tool, templates can be development with specific Ontology in mind e.g. given an ontology for Security, application developers can create extensions for it and the output project would be the security component for the application.

This tool currently is indesign phase, it has a lot of potential. It (along with key pluggable template systems) could be integrated in AIOTES tool suite directly, for example to provide, or extend, non-semantic platforms with quick and easy interfaces to the AIOTES semantics, or to provide the means to easily allow simple application development by non-technical developers from an ontology (or through ontology based specific tools).

This tool, in combination with protégé it self, could provide support for the functions for many of the Semantic Interoperability Layer as well as Data analytics, and, of course, code generation.

#### 4.2.1.2.6.2 universAAL Control Center 2.0

In an effort to make running tools (4.2.1.2) more coherent a similar approach to the AAL Studio (4.2.1.2.1) will be followed. The result is a graphical framework consisting on many plugins that provide different functionality for the deployer. This tool will also be useful for developers. See the full description in Section 5.2.2.3.

### 4.2.1.3 Mapping between universAAL and ACTIVAGE development tools

Given the large amount of tools, some can be used directly for ACTIVAGE, in most cases the concept they embody can be adapted to AIOTES tool, in other cases the tools are too universAAL specific for any use in AIOTES.

Table 3: universAAL Tools Mappings with Activage development tools

universAAL development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
<a href="#">Support tools</a> (documentation, wiki, source code samples)	<a href="#">Support</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		The universAAL support tools (documentation, wiki, code samples, etc.) will be used as part of the overall AIOTES support material.	
Project and Item wizard	IDE / <a href="#">Source code templates</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
			These tools are too universAAL specific (they relate to universAAL projects and modules). A Similar wizard could be useful for AIOTES development.
Ontology project wizard, modelling tool, and transformation.	IDE / Code Generator	Yes	No
		<b>How?</b>	<b>Why?</b>
			These tools are too universAAL specific (they relate to universAAL ontology development). This tool would only make sense if AIOTES requires a project around an

			ontology. A generic tool for designing tools and exporting code is porposed, see 4.2.1.2.6.1
Log Monitor	Not mapped with AIOTES		
Makro Recorder	Not mapped with AIOTES		
Sparql Tester	Data Lake / ACTIVAGE data model Workbench	Yes <b>How?</b>	No <b>Why?</b> This tool is used primarily to provide an accessible interface for developers to the SparQL database backend, and test or organize database operations. This functionality is expected to be included in ACTIVAGE data model Workbench.
Compliance and Code Quality	Not mapped with AIOTES		
Maven build extensions	Not Mapped with AIOTES		
Maven Achetypes	IDE / <a href="#">Source code templates</a>	Yes <b>How?</b> These tools can be used to generate universAAL projects, but also karaf features, specially indicated for OSGi based deployments.	No <b>Why?</b>
Testing frameworks	Not Mapped with AIOTES		
Reasoner and orchestrator	IDE / Service Composer	Yes <b>How?</b> The universAAL Modules can be used through the SIL to execute their composition operations.	No <b>Why?</b>
Template Based Ontology Code Geneartor	Semantic Interoperability tools; IDE / Code Generator	Yes <b>How?</b> The tool will have the capability of defining its own templates, specific tempaltes for platform specific adaptations (given an arbitary ontology) can de defined. It can also be used to compose other tools which generate code or templates. Lastly the generic templates (e.g. SQL from OWL) can be used for adating applications to AIOTES Ontologies.	No <b>Why?</b>
universAAL Control Center	Semantic Interoperability Layer tools / * ; Visual Analytics tools / *	Yes <b>How?</b>	No <b>Why?</b> uCC is too universAAL Specific. The concept of an In-platform running tool for monitoring the

deployment can be useful for ACTIVAGE.

## 4.2.2 SOFIA2

SOFIA2 is a middleware that allows the interoperability of multiple systems and devices, offering a semantic platform to make real world information available to smart applications (Internet of Things).

It is multi-language and multi-protocol, enabling the interconnection of heterogeneous devices. It provides publishing and subscription mechanisms, facilitating the orchestration of sensors and actuators in order to monitor and act on the environment.

Cross-platform and multi-device through its SDK, APIs and extension mechanisms that allow integration with any device.

### 4.2.2.1 Documentation and WIKI

SOFIA2 has available in site ([http://sofia2.com/desarrollador\\_en.html#documentacion](http://sofia2.com/desarrollador_en.html#documentacion)) several documentations, it is proving information about:

- SOFIA2 Basic information
- SOFIA2 User level.
- SOFIA2 Developer
- SOFIA2 Advanced developer.

In addition provides more news, information, help and guides in its Blog : <https://about.sofia2.com/>

### 4.2.2.2 APIs and libraries

SOFIA2 offers several communication protocols between KP (device) and SIB(interface of communication whit SOFIA2):

- MQTT (Message Queue Telemetry Transport) is a connectivity protocol focusing in M2M (machine-to-machine) and IoT (Internet of Things). It is a lightweight messaging protocol based on TCP and especially designed for remote devices with little memory and little processing power. It is based in a publish/subscribe messaging model that eases one-to-many distribution.
- Restful: the deployment of SOFIA2 in sofia2.com provides a RESTful Gateway to invoke operations on that instance. This Gateway works around SSAPResource, which represents, along with the Gateway HTTP verbs, the different SSAP operations.
- AJAX (Asynchronous JavaScript And XML) is a web development technique to create interactive applications or (Rich Internet Applications). Those applications run in the client, meaning in the user's browser, while at the same time an asynchronous communication with the server is kept on the background.
- WebSocket is a technology that provides a bidirectional communication channel and full duplex on a single TCP socket. It is designed to be implemented in browsers and web servers, but can be used by any client/server application. The use of this technology provides similar functionality to opening multiple connections in different ports, but multiplexing different WebSocket services over a single TCP port (at the cost of a small protocol overhead) .

In addition, SOFIA2 provides several APIs to connect in easy way different devices using several technologies:

- JAVA API: SOFIA2 provides a Java API to develop KPs. This API is made of a set of JAVA classes and interfaces that ease the generation and maintenance of SSAP message, as well as the connection with the Platform using connectors that communicate with the Platform's gateways.
- ANDROID API: The Android application development is created in Java programming language and a set of development tools called Android SDK. SOFIA2 provides a Java API for developing KPs on the Android SDK. This API consists of a set of Java classes and interfaces that facilitate the generation and process of SSAP messages as well as the connection to the platform through connectors that communicate with the platform's gateways
- ARDUINO API: This API provides the Platform for Arduino devices. It includes operations to interoperate with the SIB: connect, send and receive messages from the SIB.
- JAVASCRIPT API: The Platform also provides an API to interoperate with the SIB using JavaScript. The JavaScript API provides a set of functions covering all the SSAP operations, thus abstracting the programmer from the message building process.
- NODE JS API: Node.js (<http://nodejs.org/>) is a platform that enables the development of JavaScript on the server side through V8 JavaScript engine developed by Google. Its architecture is eventbased and designed for asynchronous programming. It consists of several modules that facilitate the use of this language. SOFIA2 provides a Node.js API for the development of KPs. This API is a set of utilities that facilitate the generation and processing of SSAP messages and the connection with the platform through MQTT to communicate with the platform's gateways.
- C API: It provides a dynamic link library for the development of KPs using the C language. For compatibility among platforms, the source code of the library is provided along with a Makefile to compile it.

### 4.2.2.3 Source code public access

SOFIA2 has created a Git-Hub repository where to find all the open source code to apply to your field.

Source code of SOFIA2 is accessible through the <https://github.com/Sofia2>

### 4.2.2.4 SOFIA2 tutorial and sample source code

SOFIA 2 offers tutorial in YouTube Channel in the form of videos that show how to use the platform: [www.youtube.com/channel/UC6VGV\\_IN9gB2mJcPdIYHB0A](http://www.youtube.com/channel/UC6VGV_IN9gB2mJcPdIYHB0A).

In addition, SOFIA2 has a tutorial of first steps in SOFIA2:

<https://www.iorad.com/player/19843/Primeros-Pasos-Sofia2>

In relation with samples and source code, there are several demonstrators to show example of how easy is to work with Sofia2. All examples are available with its source code. The next table does a summary of them.

Table 4: SOFIA2 existing samples and source code

Example	DESCRIPTION
<b>Geographic viewer</b>	This demonstrator shows geographically public info managed by Sofia2. <a href="https://sofia2.com/gsma_dashboard/Apps/DGSMA.html">https://sofia2.com/gsma_dashboard/Apps/DGSMA.html</a>
<b>OpenData Viewer</b>	OpenData Viewer integrated in the platform. <a href="https://sofia2.com/console/opendata/search#">https://sofia2.com/console/opendata/search#</a>
<b>Dashboard Smart Health</b>	Dashboard example where each patient have a wearable bracelet type capable of measuring information on steps walked, sleep, oximetry, .... <a href="https://sofia2.com/demos/smarthealth/pages/dashboard_phillip.html">https://sofia2.com/demos/smarthealth/pages/dashboard_phillip.html</a>
<b>Geographical search Twitter</b>	This demonstrator allows you to search in a custom zone tweets talking about this topic. <a href="http://sofia2.com/demos/tweets_finder/tweets_finder.html">http://sofia2.com/demos/tweets_finder/tweets_finder.html</a>
<b>Demo Streaming Twitter</b>	This demonstrator shows the capabilities of the platform is to receive real-time information from Twitter and their representation once stored. <a href="http://sofia2.com/Kp_TwitterReglaLexico/">http://sofia2.com/Kp_TwitterReglaLexico/</a>

#### 4.2.2.5 Useful tools in ACTIVAGE context

SOFIA2's Platform has integrated several tools for deployment and development. The current section shows tools for development.

##### Ontologies management tool

This allows for a complete management of the ontologies, including:

- Creating , modifying and delete an ontology and a ontology group.
- Searching an ontology and ontology group following some criteria.
- Finding and subscribing to an ontology and ontology group.
- Subscription to an ontology
- Authorization to one ontology or group of ontologies. The permissions a user can have in relation to ontology are as follows:
  - QUERY: They user can launch queries about ontology insertions performed by the KP"s that the user owner has created.
  - INSERT: The user can make ontology additions for the KP"s that the user owner has created.
  - ALL: The user with this privilege on an ontology has both QUERY and INSERT permissions.

The goal of the ontologies is becoming the schema against which the ontology insertions made by the KP"s will be validated.

### KPs/APPs management tool

This allows to manage the KP"s with which the Platform will interact. This can be managed in several ways:

- Create and modify KP"s.
- Search KP"s.
- See active KP"s.
- Manage the tokens and instances associated to a KP.

### Token management tool

This functionality allows to manage the tokens associated to a KP. All the users extant in Platform can access this functionality, thus a user can manage the tokens on the KP"s that user has the right permissions:

- Users with ADMINISTRADOR role: Can manage the tokens of all the KP"s in the Platform.
- Users with COLABORADOR role and others: Can manage the tokens of all the KP"s that are owned by that user.

### Rules management tool

From this functionality will be allowed management Scripts for the following types of users:

- Users with ADMINISTRADOR role can management all rules created on the platform.
- Users with COLABORADOR role can create CEP Rules, events and Scripts on the ontologies that the users have Insert or All permissions. Users may view and modify onthe Rules that they have create.

### Visualization tool

That tool allows work whit the type of gadget you want to create a screen to choose a KP and name the gadget will appear. In the case of selecting the External HTML, we should introduce the external URL. To save, press "Create" button. These gadgets can be selected to create a Dashboard

### Predefined Queries Management

Itallows predefined queries on ontologies stored in the platform to retrieve instances of those ontologies sent by KPs. You can query in two types of languages: o SQL-Like: If the SQL-like language is used. o Native: If the native language of BDTR (MongoDB) is used. The tool provides a list of predefined queries with the options: View, Edit, Launch or Delete.

In other words, all these tools participate in the deployment and expansion of the platform itself but are not directly related to any other element of the ACTIVAGE Framework. Thereby, It is important to highlight that these tools are PLATFORM SPECIFIC and cannot be generalized to AIOTES.

### Send SSAP Messages tool

we can send SSAP messages to the SIB and simulate a communication between KP"s and SIB from this menu option.

When accessing, you will be shown a page with two text areas. The one in the left must contain the SSAP Message that you want to send to the SIB. The one in the right will contain the synchronous message with the SIB’s response.

### API Manager tool

This section is used to allow the availability of Ontologies as APIs. You can also make the subscription to APIs published and query the API Key generated for invoking them.

This allows platform users to interact SOFIA2 through the REST resources without the need to handle the advanced concepts of SOFIA2 (ontologies, KPs,...), allowing also SOFIA2 to have a catalogue of APIs REST that allows the user to access the information stored to develop their own applications or extend their own in a simple way.

Only users with Administrator and Collaborator role will have access to My APIs option. The other options are available to all users.

## 4.2.2.6 Mapping between SOFIA2 and ACTIVAGE development tools

Part of the SOFIA2 development tools, described through Section 0 can be used within the AIOTES infrastructure as ACTIVAGE development tools (Section 4.1). This mapping is presented in Table 5 where it is specified if a tool can be generalized to be used within AIOTES and how.

Table 5: Mapping between SOFIA2 and ACTIVAGE development tools.

SOFIA2 development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
Ontologies management tool	Not mapped with AIOTES		
KPs/APPs management tool	Not mapped with AIOTES		
Token management tool	Not mapped with AIOTES		
Token management tool	Not mapped with AIOTES		
Rules management tool	Not mapped with AIOTES		
Visualization tool	Not mapped with AIOTES		
Predefined Queries Management	Not mapped with AIOTES		
Send SSAP Messages tool	Not mapped with AIOTES		
API Manager tool	Not mapped with AIOTES		
<a href="#">Support tools</a> (documentation, wiki, source code samples)	<a href="#">Support</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		The SOFIA2 support tools (documentation,	

		wiki, code samples, etc.) will be used as part of the overall AIOTES support material.	
<a href="#">Source code samples</a>	IDE / <a href="#">Code generator</a>	Yes	No
	IDE / <a href="#">Source code templates</a>	<b>How?</b>	<b>Why?</b>
		SOFIA2 source code samples can be used to design source code templates for SOFIA2, which will be used as part of the code generator and code templates ACTIVAGE development tools.	

## 4.2.3 OpenIoT

This section is devoted to the presentation of the components comprising the prototype implementation release of the core OpenIoT platform that enable a user/developer to download, install and use the modules of OpenIoT platform. Since the OpenIoT platform will keep evolving over time, an updated version of the information provided in this section will be provided regularly at the OpenIoT Wiki<sup>8</sup> space under the Documentation<sup>9</sup> section.

### 4.2.3.1 Service Delivery & Utility

The Service Delivery & Utility Manager has a dual functionality. On the one hand (as a service manager), it is the module enabling data retrieval from the selected sensors comprising the OpenIoT service. On the other hand, the utility manager maintains and retrieves information structures regarding service usage and supports metering, charging and resource management processes.

#### 4.2.3.1.1 API

The current release of the OpenIoT Service Delivery & Utility Manager implements the functionalities/capabilities that are reflected in the interface listed in Table 6.

Table 6: List of primitives comprising the OpenIoT SD&UM implemented API

<<interface>> SDUManagerInterface
<b>pollForReport</b> (applicationID: String): SdumServiceResultSet
<b>getApplication</b> (applicationID: String): OAMO
<b>getService</b> (serviceID: String): OSMO
<b>getAvailableAppIDs</b> (userID: String): DescriptiveIDs
<b>getAvailableServiceIDs</b> (applicationID: String): DescriptiveIDs
<b>getUser</b> (userID: String): OpenlotUser

The services description as long as their inputs and outputs are listed Table 7.

<sup>8</sup> <https://github.com/OpenlotOrg/openiot/wiki>

<sup>9</sup> <https://github.com/OpenlotOrg/openiot/wiki/Documentation>



Table 7: Service Delivery &amp; Utility Manager implemented API definition

Service Name	Input	Output	Info
pollForReport	String serviceID	SdumServiceResultSet	Invokes a previously defined Service having the specified applicationID. This call will produce only one Result Set.
getService	String serviceID	OSMO	Used to retrieve the description (OSMO) of an available service. Requires as input a Service ID.
getApplication	String applicationID	OAMO	Used to retrieve the description (OAMO) of an available Application. Requires as input an Application ID.
getAvailableAppIDs	String userID	DescriptiveIDs	Used to retrieve the available applications (a list of applicationID / ServiceName / ServiceDescription triplet) already registered by a specific user. Requires as input a User ID.
getAvailableServiceIDs	String serviceID	DescriptiveIDs	Used to retrieve the available services (a list of serviceID / ServiceName / ServiceDescription triplet) already registered by a specific user. Requires as input a Service ID.
getUser	String userID	OpenlotUser	Used to retrieve the user's information for implementing access control mechanisms.

#### 4.2.3.1.2 Published Interface

This module is expected to be used from the OpenIoT Request Presentation user interface. Third party applications can invoke SD&UM services via restful web services at the URLs listed below:

- Welcome message listing the available services:  
<http://localhost:8080/sdum.core/rest/services/>
- Poll for Report:  
<http://localhost:8080/sdum.core/rest/services/pollforreport>
- Get Service Status:  
<http://localhost:8080/sdum.core/rest/services/getServiceStatus>
- Get Application:  
<http://localhost:8080/sdum.core/rest/services/getApplication>
- Get Service:  
<http://localhost:8080/sdum.core/rest/services/getService>
- Get User:  
<http://localhost:8080/sdum.core/rest/services/getUser>
- Get Available Application IDs:  
<http://localhost:8080/sdum.core/rest/services/getAvailableAppIDs>
- Get Available Service IDs:  
<http://localhost:8080/sdum.core/rest/services/getAvailableServiceIDs>

### 4.2.3.2 Linked Stream Middleware Light

Linked Stream Middleware Light (LSM-Light) is a platform that brings together the live real world sensed data and the Semantic Web. The implementation of the OpenIoT platform uses the LSM Middleware, which has been re-designed with push-pull data functionality and cloud interfaces for enabling additional cloud-based streaming processing.

An LSM deployment is available at <http://lsm.deri.ie/>. It provides functionalities such as 1) Wrappers for real time data collection and publishing; 2) A web interface for data annotation and visualization; and 3) A SPARQL endpoint for querying unified Linked Stream Data and Linked Data. The first and third functionality are the ones used in the proof-of-concept implementation in OpenIoT.

#### 4.2.3.2.1 API

In order for LSM-Light to support stream data processing programmatically, a Java API is provided. By using this API, a developer can add, delete and update GSN-generated sensor data into the implemented LSM-Light Server (triple store). Table 8 below illustrates the main API primitives that provide the LSM-Light functionalities, while Table 9 provides more details about all services that comprise the API.

Table 8: List of primitives comprising the OpenIoT LSM-Light API

<<interface>> LSMServerInterface	
<b>getSensorById</b> (sensorID:String):	Sensor
<b>getSensorBySource</b> (sensorSource:String):	Sensor
<b>sensorAdd</b> (newSensor:Sensor):	void
<b>sensorDataUpdate</b> (observation:Observation):	void
<b>sensorDataUpdate</b> (triples:String):	void
<b>deleteTriples</b> (graphURL:String,triples:String):	void
<b>deleteTriples</b> (graphURL:String):	void

Table 9: LSM-light API Specification

Service Name	Input	Output	Info
getSensorById	String sensorID	Sensor	Used to retrieve an existing sensor from LSM by sending a request. Requires as input a sensorID, in String format, which is a unique value to identify the sensor. Returns a Sensor object that includes all the available metadata describing the sensor.
getSensorBySource	String sensorSource	Sensor	Used to retrieve an existing sensor from LSM by sending a request. Requires as input a sensorSource in String format. Returns a Sensor object that includes all the available metadata.
sensorAdd	Sensor sensor	void	Used to register a new sensor into LSM. Requires as input a Sensor class instance. This method returns a notification and sensorId indicating whether the sensor was successfully added or not.
sensorDataUpdate	Observation observation	void	Used to update the latest observed data generated by a sensor. Requires as input an Observation object that includes all the available observed data. This method returns a notification indicating whether the observed data was successfully updated or not.

deleteTriples	String graphURL	void	Used to clear all the triple data of a specific graph. Requires as input the graphURL. This method returns a notification indicating whether the data were successfully removed or not. Note that the data cannot be restored after this method is called.
deleteTriples	String graphURL, String triples	void	Used to clear specific triples from a specific graph. Requires as input the graphURL and triple patterns.

### 4.2.3.3 User Interfaces

OpenIoT User Interfaces module comprises of three main components that can be useful to the development of ACTIVAGE UI. These are:

- Request Definition module: provides WYSIWYG UI to create service (similar to NodeRED) which can be an analytics service in case of ACTIVAGE.
- Request Representation: similar to Feature/Result viewer where it displays results using graphical representations.
- Schema Editor: only supports annotating sensors using the OpenIoT ontology. For ACTIVAGE ontology, it needs major changes to work.

#### 4.2.3.3.1 Request Definition

The request definition module is a web application that allows end-users to visually model their OpenIoT-based services using a node-based WYSIWYG (What-You-See-Is-What-You-Get) UI (User Interface). Modelled service graphs are grouped into “applications” (OAMOs). Each application e includes a collection of different services (OSMOs) which represent real life data (i.e. weather reports). This enables end-users to manage (describe/register/edit/update) their applications from a single user interface.

All modelled services are stored by the OpenIoT Scheduler and are automatically loaded when a user accesses the web application.

Figure 48 illustrates the main application interface components:

- The menu bar provides commands for creating new applications or for opening existing applications for editing. Once an application has been opened for editing, its name will appear at the top right of the menu bar.
- The central pane serves as the workspace area for modelling services.
- The node toolbox (left pane) contains the list of nodes that can be dragged into the workspace. Nodes are grouped by functionality.
- The properties pane (right pane) provides access to any selected node's properties.
- The console pane (bottom pane) provides workspace validation information (problems/warnings) as well as a debug preview of the generated SPARQL code for the designed service.

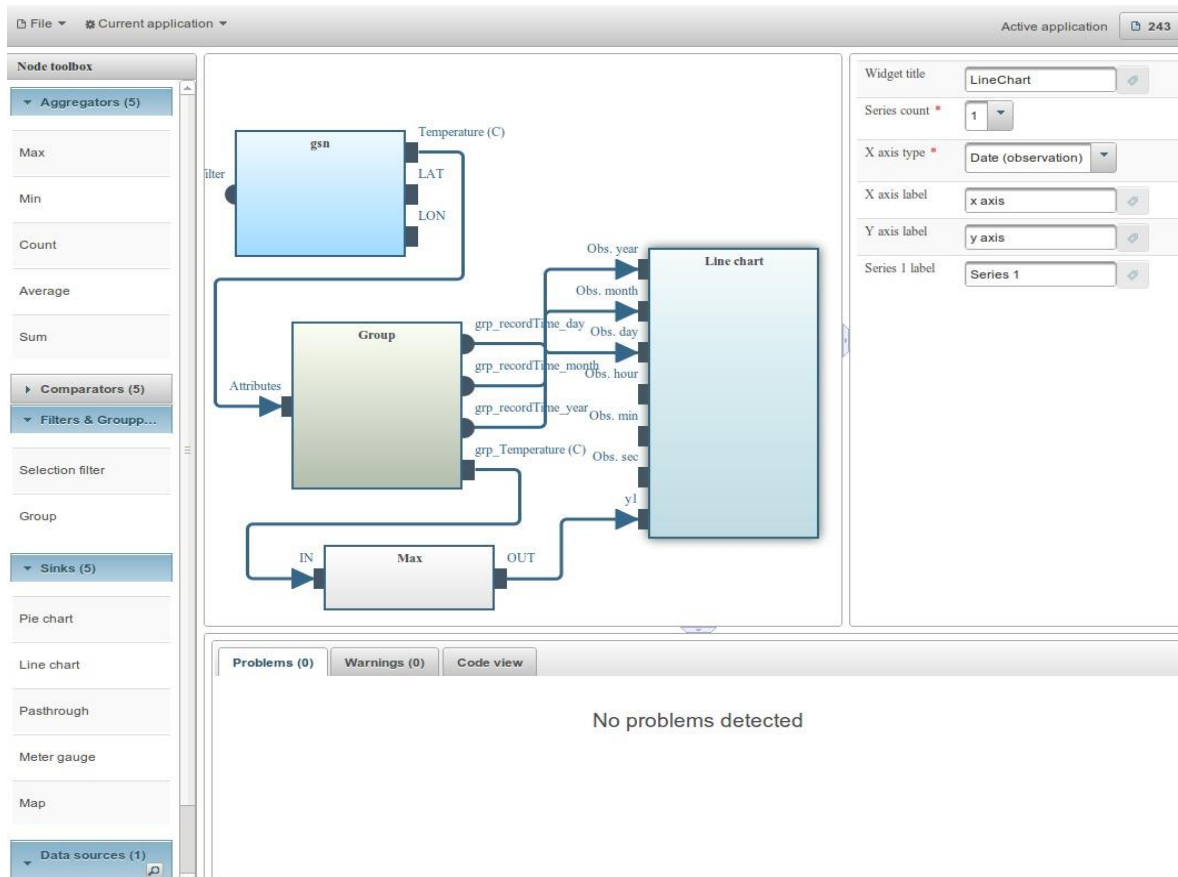


Figure 48: Request Definition User Interface (UI)

Below is a list of the Request Presentation main functionalities:

- Create a new application
- Load and Edit an existing application
- Modelling the service graph of an application by the configuration and usage of the provided toolboxes and more specifically the:
  - Data source node
  - Selection filter node
  - Comparator nodes
  - Group node
  - Aggregation nodes
  - Sink nodes
  - Line chart sink node
  - Pie chart sink node
  - Meter gauge sink node
  - Map sink node
  - Passthrough sink node
- Workspace validation
- Save/update an application

### 4.2.3.3.2 Request Representation

The request presentation module is a web application that provides end-users with a visual interface to services created using the Request Definition web application. When a user accesses the web application, all his/her modelled applications are automatically loaded. Each application contains one or more visualization widgets.

To access the widget dashboard for a particular application, click on the file menu and then the open application sub-menu, and select an application to load. The request presentation layer will parse the application metadata and generate a self-updating widget dashboard (Figure 49).

Dashboards refresh automatically every 30 seconds. However, the user may manually trigger an update by clicking on the current application menu and selecting the “Manual data refresh” option. To clear the data of a specific widget, click on the “Clear data” button on its top-right corner.

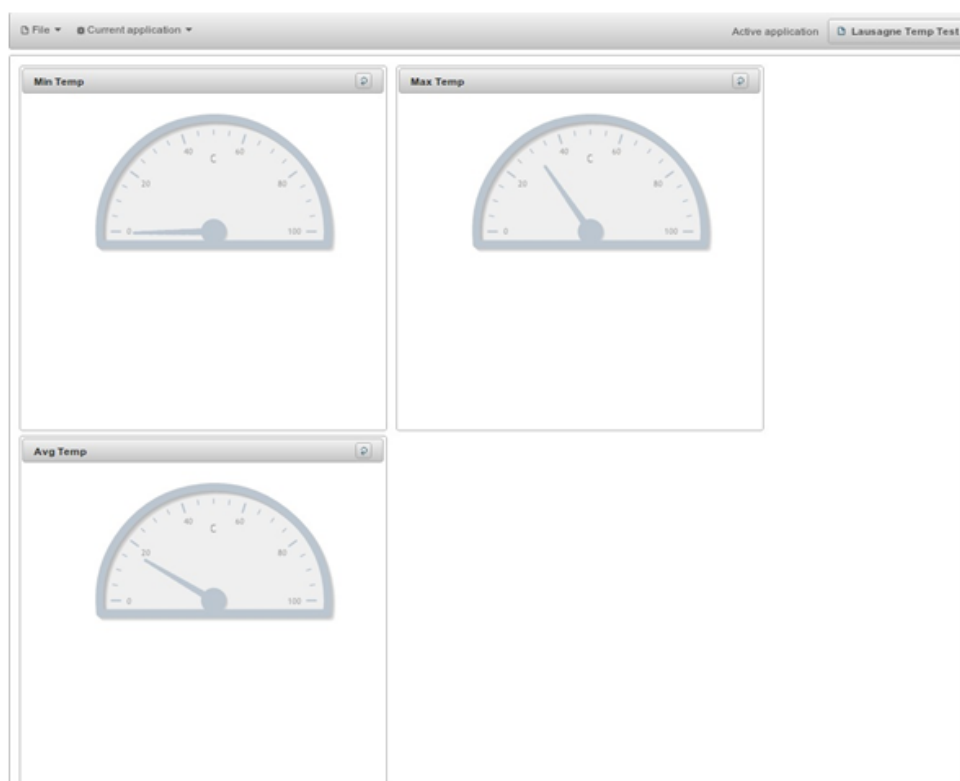


Figure 49: Request Presentation UI

### 4.2.3.3.3 Schema Editor

The Sensor Schema Editor supports the average user in annotating sensor and sensor-related using the OpenIoT ontology and Linked Data principles. The interface automates the generation of RDF descriptions for sensor node information submitted by users.

The Sensor Schema Editor has two parts, namely the frontend interface and the backend LD4Sensors Server. The Schema Editor interface depends on the LD4Sensors Server for generating descriptions of sensor metadata and sensor observations.

The LD4Sensor exposes a REST API that takes as input the sensor metadata and returns the RDF representation. The Sensor Schema editor is developed using a JSF framework and

deployed on JBOSS AS 7.1.1. The LD4Sensor server is a standalone server that runs the restlet framework (<http://restlet.org/>).

#### 4.2.3.4 Utilities & Libraries

Finally in OpenIoT a project called Commons is maintained where the "common" Objects, Schemata and utilities used for most of the modules across the OpenIoT platform are stored and developed. This project is included in the binary file distribution as a library to all the projects that are using it but has to be added as a separate project for development. This project is available under the `/utils/utis.common`<sup>10</sup> folder.

The modules that are currently using this library are the:

- Request Definition
- Request Presentation
- Service Delivery & Utility Manager (SD&UM)
- Scheduler
- LSM-Light

Currently util.common hosts the following objects, schemata, utilities:

- Schemata (xsd):
  - OSDSpec
  - SdumServiceResultSet
  - SPARQL
    - protocol-types,
    - rdf,
    - result
  - AppUsageReport
  - DescriptiveIDs
  - SensorTypes
- Java models generated from schemata:
  - descriptiveids
  - osdspec
  - SDUM
    - appusagereport
    - serviceresultset
  - sensortypes
  - sparql
    - protocol-types,
    - rdf,
    - result
- Utilities
  - CDataAdapter
  - PropertyManagement

---

<sup>10</sup> <https://github.com/OpenlotOrg/openiot/tree/master/utils/utis.common>

## 4.2.3.5 Source Code & Binaries

### 4.2.3.5.1 Source Code Availability

The OpenIoT repository is hosted at the GitHub<sup>11</sup> and can be found at the following link: <https://github.com/OpenlotOrg/openiot>

The OpenIoT repository is divided into branches. Each branch is separated into two thematic categories. One is the Documentation (i.e., site storage hosted at the “gh-pages”). And the other one is the Open IoT source code branches. Under the source code category various Branches will exist that are listed below:

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

### 4.2.3.5.2 Source code Structure

The OpenIoT source code is organised in functionality themes. For example all utilities are under the “utils” folder and all user interfaces under the “ui” folder. The code structure is shown below:

- doc: provides all the related documents with the platform.
- Modules: provides the core modules of the platform
  - CUPUS
  - QoSManager
  - 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
    - security-webapp-demo
- sandbox: provides space for developers to store their test code/apps (developers “playground”).
- Ui: provides all the modules related to the User Interface
  - RDFSensorSchemaEditor

---

<sup>11</sup> <http://github.com/>

- Ide.core
  - ui.requestCommons minor
  - ui.requestDefinition
  - ui.requestPresentationaEditor
- utils: provides utilities related with the platform
- demoData
  - lib
  - utils.common

### 4.2.3.5.3 Binaries Availability

OpenIoT binaries are available through the OpenIoT Wiki site<sup>12</sup> under the Users>Downloads<sup>13</sup> 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.

### 4.2.3.6 Documentation

The OpenIoT Wiki is publicly available on Github<sup>14</sup> which provides access to all support materials necessary for developers and users to start working on and/or using the platform.

### 4.2.3.7 Mapping between OpenIoT and ACTIVAGE Development Tools

Part of the OpenIoT development tools described above correspond to specific ACTIVAGE development tools, described in Section 4.1. This mapping is presented in Table 10. Some of the mapped tools may be used within the AIOTES infrastructure, possibly with some modifications or generalizations. Table 10 also summarizes which tools can be generalized to be used within AIOTES, or are too specific to be included.

Table 10: Mapping between OpenIoT and ACTIVAGE development tools.

OpenIoT development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
Service Delivery & Utility	Not mapped with AIOTES		
Linked Stream Middleware Light	Not mapped with AIOTES		
User Interfaces / Request Definition	Data/Visual Analytics Tools / Data Analyser	Yes	No
		<b>How?</b>	<b>Why?</b>
		Request Definiton module provides WYSIWYG UI to create service (similar to NodeRED) which can be	

<sup>12</sup> <https://github.com/OpenlotOrg/openiot/wiki>

<sup>13</sup> <https://github.com/OpenlotOrg/openiot/wiki/Downloads>

<sup>14</sup> <https://github.com/OpenlotOrg/openiot/wiki>



		an analytics service in case of ACTIVAGE.	
User Interfaces / Request Representation	Data/Visual Analytics Tools / Feature/Result viewer	Yes	No
		<b>How?</b>	<b>Why?</b>
		Request Representation is similar to Feature/Result viewer where it displays results using graphical representations.	
User Interfaces / Schema Editor	Semantic Interoperability Layer tools / Device Semantics Editor	Yes	No
		<b>How?</b>	<b>Why?</b>
			Schema Editor only supports annotating sensors using the OpenIoT ontology. For ACTIVAGE ontology, it needs major changes to work.
Utilities & Libraries	Not mapped with AIOTES		
Documentation	Support	Yes	No
		<b>How?</b>	<b>Why?</b>
		The OpenIoT Documentation can be used as part of the overall AIOTES support material.	
Source Code & Binaries	IDE / Code generator IDE / Source code templates	Yes	No
		<b>How?</b>	<b>Why?</b>
			OpenIoT code base is old and uses many outdated libraries which might generate conflicts during compilation. Until we are clear of how the code generating functions work, we cannot commit to making these features.

## 4.2.4 sensiNact

Eclipse sensiNact aims at creating a common environment in which heterogeneous devices can exchange information and interact among each other in the IoT world.

Eclipse sensiNact is composed of two tools, sensiNact Gateway aiming at integrating devices and aggregating data from various sources and sensiNact Studio aiming at interacting with the sensiNact Gateway to visualize the devices and the data.



#### 4.2.4.1 sensiNact documentation and WIKI

The sensiNact wiki documentation is accessible at <https://wiki.eclipse.org/SensiNact> and help to install, develop and use sensiNact in application environment with different use cases.

The wiki proposes two guides. A quick guide for the users wanting to setup the installation of sensiNact quickly and full guides where a detailed description is proposed to understand some of the concepts behind sensiNact.

A Tutorials section proposes various tutorials to help users and developers to learn more about sensiNact.

- How to create a sNa application (requires no particular knowledge)
- How to develop a southbound bridge (requires knowledge about Java, such as the architecture and the data model of sensiNact)

The sensiNact project uses the continuous integration infrastructure from Eclipse.

The Eclipse Jenkins runs a compilation of the sensiNact Gateway every day (or night, depending on your timezone). The resulting compilation generates a stand alone snapshot distribution that is available at the following address: latest build. Ten builds are kept in the download area of Eclipse.

Each module of the sensiNact gateway is available in the Eclipse Nexus.

#### 4.2.4.2 sensiNact API doc and swagger

As described in the D3.2 deliverable, the sensiNact IoT platform provides the following REST APIs:

**Storing API** A REST API is available for both data read (paginated or not) and write (singular and plural) in the historical database. Statistical information is also available. More details on the historical storage API can be found in <https://git-lialp.intra.cea.fr/sensinact/historical-storage/tree/master/rest-endpoint>, and a swagger documentation page is given in <http://193.48.18.251:8080/swagger-ui.html>.

**API Format** For all sensiNact available REST API (historical storage, data model, event and service API) format is JSON.

**Data Model API** The sensiNact data model API is based on the information model as described in Figure 50: sensiNact generic data model. This generic data model allows a similar access to the sensors and actuators using heterogeneous protocols.. A REST API is available for reading inside a deployed instance of the data model. Using this API it is possible to retrieve providers, services and resources, get last measured value on a given resource and set current value. More details on the data model API are given in <http://open-platforms.eu/library/sensinact-aka-butler-smart-gateway/> and a swagger documentation can be found in <http://sensinact.ddns.net/swagger-api/index.html#/>.

**Services and Events API** Subscription and unsubscription to a resource exposed by a specified service provided by a given provider can be managed through the same API. See <http://open-platforms.eu/library/sensinact-aka-butler-smart-gateway/> and the swagger documentation <http://sensinact.ddns.net/swagger-api/index.html#/>.

The sensiNact Service and Resource model allows exposing the resources provided by an individual service. The latter, characterized by a service identifier, represents a concrete physical device or a logical entity not directly bound to any device. Each service exposes resources and could use resources provided by other services. Figure 50 depicts the Service and Resource model.

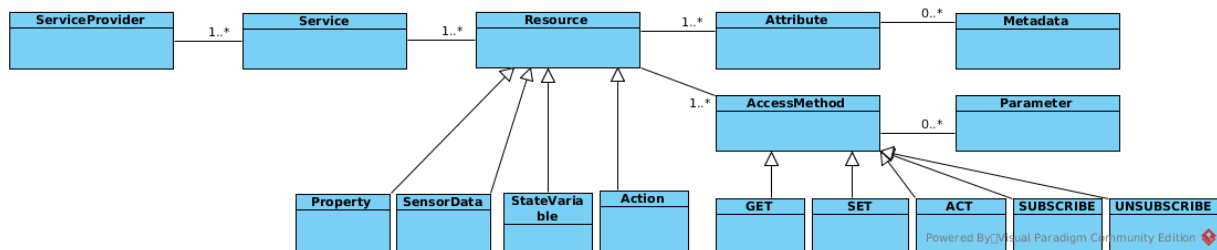


Figure 50: sensiNact generic data model. This generic data model allows a similar access to the sensors and actuators using heterogeneous protocols.

Resources and services can be exposed for remote discovery and access using different communication protocols, such as HTTP REST, JSON-RPC, etc., and advanced features may also be supported (as semantic-based lookup). Resources are classified as shown in Figure 50, while the access methods are described in Table 11.

Table 11: sensiNact resources types and description.

TYPE	DESCRIPTION
<b>SENSORDATA</b>	Sensory data provided by a service. This is real-time information provided, for example, by the SmartObject that measures physical quantities.
<b>ACTION</b>	Functionality provided by a service. This is mostly an actuation on the physical environment via an actuator SmartObject supporting this functionality (turn on light, open door, etc.) but can also be a request to do a virtual action (play a multimedia on a TV, make a parking space reservation, etc.)
<b>STATEVARIABLE</b>	Information representing a SmartObject state variable of the service. This variable is most likely to be modified by an action (turn on light modifies the

	light state, opening door changes the door state, etc.) but also to intrinsic conditions associated to the working procedure of the service
<b>PROPERTY</b>	Property exposed by a service. This is information which is likely to be static (owner, model, vendor, static location, etc.). In some cases, this property can be allowed to be modified.

Table 12: sensiNact resource's access method.

TYPE	DESCRIPTION
<b>GET</b>	Get the value attribute of the resource
<b>SET</b>	Sets a given new value as the data value of the resource
<b>ACT</b>	Invokes the resource (method execution) with a set of defined parameters
<b>SUBSCRIBE</b>	Subscribes to the resource with optional condition and periodicity
<b>UNSUBSCRIBE</b>	Remove an existing subscription

The access methods that can be associated to a resource depend on the resource type, for example, a GET method can only be associated to resources of type Property, StateVariable and SensorData. A SET method can only be associated to StateVariable and modifiable Property resources. An ACT method can only be associated to an action resource. SUBSCRIBE and UNSUBSCRIBE methods can be associated to any resources.

#### 4.2.4.3 sensiNact source code public access

The sensiNact platform is an open source project, hosted as a technology project in the Eclipse foundation. The sensiNact license is the Eclipse License v1.0.

Source code is accessible through the Eclipse git repositories:

<https://projects.eclipse.org/projects/technology.sensinact/developer>

You can use the code from these repositories to experiment, test, build, create patches, issue pull requests, etc. This project uses Gerrit Code Review; please see contributing via Gerrit.

sensinact/org.eclipse.sensinact.gateway

Clone: <https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.gateway>

Review With Gerrit: <https://git.eclipse.org/r/p/sensinact/org.eclipse.sensinact>

Browse Repository: <http://git.eclipse.org/c/sensinact/org.eclipse.sensinact.git>

sensinact/org.eclipse.sensinact.studio

Clone: <https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.studio>

Review With Gerrit: <https://git.eclipse.org/r/p/sensinact/org.eclipse.sensinact.studio>

Browse Repository: <http://git.eclipse.org/c/sensinact/org.eclipse.sensinact.studio.git>

sensinact/org.eclipse.sensinact.studioweb

Clone: <https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.studioweb>

Review With Gerrit: <https://git.eclipse.org/r/p/sensinact/org.eclipse.sensinact.studioweb>

Browse Repository: <http://git.eclipse.org/c/sensinact/org.eclipse.sensinact.studioweb.git>

#### 4.2.4.4 sensiNact tutorial and sample source code

Through the sensiNact wiki web page, it is possible to reach the tutorials public repository, organized as follow:

##### sensiNact tutorials

This section proposes various tutorials to help users and developers to learn more about sensiNact.

- How to create a sNa application (requires no particular knowledge)
- How to develop a southbound bridge (requires knowledge about Java, such as the architecture and the data model of sensiNact)

##### sensiNact/Tutorial Studio

This tutorial aims at discovering the sensiNact Gateway and the sensiNact Studio. In this tutorial, you will mainly use the Studio or the RESTful API enabling to interact with the Gateway.

This tutorial give details on the following topics:

- sensiNact environment overview
- Configure the Studio
- Create an application
- Deploy and start an application
- Access the Gateway using the RESTful API

#### 4.2.4.5 sensiNact ACTIVAGE development tools

##### 4.2.4.5.1 sensiNact ACTIVAGE documentation and WIKI

The co-conception process organized in DS6 resulted in a specification document called “cahier de conception”. This conception file describes the features that are expected to be implemented, based on the IoT system to be deployed in DS6. These feature specifications are used as basis for the functional analysis that specifies the needed IoT infrastructure and topology. Resulting hardware devices and communication network are described in D3.6 deliverable.

The functional analysis provides also the service layer specifications that describes the mandatory AHA functions to be provided in the service layer in order to fulfil the expected features.

This service layer specification specifies the eight first AHA functions for DS6.

The following function providers are available in the sensiNact AHA service API, classified by three different main categories: building configuration functions, AHA alert functions and AHA monitoring functions.

**Building configuration functions API** proposes two providers in order to describe and parameterize the instrumented rooms and buildings.

AHA-building function provider

AHA-studio function provider

**Notification and alert functions API** proposes providers that trigger notifications (towards patient) and alerts (towards carers) when attention should be given on the elderly comfort and activity.

- AHA-temperature-control
- AHA-bedroom-activity-alert
- AHA-bathroom-activity-alert
- AHA-shower-alert

**Activity monitoring function API** provides service to measure both “in live” and historized nightly and daily activity.

- AHA-night-rising-monitor
- AHA-day-laying-monitor

These 8 AHA functions are implemented on the top of the generic sensiNact API, using the provider/service/resource pattern described in section 4.2.4.2. AHA functions are specialization of providers, with dedicated services and resources.

The table below describes the 8 specified AHA function for DS6:

Table 13: The eight implemented sensiNact AHA functions.

Category	Function	Description
<b>Building configuration</b>	AHA-building	Provides service to configure the overall building gathering several studios
	AHA-studio	Provides service to configure the elementary living place considered for AHA functions. This elementary living place is called studio and is constituted of a bedroom and a bathroom
<b>Notifications and alerts</b>	AHA-temperature-control	Provides service to trigger a notification to the patient when an uncomfortable temperature is measured in a given bedroom, and then alert to carer if necessary (no notification acquitment by the patient)
	AHA-bedroom-activity-alert	Provides service to trigger a notification to the patient when a long inactivity is detected in a given bedroom, and then alert to carer if necessary (no notification acquitment by the patient)
	AHA-bathroom-activity-alert	Provides service to trigger alert to carers when long inactivity is detected in a given bathroom
	AHA-shower-alert	Provides service to trigger alert to carers when long shower duration is measured in a given bathroom
<b>Activity monitoring</b>	AHA-night-rising-monitor	Provides service to get “in live” and past occurrences of risings during night in a given bedroom.
	AHA-day-laying-monitor	Provides service to get “in live” and past occurrences of bed presences during daytime in a given bedroom.

Two supplementary functions are under development in the DS6 deployment site

Table 14: Two supplementary AHA functions under development in sensiNact.

Category	Function	Description
<b>Notifications and alert</b>	AHA-pain-alert	Provides service to trigger a notification to the patient to auto-evaluate his pain degree and then alert the carer if

		necessary
<b>AIOTES management</b>	AHA-kpi	Provides service to measure technical performance indicators such as the number of installed sensors/rooms/buildings, size and number of records in the rough database.

The link to download and view the documentation is:

[ACTIVAGE > 3 ACTIVAGE PROJECT > 010 Work Packages > 004 WP4 - CERTH - App. Su... > T4.1 > ACTIVAGE-DS6-sensiNact-AHA-service-API-171204.xlsx](#)

### 4.2.4.5.2 sensiNact ACTIVAGE API doc and swagger

#### 4.2.4.5.2.1 Documentation of the sensiNact AHA service API

As described in Table 13 and Table 14, the sensiNact AHA service API gives access to ten specified functions (see Figure 51)

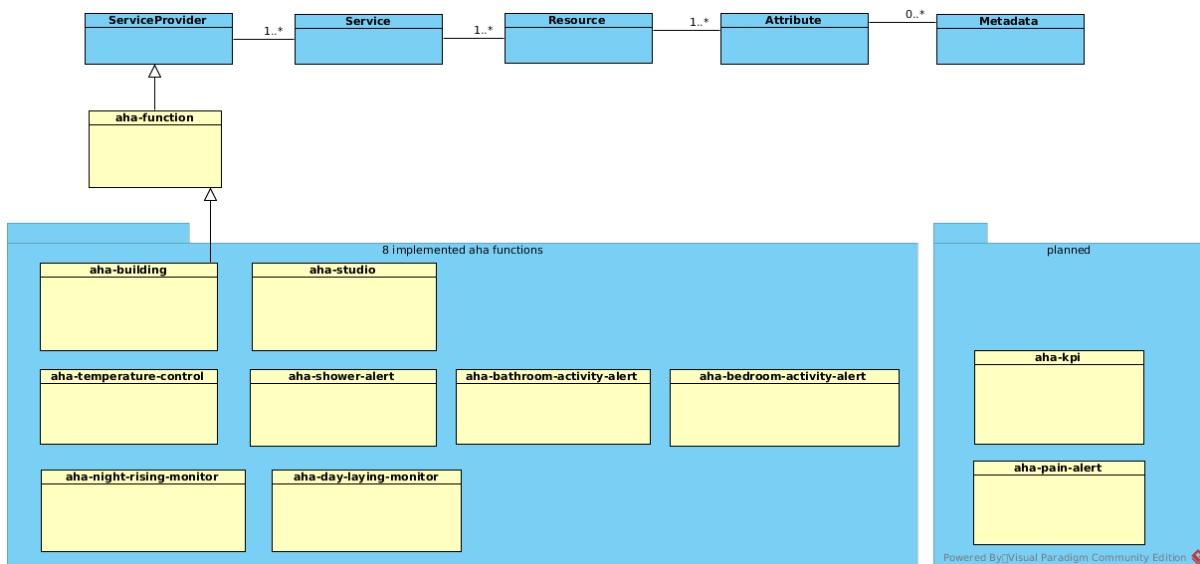


Figure 51: the sensiNact AHA functions (implemented and in progress)

The sensiNact AHA service API specializes the generic sensiNact API, proposing one dedicated provider by function.

The class diagram below describes the sensiNact AHA service API specialization from the generic sensiNact API

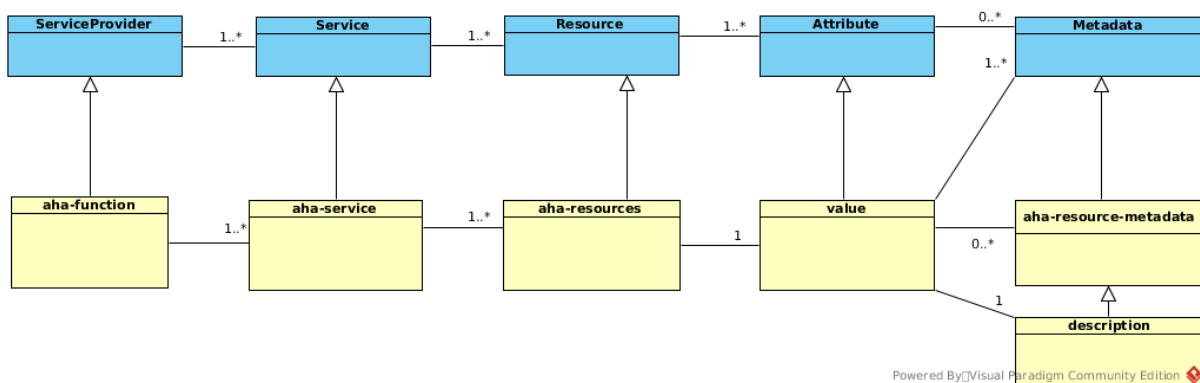


Figure 52: sensiNact AHA service API specialization

This hierarchy diagram shows the specialization of AHA service API in terms of attribute (which is always value attribute) and metadata for the value attribute that always contains at least one description metadata.

This last constraint on description metadata makes the API auto documented. Thus API user can find information on every resource through this resource/value/description information.

The class diagram below describes the available AHA services specified for AHA function providers

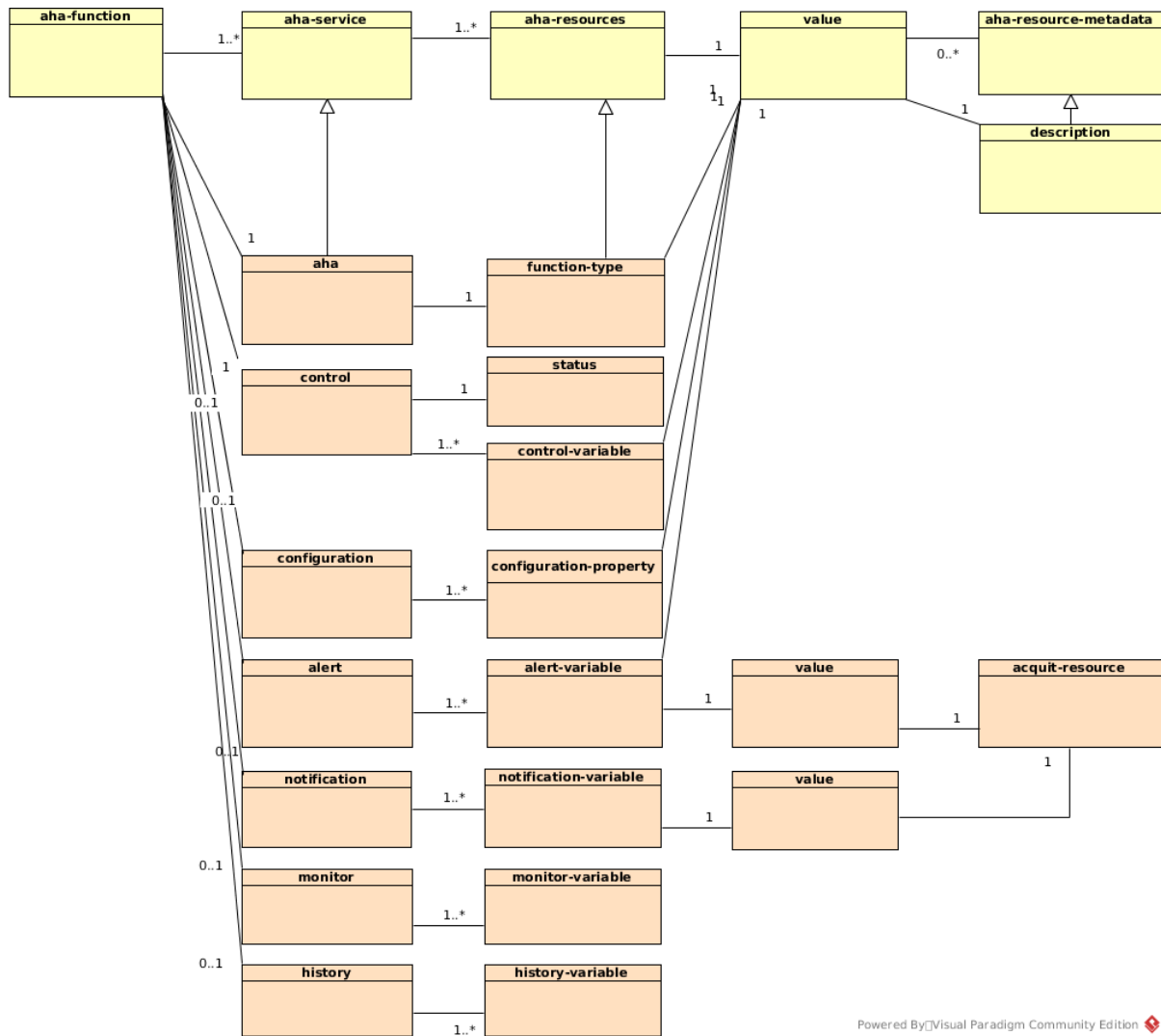


Figure 53: list of available sensiNact AHA services

This class diagram shows all available services for AHA functions. There are mainly 7 different AHA services: ‘aha’, ‘configuration’, ‘alert’, ‘notification’, ‘control’, ‘monitor’, ‘history’.

All AHA functions are marked with a “aha” named service. This “aha” service is used when using the sensiNact API, to filter only AHA function providers. The “function-type” resource in the “aha” service gives a supplementary key for filtering among AHA functions. Example of function-type resource value is “TEMPERATURE\_CONTROL”

All AHA functions also provide a “control” service. The “control” service holds at least one “status” resource that is used to activate/deactivate the AHA function. This “control” service can contain also acquitment resources to be set when acquitting notifications (by patient) and alerts (by carer).



The “configuration” service is used to parametrize the function when necessary. Example of configuration resource is “comfort-range-threshold” that set the lower temperature value of the comfort range in a given room.

The “alert” and “notification” services bring exposes the resource value of the notification to patient (notification service) or alert to carers (alert service). The alert and notification resources need to be linked to acquitment resources (that should be set by patient/carers). This link between alert/notification resource with acquitment resource is described in the “acquit-resource” metadata.

The “monitor” service gives the activity status with dedicated resource values. Example of monitor resource is “is-night-rising” that checks if the patient is currently rising during the night.

The “history” service provides predefined history samples for a given resource, with a constant horizon (30 days) and a given sampling period (1 day). Example of such a history resource is “last-30-nights-rising-counters” that returns an array of 30 integers where each integer is the number of night risings for each of the last 30 days. Index 0 is today, 1 is yesterday (etc.). These history data are computed through a dedicated “historical statistics agent” bundle embedded in the ACTIVAGE specialized version of the sensiNact distribution (cf.4.2.4.5.3 )

A complete documentation of the height available sensiNact AHA functions is available and published in the livelink ACTIVAGE project shared repository. The link to download and view the documentation is:

[ACTIVAGE > 3 ACTIVAGE PROJECT > 010 Work Packages > 004 WP4 - CERTH - App. Su... > T4.1 > ACTIVAGE-DS6-sensiNact-AHA-service-API-171204.xlsx](#)

In addition to these functions API, the sensiNact AHA service API provides a helper API which role is to ease the use of the functions API:

The sensiNact AHA helper API gives also filtering and grouping features:

Filtering API: available filtering by alert / monitoring / building

Grouping API: feature to set a group of provider with same resource values in a single request.

#### 4.2.4.5.2.2 Swagger of the sensiNact AHA service API

A swagger documentation of the sensiNact AHA service API is available on line:

<http://193.48.18.245:8081/swagger-api/index.html>

The swagger web application provides the list of available requests as illustrated in Figure 54.

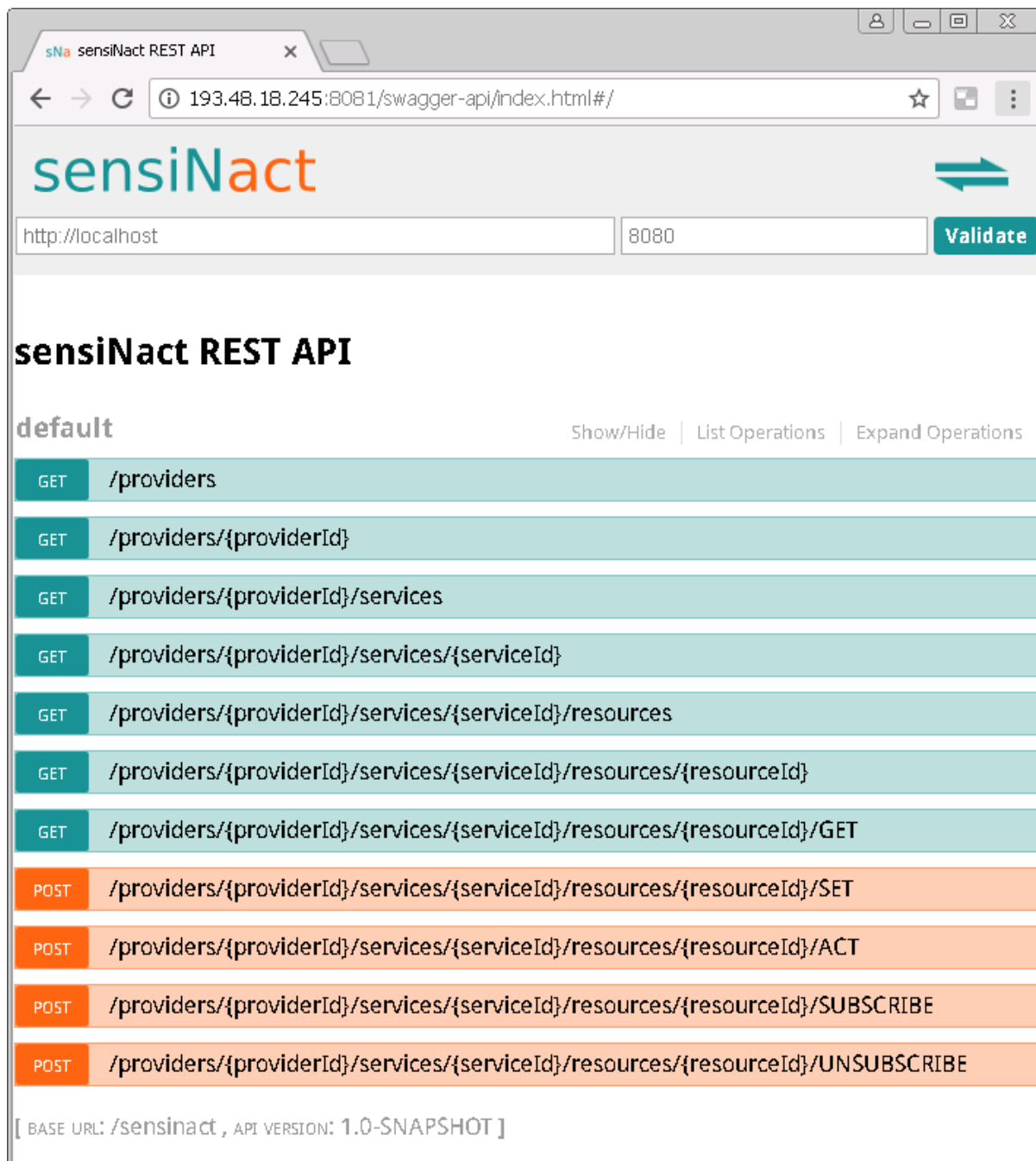
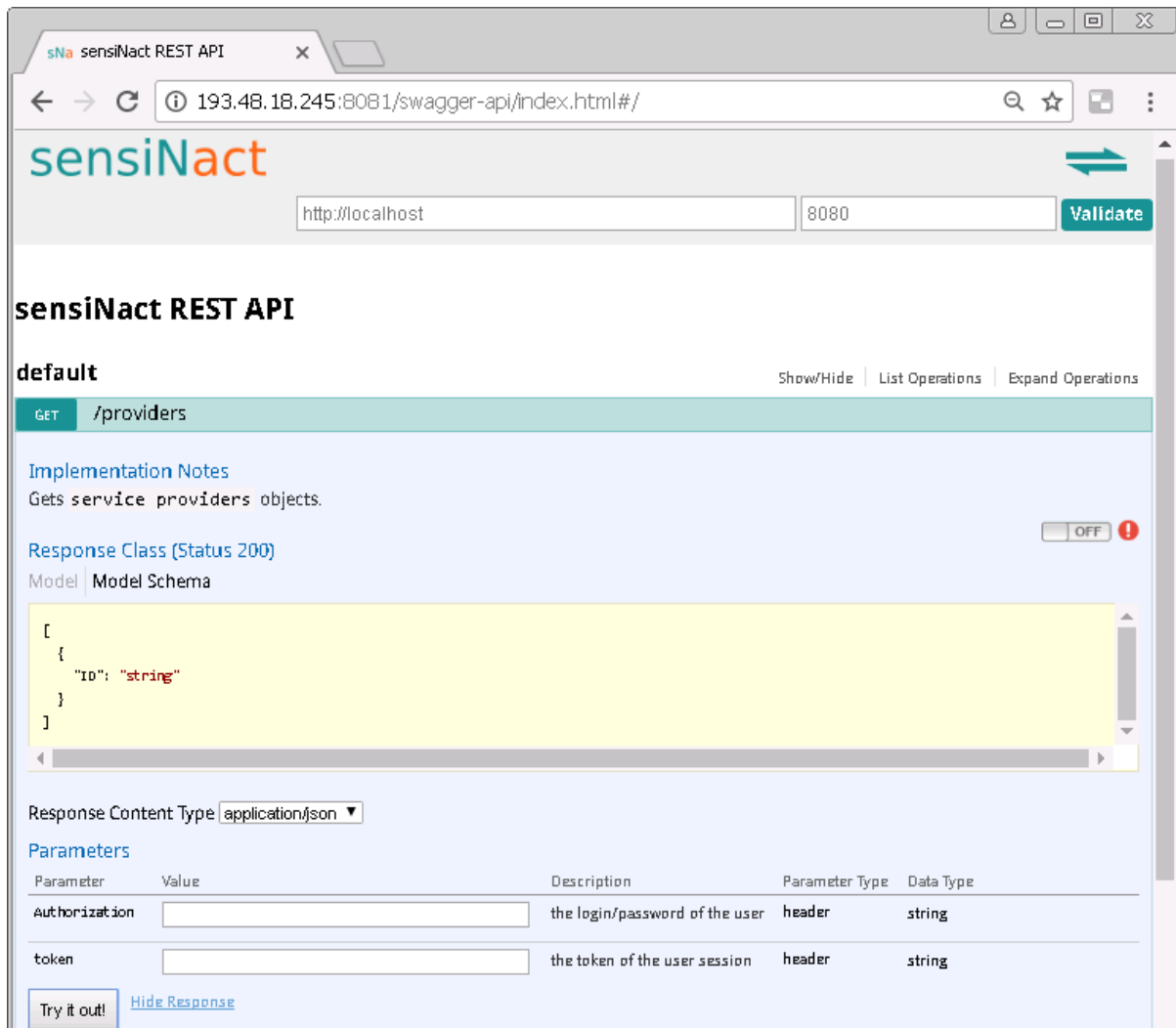


Figure 54: swagger screenshot for the sensiNact AHA service REST API

For each request, through the web application, it is possible to test the REST http requests as specified in the documentation (cf. Section 5).

For example, the request to get all the available providers is described expanding the “GET /providers” section:



The screenshot shows a web browser window displaying the Swagger UI for the sensinact REST API. The browser address bar shows the URL `193.48.18.245:8081/swagger-api/index.html/#/`. The page title is "sensiNact REST API". Below the title, there is a search bar with "http://localhost" and a port field with "8080", and a "Validate" button. The main content area is titled "sensiNact REST API" and shows the "default" environment. The selected endpoint is "GET /providers". Underneath, there are "Implementation Notes" stating "Gets service providers objects." and a "Response Class (Status 200)" section. The response class is a JSON array of objects, with a sample shown in a yellow box: 

```
[ { "id": "string" } ]
```

. Below the response class, there is a "Parameters" table with two entries: "Authorization" and "token".

Parameter	Value	Description	Parameter Type	Data Type
Authorization	<input type="text"/>	the login/password of the user	header	string
token	<input type="text"/>	the token of the user session	header	string

At the bottom of the parameters table, there is a "Try it out!" button and a "Hide Response" link.

The swagger description for this particular “GET / providers “ request contains:

- A short description of the request
- The format of the expected answer
- The mandatory and optional parameters

Once parameter values have been entered, click the “Try it out!” button send the request.

The sent http request is displayed, and the response code, header and body.

Try it out!
[Hide Response](#)

**Curl**

```
curl -X GET --header "Accept: application/json" "http://193.48.18.245:8081/sensinact/providers"
```

**Request URL**

```
http://193.48.18.245:8081/sensinact/providers
```

**Response Body**

```
{
  "statusCode": 200,
  "providers": [
    "bedroom-alert-207",
    "night-rising-monitor-207",
    "207",
    "sensinact",
    "temperature-control-pt1",
    "shower-alert-207",
    "aha-bedroom-activity-alert-bedroom-PTL",
    "aha-shower-activity-alert-bathroom-PTL",
    "temperature-control-208",
    "PTL-studio",
    "aha-night-rising-monitor-bedroom-PTL",
    "day-laying-monitor-207",
    "aha-bathroom-activity-alert-bathroom-PTL",
    "SSR Les Granges",
    "bathroom-alert-207",
    "temperature-control-207",
    "PTL",
    "208"
  ]
}
```

**Response Code**

```
200
```

**Response Headers**

```
{
  "date": "Thu, 21 Dec 2017 11:27:25 GMT",
  "server": "Jetty(9.2.6.v20141205)",
  "x-powered-by": "Jetty(9.2.6.v20141205)",
  "access-control-allow-methods": "GET,POST",
  "content-type": "application/json; charset=utf-8",
  "access-control-allow-origin": "*",
  "x-auth-token": "3df4f368aec5b31e6b8feaf601f94a8e386f02f4",
  "content-length": "518"
}
```

### 4.2.4.5.3 sensinAct AHA historical statistics agent

The sensinAct AHA historical statistics agent is a bundle embedded in the ACTIVAGE distribution of sensinAct which role is to compute specialized statistics aggregations in the database. These AHA specialized statistics computations have been specified by the medical team with ethical considerations compatible with GDPR recommendations (limitation of data processing and storage). This module is designed to compute data for two different sampling periods (mainly daily and rarely hourly) for a time horizon of 30 days. The table below gives the list of available statistics computations provided by the sensinAct AHA historical statistics agent.

Table 15: sensiNact AHA historical statistics agent (30 days horizon)

Statistics aggregation	Sampling period
dayLayingCounter	day
dayLayingDuration	day
nightRisingCounter	day
nightRisingDuration	day
tapOnDuration	day
showerOnDuration	day
showerAlertCounter	day
washingNotificationCounter	day
temperatureAlertCounter	day
stepCounter	day and hour
painAlertCounter	day
averagePainLevel	day and hour
falseAlertCounter	day
alertCounter	day
notificationAcquitmentCounter	day

#### 4.2.4.5.4 sensiNact public access to ACTIVAGE dedicated source code

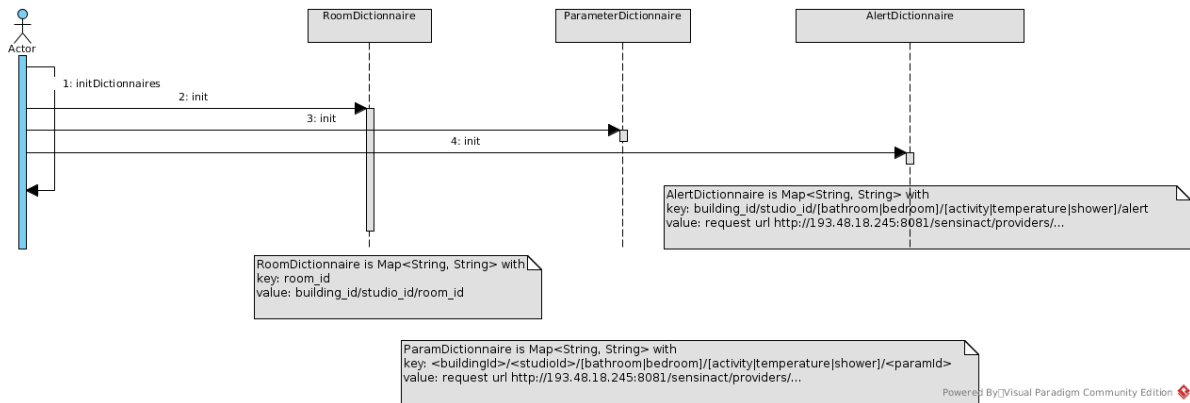
The source code is not available yet in a public git repository. AHA sensiNact bridges implementation will be pushed to the sensiNact public git repository in the next months.

#### 4.2.4.5.5 sensiNact tutorial and sample source code

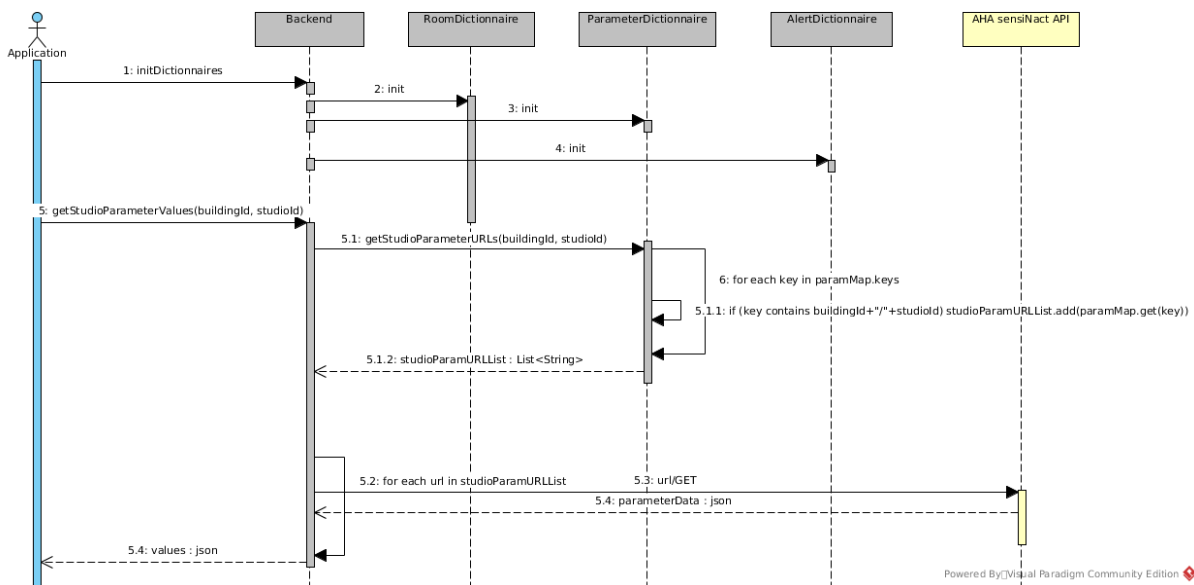
In this section, we present the work that has been done in collaboration with application developers, from the Technosens technical team, customers of the AHA service API.

The work aims to make the use of AHA service API easier, and to minimize the number of http requests.

The principle of the code optimization is to manage temp dictionaries in a backend between the application and the AHA service API.

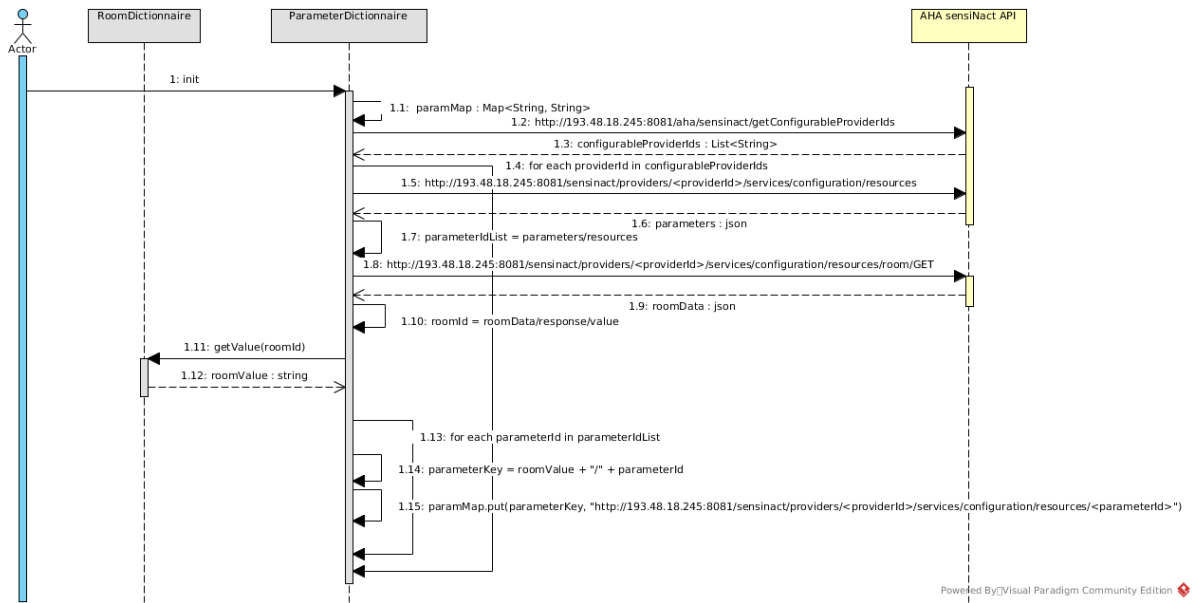


The sequence diagram below illustrates the use of a parameter dictionary to improve the configuration parameter access for a given studio in a building.



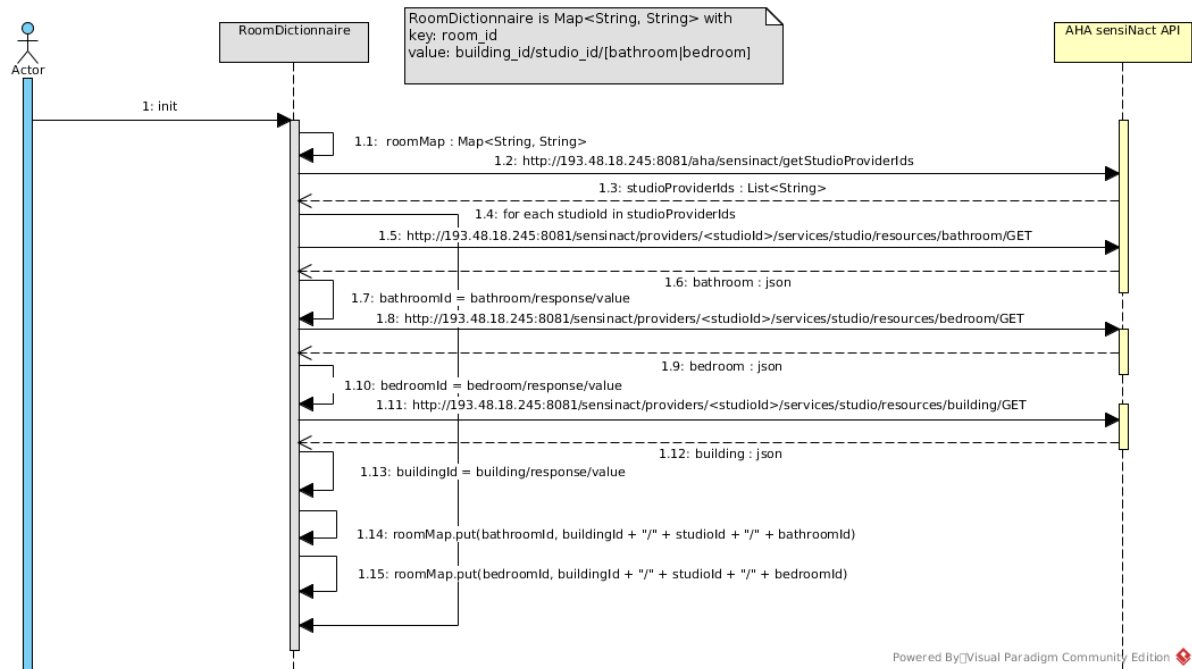
Following the proposed sequence diagram, several requests to retrieve the needed providers are factorized in a pre initialization of the parameter dictionary.

The next sequence diagram describes the initialization of the parameter dictionary

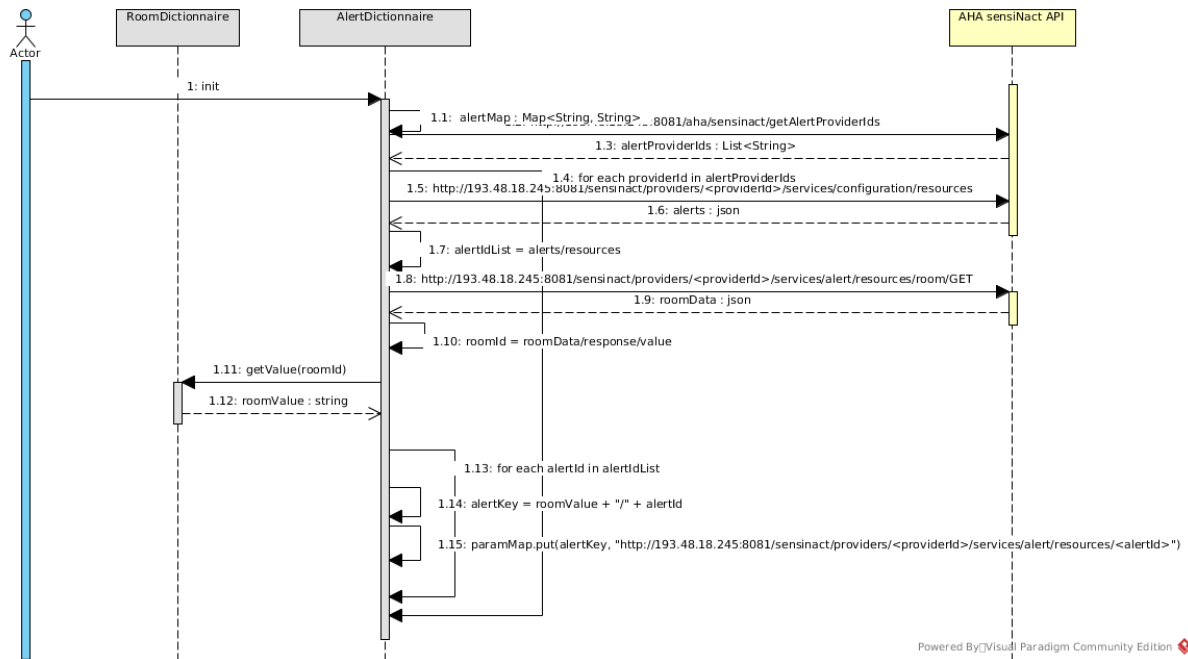


The same way, sequence diagrams are available to pre initialize room dictionary and alert dictionary

The next sequence diagram describes the initialization of the room dictionary



The next sequence diagram describes the initialization of the alert dictionary



From these sequence diagram, Technosens implemented the backend component in the application layer, and provided source code that can be used as sample source code for the AHA service API usage.

First, simple pojo for room and response are proposed:

```

package fr.technosens.alertes.backend.sensiNact;

class SNaRoom {

    final String buildingId, studioId, studioName, roomId, roomType;

    SNaRoom(String buildingId, String studioId, String studioName, String roomId, String
roomType) {
        this.roomId = roomId;
        this.buildingId = buildingId;
        this.studioId = studioId;
        this.studioName = studioName;
        this.roomType = roomType;
    }
}
    
```

```

package fr.technosens.alertes.backend.sensiNact;

import java.util.ArrayList;
import java.util.List;

class SNaResponse {

    int statusCode;

    boolean isSuccess() {
        return statusCode == 200;
    }

    static class ProviderIdsResponse extends ArrayList<String> {}

    static class ResourcesResponse extends SNaResponse {
        List<String> resources;
    }

    static class ResourceResponse extends SNaResponse {
    }
}
    
```



```

ResourceResponseResponse response;

static class ResourceResponseResponse {
    String name;
    long timestamp;
    String value;
    String type;
}
}
}

```

The next class shows an implementation of the room map initialization in method `buildRoomMap()`.

```

package fr.technosens.alertes.backend.sensiNact;

import com.google.gson.Gson;
import com.google.gson.stream.JsonReader;

import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.lang.reflect.Type;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

public final class SNaBackend {

    private Map<String, SNaRoom> roomMap;

    private final String baseUrl;

    public SNaBackend(String baseUrl) {
        this.baseUrl = baseUrl;
    }

    public synchronized String[] loadBuildings() throws IOException {
        List<String> buildingProviderIds = requestGet("aha/sensinact/getBuildingProviderIds",
SNaResponse.ProviderIdsResponse.class);
        return buildingProviderIds.toArray(new String[buildingProviderIds.size()]);
    }

    private <T> T requestGet(String path, Type typeOfResult) throws IOException {
        return request(path, null, typeOfResult);
    }

    private <T> T request(String path, String body, Type typeOfResult) throws IOException {
        HttpURLConnection urlConnection = (HttpURLConnection) new URL(baseUrl +
path).openConnection();
        if(body != null) {
            urlConnection.setDoOutput(true);
            urlConnection.setRequestProperty("Content-Type", "application/json");
            urlConnection.setRequestProperty("Accept", "application/json");
            BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(urlConnection.getOutputStream()));
            writer.write(body);
            writer.flush();
        }
        T response = new Gson().fromJson(new JsonReader(new
InputStreamReader(urlConnection.getInputStream())), typeOfResult);
        if(response == null) throw new IOException("Null response");
        if(response instanceof SNaResponse && !((SNaResponse) response).isSuccess()) throw new
IOException("Error status code: " + ((SNaResponse) response).statusCode);
        if(response instanceof SNaResponse.ResourceResponse && ((SNaResponse.ResourceResponse)
response).response == null) throw new IOException("Response subelement is null");
    }
}

```

```

        return response;
    }

    private synchronized Map<String, SNaRoom> getOrBuildRoomMap() throws IOException {
        if(roomMap == null) buildRoomMap();
        return roomMap;
    }

    private synchronized void buildRoomMap() throws IOException {
        Map<String, SNaRoom> map = new HashMap<>();
        List<String> studioProviderIds = requestGet("aha/sensinact/getStudioProviderIds",
SNaResponse.ProviderIdsResponse.class);
        for(String studioProviderId : studioProviderIds) {
            SNaResponse.ResourceResponse roomResponse = requestGet("sensinact/providers/" +
studioProviderId + "/services/studio/resources/id/GET", SNaResponse.ResourceResponse.class);
            String studioName = roomResponse.response.value;

            SNaResponse.ResourceResponse buildingResponse = requestGet("sensinact/providers/"
+ studioProviderId + "/services/studio/resources/building/GET",
SNaResponse.ResourceResponse.class);
            String buildingId = buildingResponse.response.value;

            SNaResponse.ResourceResponse bedroomResponse = requestGet("sensinact/providers/" +
studioProviderId + "/services/studio/resources/bedroom/GET",
SNaResponse.ResourceResponse.class);
            String bedroomId = bedroomResponse.response.value;
            map.put(bedroomId, new SNaRoom(buildingId, studioProviderId, studioName,
bedroomId, Alert.ROOMTYPE_BEDROOM));

            SNaResponse.ResourceResponse bathroomResponse = requestGet("sensinact/providers/"
+ studioProviderId + "/services/studio/resources/bathroom/GET",
SNaResponse.ResourceResponse.class);
            String bathroomId = bathroomResponse.response.value;
            map.put(bathroomId, new SNaRoom(buildingId, studioProviderId, studioName,
bathroomId, Alert.ROOMTYPE_BATHROOM));
        }
        roomMap = map;
    }
}

```

More java sample source codes are available for download in livelink. The code is provided by the Technosens partner and describes how to use efficiently the sensiNact AHA service API:

[ACTIVAGE > 3 ACTIVAGE PROJECT > 010 Work Packages > 004 WP4 - CERTH - App. Su... > T4.1 > ealertes-backend.zip](#)

The sample source code illustrates how to build the alert dictionary and use it to handle alert triggering and acquitment.

#### 4.2.4.6 Summary of sensiNact development tools provided for ACTIVAGE

The table below summarizes the sensiNact development tools, the generic ones and those developed for AHA services, specified for the ACTIVAGE project needs (in DS6).

Table 16: Summary of the sensiNact ACTIVAGE development tools

	documentation	API doc and swagger	Public source code	Tutorial	Other
sensiNact	<a href="#">sensiNact wiki</a>	Swagger (real data) for live data <a href="#">sensinact.ddns.net/swagger-api/</a> and historic data	<a href="#">Eclipse sensiNact Git repository</a>	<a href="#">sensiNact tutorial section in wiki</a>	<a href="#">sensiNact studio web</a>

		<a href="http://193.48.18.251:8080/swagger-ui.html">193.48.18.251:8080/swagger-ui.html</a>			
<b>AHA sensiNact service API</b>	<a href="#">sensiNact AHA service API shared documentation</a>	<a href="http://193.48.18.245:8081/swagger-api/">193.48.18.245:8081/swagger-api/</a> (mock instance)	To be published	Sequence diagrams and <a href="#">sample source code in livelink</a>	A public sensiNact instance running with mocked AHA functions provides dynamic random values for test purposes

#### 4.2.4.7 Mapping between sensiNact and ACTIVAGE Development Tools

Part of the sensiNact development tools described above correspond to specific ACTIVAGE development tools, described in Section 4.1. This mapping is presented in Table 17. Some of the mapped tools may be used within the AIOTES infrastructure, possibly with some modifications or generalizations. Table 17 also summarizes which tools can be generalized to be used within AIOTES, or are too specific to be included.

Table 17: Mapping between sensiNact and ACTIVAGE development tools

sensiNact development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
sensiNact AHA historical statistics agent	Data/Visual Analytics Tools / Data Analyser	Yes	No
		<b>How?</b>	<b>Why?</b>
		Embedded in the sensiNact ACTIVAGE distribution, computation results are accessible through the sensiNact AHA API and through the AIOTES API	
sensiNact Studio Web	Data/Visual Analytics Tools / Feature/Result viewer	Yes	No
		<b>How?</b>	<b>Why?</b>
		The sensiNact Studio web is a web application used for resources and data monitoring, and can be used as basis for the AIOTES management module.	
Support tools (documentation, wiki, swaggers, source code samples)	Support	Yes	No
		<b>How?</b>	<b>Why?</b>
		The sensiNact support tools (documentation, wiki, swaggers, code samples, etc.) will be used as part of the overall AIOTES support material.	

### 4.2.5 FIWARE

FIWARE<sup>15</sup> is an open middleware platform, driven by the European Union under the Future Internet Public Private Partnership Programme<sup>16</sup>, for the development and global

<sup>15</sup> <https://www.fiware.org/>

deployment of Smart Applications for Future Internet in multiple vertical sectors. Is an open sustainable ecosystem, built around public, royalty-free and implementation-driven software standards.

## 4.2.5.1 Supports Tools

### 4.2.5.1.1 Documentation and WIKI

FIWARE provides an enhanced OpenStack-based cloud environment plus a rich set of open standard APIs that make it easier to connect to the Internet of Things, process and analyze Big data and real-time media or incorporate advanced features for user interaction.

The FIWARE wiki<sup>17</sup> complements the general FIWARE landing page by providing information about:

- Useful resources to get started with FIWARE
- FIWARE Platform documentation
- FIWARE Agile dynamics
- FIWARE Lab resources
- FIWARE Community resources
- Useful Resources for the challenges and hackathons
- Questions

This sections are described in detail below.

#### Useful resources to get started with FIWARE

This section contains a list of links of interest when a developer start using FIWARE.

First of all it is provided a quick FIWARE tour guide<sup>18</sup> focused on the most common instructions to make a programmers familiar with FIWARE. On the other hand, a link to the GitHub FIWARE platform repository is provided. It is also included the FIWARE Catalogue<sup>19</sup> which a rich library of components (Generic Enablers) with reference implementations that allow developers to put into effect functionalities such as the connection to the Internet of Things or Big Data analysis, making programming much easier. All of them are public, royalty-free and open source. Lastly, it is presented the FIWARE Academy, where it can be found training courses, lessons and many other contents regarding FIWARE technology.

#### FIWARE Platform documentation

This category brings information related to FIWARE documentation including:

---

<sup>16</sup> <https://www.fi-ppp.eu/>.

<sup>17</sup> [https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Welcome\\_to\\_the\\_FIWARE\\_Wiki](https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Welcome_to_the_FIWARE_Wiki)

<sup>18</sup> <https://www.fiware.org/developers-entrepreneurs/>

<sup>19</sup> <https://catalogue.fiware.org/>

- A Developer Guidelines where this guide is intended to describe the best practices for FIWARE Platform Development, particularly for those projects which deal with the development of a GEi or accompanying module.
- A summary information about the FIWARE GERis and associated specs in FIWARE Release 6.
- Information about current supported features and roadmap of products working as FIWARE GE reference implementations (GERis).

### FIWARE Agile dynamics

In this section it is presented the FIWARE Agile Development Methodology<sup>20</sup> which elaborates on how Agile principles are being applied in FIWARE and a list of Releases and Sprints of the platform.

### FIWARE Lab resources

Contains information about activities of the FIWARE Lab Chapter as well as general information on the set-up and operation of FIWARE Lab Nodes.

### FIWARE Community resources

This section brings information related to FIWARE Community resources including:

- **FIWARE Technical Steering Committee:** a complementary wiki that complements the general FIWARE Community page by providing information about the FIWARE Technical Steering Committee activities.
- **FIWARE Community Support Chapter:** includes the activities to maintain the FIWARE Catalogue and FIWARE Academy platforms.
- **FIWARE QA Activities:** analyse and assess the level of quality of FIWARE Generic Enablers, from a functional and non-functional point of views.
- **FIWARE Chapter Active Contributors:** a public list of FIWARE active contributors.

### Useful Resources for the challenges and hackathons

This section includes documentation from FIWARE events such as valuable information for developers running the FIWARE hackathon at Campus Party Brazil 2015 in addition to sample applications created in the hackathons.

### Questions

In this category, different facilities to ask for info/help about FIWARE and forward specific requests are provided.

#### 4.2.5.1.2 API doc and swagger

The FIWARE platform has a large number of components with very different APIs, which makes it impossible to explain the details of each one. So as to benefit from FIWARE

---

<sup>20</sup> [https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE\\_Agile\\_Development\\_Methodology](https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Agile_Development_Methodology)

automatic documentation generation systems developers must use markdown for documentation and Apiary Blue Print for API specification. Usually these use RESTful APIs where the content format is JSON. There are also programming interfaces in Java for some of the components.

FIWARE APIs share the following characteristics:

- They are RESTful web services.
- Each HTTP request in a FI-WARE RESTful API may require the inclusion of specific authentication credentials
- Resource representation is transmitted between client and server by using HTTP 1.1 protocol.
- They may support XML or JSON as representation format for request and response parameters. Each API specification indicates which of them is the default format.

More information about the different APIs that make up the different FIWARE services can be found in the link provided below:

Summary of FIWARE API Open Specifications is available on [https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Summary\\_of\\_FIWARE\\_API\\_Open\\_Specifications](https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Summary_of_FIWARE_API_Open_Specifications).

The common aspects in FI-WARE Open Restful API Specifications can be found under [plugins/mediawiki/wiki/fiware/index.php/Common\\_aspects\\_in\\_FI-WARE\\_Open\\_Restful\\_API\\_Specifications](https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Common_aspects_in_FI-WARE_Open_Restful_API_Specifications).

Additionally, each GE can be accessed through its own API (normally REST), for example [http://fiware-orion.readthedocs.io/en/master/user/walkthrough\\_apiv2/index.html](http://fiware-orion.readthedocs.io/en/master/user/walkthrough_apiv2/index.html).

Regarding the Swagger specification, for that purpose Apiary Blue Print was selected. There isn't any plan to develop a Swagger specification.

### 4.2.5.1.3 Source code public access

The FIWARE Community is not only formed by contributors to the technology (the FIWARE platform) but also those who contribute in building the FIWARE ecosystem and making it sustainable over time. As such, individuals and organizations committing relevant resources in FIWARE Lab activities or activities of the FIWARE Accelerator, FIWARE mundus or FIWARE iHubs programmers are also considered members of the FIWARE community.

Source code of FIWARE is accessible through the GitHub Repositories:

<https://github.com/Fiware>

### 4.2.5.1.4 FIWARE tutorial and sample source code

There is not a unique tutorial to start using FIWARE. Since FIWARE is composed by several components, a specific guide for each one is needed which makes it impossible to detail each one. In the general FIWARE landing page, developers can find a section called quick FIWARE tour guides<sup>21</sup> dedicated to the gathering all the getting starting guides of the FIWARE components. Table 18 lists all the guides along with a short description.

---

<sup>21</sup> <https://www.fiware.org/developers-entrepreneurs/>

Table 18: FIWARE existing tour guides

TOUR GUIDE	DESCRIPTION
<b>Development of Context-Aware Applications using FIWARE</b>	Orion Context Broker allows you to model, manage and gather context information at large scale enabling context-aware applications.
<b>Real time processing of Context Events</b>	Proton Complex Event Processing (CEP) analyses events, e.g. updates on context, in real-time to detect scenarios where actions have to be triggered or new events are created.
<b>Publication of Context Information as Open Data</b>	FIWARE incorporates CKAN as part of its architecture for open data. CKAN extensions allow to manage access not only to static historic data but real-time context information, as well as the management of access control rights.
<b>Creating Application Dashboards</b>	Wirecloud is a web mashup platform aimed at empowering end users, without programming skills, to easily create fully-fledged application dashboards built up from widgets, operators and other pre-existing mashups.
<b>Providing an Advanced User Experience (UX)</b>	These components allow you to incorporate advanced features in your web-based user interface, such as Augmented Reality or 3D visualization.
<b>Connection to the Internet of Things</b>	IDAS IoT Agents allow your application to easily gather context information from sensors or actuate upon physical objects.
<b>Handling Authorization &amp; Access Control to APIS</b>	FIWARE brings a powerful framework that will allow you to setup Authorization and Access Control policies based on widely adopted Security standards (OAuth, XACML).
<b>Big Data Analysis of Historic Context Information</b>	Cygnus allows you to inject historic context information records into an HDFS based storage. BigData analysis or advanced queries can then be performed over historic data.
<b>Real time processing of Media Streams</b>	Kurento allows you to process, in real-time, multimedia information so that you can incorporate into your application extended sensing capabilities based on cameras or microphones (detecting faces, crowds, plates,...)
<b>Hosting your Application on a FIWARE Cloud</b>	The FIWARE Cloud is an Infrastructure as a Service platform based on OpenStack, comprising the Compute (Nova), Storage (Cinder), Network (Neutron) and Image (Glance) services. All the application components running in a centralized data center can be provisioned and managed using the FIWARE Cloud capabilities.

Furthermore, in the FIWARE Academy<sup>22</sup> developers have at their disposal a large number of courses to training on the use of FIWARE Enablers.

#### 4.2.5.2 Tools supported by FIWARE

The FIWARE Platform comprises a set of building blocks that ease creation of smart Internet Applications. These technological blocks, called Generic Enablers (GEs), allow developers

<sup>22</sup> <http://edu.fiware.org/>

to put into effect functionalities such as the connection to the Internet of Things or Big Data analysis. All of them are public, royalty-free and open source.

Generic Enablers consists of library of components with reference implementations that provides developers a wide range of general-purpose functions to make programming easier. These functions, offered through well-defined APIs, facilitate the development of smart applications in multiple sectors such as those detailed below. They divided into 7 technical chapters (see FIWARE catalogue<sup>23</sup>):

- Data/Context Management
- Internet of Things (IoT) Services Enablement
- Advanced Web-based User Interface
- Security
- Interface to Networks and Devices (I2ND)
- Applications, Services and Data Delivery
- Cloud Hosting

The Reference Architecture, associated to each FIWARE chapter, can be instantiated into a concrete architecture by means of selecting and integrating products implementing the corresponding FIWARE GEs (i.e., products which are compliant with the corresponding FIWARE GE Open Specifications). However, the description of the Reference Architecture associated to a chapter does not depend on how FIWARE GEs in that chapter can be implemented. Any implementation of a FIWARE GE (also referred as FIWARE GEi) will be, by nature, replaceable.

The FIWARE platform has a large number of Generic Enablers and each GE has its own APIs (normally REST) to access them. This fact makes impossible to explain the details of each one. Therefore, this document doesn't include information of each one. Thereby, it must be pointed out that all the information of each GE can be found inside of its corresponding section of the FIWARE Catalogue<sup>23</sup>. Inside of each FIWARE GE section, it can be found an overview, general documentation, available downloads, Open API Specification of each GE and so on. It should be noted that Generic Enablers provide open interfaces for both, to Application Developers (APIs) in addition to support interoperability with other GEs.

All these tools are inherent to the FIWARE platform and participate in the creation of the platform itself, given that, as mentioned above, the architecture of FIWARE is implemented through these products.

In other words, all these tools participate in the deployment and expansion of the platform itself but are not directly related to any other element of the ACTIVAGE Framework. Thereby, It is important to highlight that these tools are PLATFORM SPECIFIC and cannot be generalized to AIOTES.

The Orion Context Broker is an implementation of the Publish/Subscribe Context Broker GE, providing the NGSI9 and NGSI10 interfaces.

Orion Context Broker allows to manage all the whole lifecycle of context information including updates, queries, registrations and subscriptions. Using the Orion Context Broker, you are able to register context elements and manage them through updates and queries. In

---

<sup>23</sup> <http://catalogue.fiware.org/>



addition, you can subscribe to context information so when some condition occurs (e.g. a context element has changed) you receive a notification.

Using these interfaces, clients can do several operations:

- Register context producer applications, e.g. a temperature sensor within a room
- Update context information, e.g. send updates of temperature
- Being notified when changes on context information take place (e.g. the temperature has changed) or with a given frequency (e.g. get the temperature each minute)
- Query context information. The Orion Context Broker stores context information updated from applications, so queries are resolved based on that information.

Once said that FIWARE has numerous GEs and presented the Orion Context Broker, the following will describe those that are more remarkable for ACTIVAGE's eco-system.

### Short Time Historic

The Short Time Historic (STH, aka. Comet) is a component of the FIWARE ecosystem in charge of managing (storing and retrieving) historical raw and aggregated time series information about the evolution in time of context data (i.e., entity attribute values) registered in an Orion Context Broker instance.

All the communications between the STH and the Orion Context Broker as well as between the STH and any third party (typically for data retrieval) use standardized NGSI9 and NGSI10 interfaces.

### Connector Framework (Cygnus)

Cygnus is a connector in charge of persisting certain sources of data in certain configured third-party storages, creating a historical view of such data.

Internally, Cygnus is based on Apache Flume, a technology addressing the design and execution of data collection and persistence agents. An agent is basically composed of a listener or source in charge of receiving the data, a channel where the source puts the data once it has been transformed into a Flume event, and a sink, which takes Flume events from the channel in order to persist the data within its body into a third-party storage.

Cygnus is designed to run a specific Flume agent per source of data.

Current stable release is able to persist the following sources of data in the following third-party storages:

- NGSI-like context data in:
  - HDFS, the Hadoop distributed file system.
  - MySQL, the well-know relational database manager.
  - CKAN, an Open Data platform.
  - MongoDB, the NoSQL document-oriented database.
  - FIWARE Comet (STH), a Short-Term Historic database built on top of MongoDB.

### Complex Event Processing - Proactive Technology Online

The CEP GE analyses event data in real-time, generates immediate insight and enables instant response to changing conditions. While standard reactive applications are based on reactions to single events, the CEP GE reacts to situations rather than to single events. A situation is a condition that is based on a series of events that have occurred within a

dynamic time window called processing context. Situations include composite events (e.g., sequence), counting operators on events (e.g., aggregation) and absence operators. The Proactive Technology Online is an implementation of the FIWARE CEP (Complex Event Processing) GE. The Proactive Technology Online is a scalable integrated platform to support the development, deployment, and maintenance of event-driven applications. The Proactive Technology Online authoring tool allows the definition of CEP applications using a web user interface. The Proactive Technology Online engine is a runtime tool that receives information on the occurrence of events from event producers, detects situations, and reports the detected situations to external consumers.

The technology and implementations of CEP provide means to expressively and flexibly define and maintain the event processing logic of the application, and in runtime it is designed to meet all the functional and non-functional requirements without taking a toll on the application performance, removing one issue from the application developer’s and system managers concerns.

### 4.2.5.3 Mapping between FIWARE and ACTIVAGE development tools

Part of the FIWARE development tools, described through Section 0, can be used within the AIOTES infrastructure as ACTIVAGE development tools (Section 4.1). This mapping is presented in Table 19 where it is specified if a tool can be generalized to be used within AIOTES and how.

Table 19: Mapping between IoTivity and ACTIVAGE development tools.

FIWARE development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
Short Time Historic	Not mapped with AIOTES		
Connector Framework (Cygnus)	Not mapped with AIOTES		
Complex Event Processing	Not mapped with AIOTES		
<a href="#">Support tools</a> (documentation, wiki, source code samples)	<a href="#">Support</a>	Yes	No
		<b>How?</b> The FIWARE support tools (documentation, wiki, code samples, etc.) will be used as part of the overall AIOTES support material.	<b>Why?</b>
<a href="#">Source code samples</a>	IDE / <a href="#">Code generator</a> IDE / <a href="#">Source code templates</a>	Yes	No
		<b>How?</b> FIWARE source code samples can be used to design source code templates for FIWARE, which will be used as part of the code generator and code templates ACTIVAGE development tools.	<b>Why?</b>

## 4.2.6 IoTivity

The IoTivity platform is an open source project sponsored by the [Open Connectivity Foundation](#) (OCF). IoTivity code contributions are shared under the [Apache 2.0 license](#).

### 4.2.6.1 Support tools

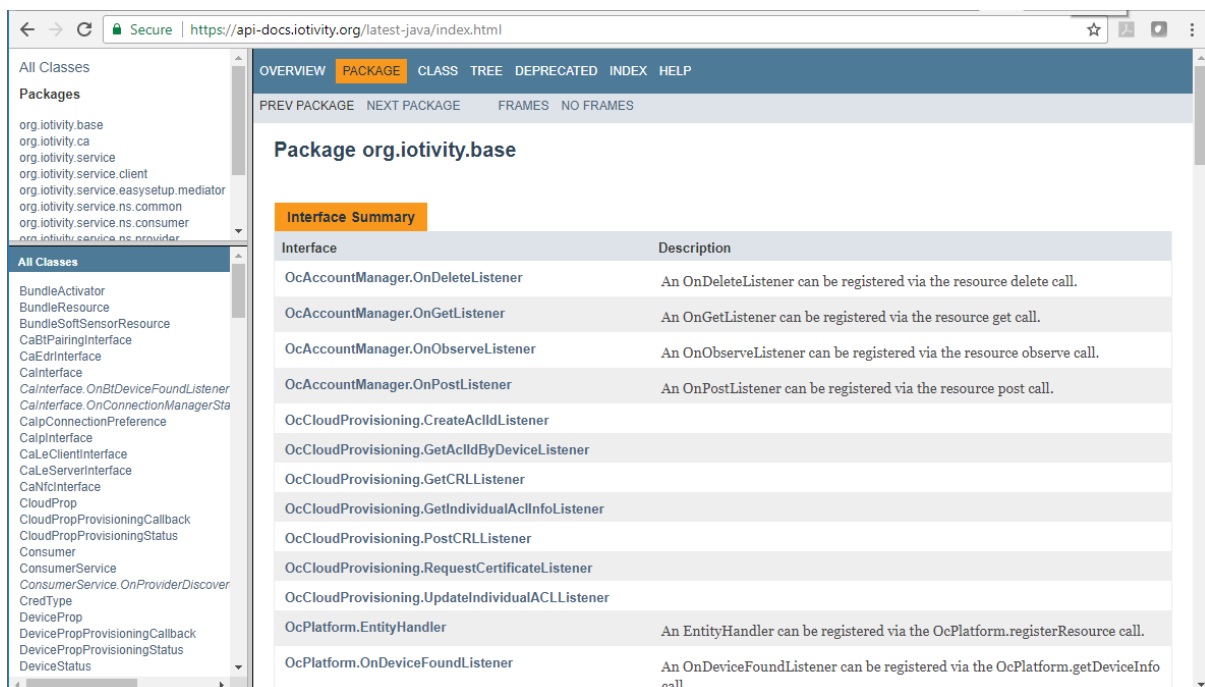
#### 4.2.6.1.1 Documentation

IoTivity offers a [wiki](#) that provides detailed documentation and tutorials [4] about setting-up and using it. The main sections of this wiki are the following:

- Getting Set up to Develop (Guidelines, how-to etc)
- Programming Guide
- Technical Notes
- Concepts (keywords)
- Release Management Function
- QA Function
- Frequently Asked Questions

IoTivity also offers detailed documentation about the IoTivity API, for different programming languages.

#### 4.2.6.1.2 APIs



The screenshot shows a web browser displaying the IoTivity Java API documentation. The URL is <https://api-docs.iotivity.org/latest-java/index.html>. The page title is "Package org.iotivity.base". The main content area is titled "Interface Summary" and contains a table with the following data:

Interface	Description
OcAccountManager.OnDeleteListener	An OnDeleteListener can be registered via the resource delete call.
OcAccountManager.OnGetListener	An OnGetListener can be registered via the resource get call.
OcAccountManager.OnObserveListener	An OnObserveListener can be registered via the resource observe call.
OcAccountManager.OnPostListener	An OnPostListener can be registered via the resource post call.
OcCloudProvisioning.CreateAclIdListener	
OcCloudProvisioning.GetAclIdByDeviceListener	
OcCloudProvisioning.GetCRLListener	
OcCloudProvisioning.GetIndividualAclInfoListener	
OcCloudProvisioning.PostCRLListener	
OcCloudProvisioning.RequestCertificateListener	
OcCloudProvisioning.UpdateIndividualACLListener	
OcPlatform.EntityHandler	An EntityHandler can be registered via the OcPlatform.registerResource call.
OcPlatform.OnDeviceFoundListener	An OnDeviceFoundListener can be registered via the OcPlatform.getDeviceInfo call.

Figure 55: IoTivity Java API documentation.

IoTivity offers an API and detailed documentation for the following supported programming languages:

- C <<https://api-docs.iotivity.org/latest-c/>>

- C++ <<https://api-docs.iotivity.org/latest/index.html>>
- Java <<https://api-docs.iotivity.org/latest-java/index.html>>

### 4.2.6.1.3 Source code samples

A majority of tutorials including sample source code covering a variety of aspects and functionalities of IoTivity are available through the official wiki of IoTivity: <https://wiki.iotivity.org/>

Various samples are also included in for each version of IoTivity: <https://github.com/iotivity/iotivity/tree/master/resource/examples>

The most indicative samples in C++ are the following:

- [devicediscoveryclient](#): This sample demonstrates the device discovery feature. The client queries for the device related information stored by the server.
- [devicediscoveryserver](#): This sample demonstrates platform and device discovery feature. The server sets the platform and device related info, which can be later retrieved by a client.
- [fridgeclient](#): This fridgeclient represents a client trying to discover the associated fridgeserver. The device resource is the only one available for discovery on the server, so we have to take the fact that we know the device tag to then generate a Resource object.
- [fridgeserver](#): The purpose of this server is to simulate a refrigerator that contains a device resource for its description, a light resource for the internal light, and 2 door resources for the purpose of representing the doors attached to this fridge. This is used by the fridgeclient to demonstrate using `std::bind` to attach to instances of a class as well as using `constructResourceObject`.
- [garageclient](#): `garageclient.cpp` is the client program for `garageserver.cpp`, which uses different representation in `OCRepresentation`.
- [garageserver](#): This sample describes how to use various JSON types in the representation.
- [lightserver](#): This sample provides steps to define an interface for a resource (properties and methods) and host this resource on the server.
- [mediaserver](#): This sample provides steps to define an interface for a resource (properties and methods) and host this resource on the server.
- [presenceclient](#): A client example for presence notification.
- [presenceserver](#): This sample provides steps to define an interface for a resource (properties and methods) and host this resource on the server.
- [roomclient](#): It defines the entry point for the console application.
- [roomserver](#): This sample shows how one could create a resource (collection) with children.
- [simpleclient](#): This sample is the starting point for understanding the main functionalities of an IoTivity client.
- [simpleclientserver](#): This sample provides steps to define an interface for a resource (properties and methods) and host this resource on the server. Additionally, it'll have a client example to discover it as well.
- [simpleserver](#): This sample is the starting point for understanding the main functionalities of an IoTivity server.

- [threadingsample](#): This sample demonstrates : running one server in main thread, another server in a separate thread, and running 2 clients in each thread.

There are also similar examples in Java

<https://github.com/iotivity/iotivity/tree/master/java/examples-java>, Java Android <https://github.com/iotivity/iotivity/tree/master/java/examples-android> and javascript <https://github.com/otcshare/iotivity-node/tree/master/js>.

The source code for IoTivity itself is also available. The source code of all versions of IoTivity is publicly available through the following page: <https://www.iotivity.org/downloads>

The master Git location for IoTivity projects is gated by an instance of the [Gerrit reviewing system](#), such that pushing a change in Git is intercepted by Gerrit and presented as a review page. The process of setting up and using Gerrit for IoTivity is documented in a pair of wiki pages:

- [https://wiki.iotivity.org/how\\_to\\_use\\_gerrit](https://wiki.iotivity.org/how_to_use_gerrit)
- [https://wiki.iotivity.org/submitting\\_to\\_gerrit](https://wiki.iotivity.org/submitting_to_gerrit)

Unreleased code and latest contributions are also available in a Git repository: <https://gerrit.iotivity.org/gerrit/gitweb?p=iotivity.git;a=summary>

Another mirror is provided in GitHub for easy forking: <https://github.com/iotivity/iotivity>

There are many ways to get involved in the IoTivity community, and not all involve contributing code. Ways to get involved to IoTivity Community are by submitting patches, reviewing submission by others, subscribing to mailing lists and participating in active channels of the platform.

#### 4.2.6.14 Tutorials

IoTivity Tutorials are included in the IoTivity wiki <[https://wiki.iotivity.org/iotivity\\_simulator](https://wiki.iotivity.org/iotivity_simulator)>, covering all aspects mentioned above, in Section 4.2.6.1.1. IoTivity tutorials also contain sample source code, as presented in Section 0.

### 4.2.6.2 Tools useful in the ACTIVAGE context

#### 4.2.6.2.1 CoAP - HTTP Proxy

[CoAP - HTTP Proxy](#) is a tool provided in order to build the connection between the Web and the Web of Things by allowing CoAP clients to interact with resources from HTTP servers.

#### 4.2.6.2.2 Cloud connection tools

IoTivity provides [cloud connection tools](#), which consist of four components:

- [Resource-directory server](#): It is responsible for registering the IoTivity Resources on Cloud, by keeping records on a MongoDB database.
- Account server: This server handles the authentication the IoTivity Server/Client that connect to the cloud in order to register their resources.
- [MessageQueue server](#): It is responsible for handling creation, publishing and subscription to topics, by using Apache Kafka messaging system.
- Interface server: It is the interface that forwards the requests from IoTivity Server to the other three components of the IoTivity Cloud through Coap over TCP.

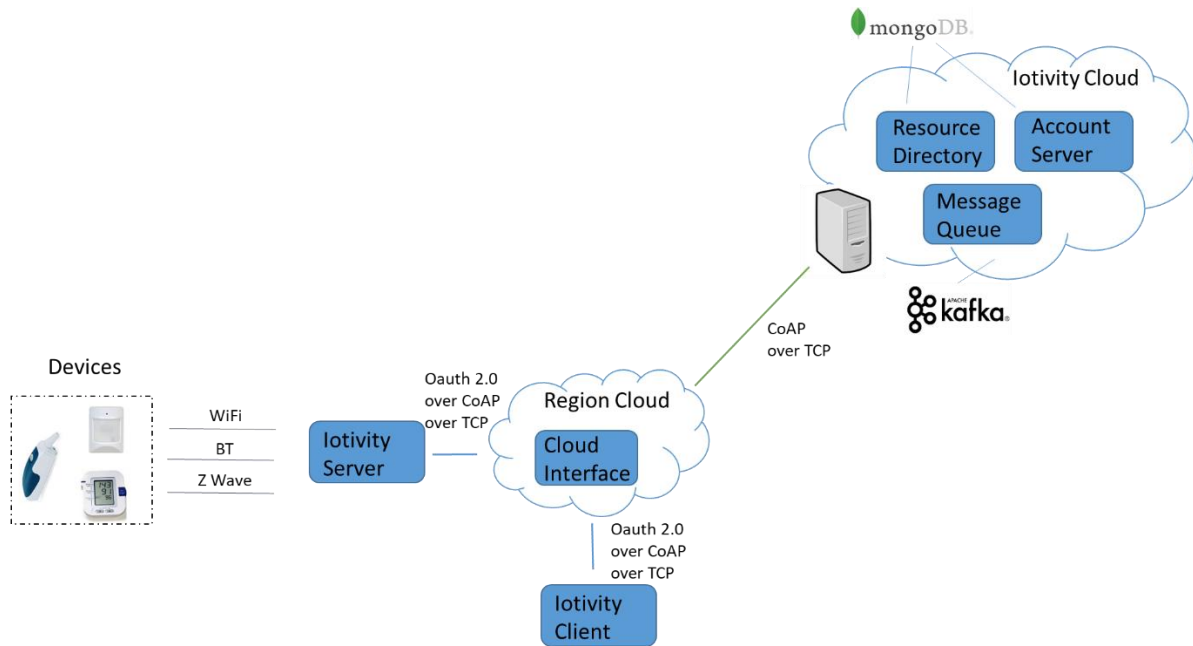


Figure 56: A view of the IoTivity architecture with Cloud functionality

In order to deploy IoTivity Cloud, it is required to build and deploy more than one component. Moreover, the cloud components have dependencies such as the Apache Kafka, the Zookeeper and the MongoDB. In case of a production or a test deployment, it would require to install all dependencies on the machine or in case of a multi-node system it would be needed to define what will run on which machine without orchestration possibilities. In order to automate the process and easily deal with these demands the IoTivity was dockerized<sup>24</sup>. The docker compose file contains all required services for running the whole IoTivity cloud stack and it is composed from:

- Apache Kafka and Zookeeper <<https://hub.docker.com/r/spotify/kafka/>>
- Mongo DB <[https://hub.docker.com/\\_/mongo/](https://hub.docker.com/_/mongo/)>
- IoTivity Interface <<https://hub.docker.com/r/iotivity/interface/>>
- IoTivity Message Queue <<https://hub.docker.com/r/iotivity/messagequeue/>>
- IoTivity Account Server <<https://hub.docker.com/r/iotivity/accountserver/>>
- IoTivity Resource Directory <<https://hub.docker.com/r/iotivity/resourcedirectory/>>

#### 4.2.6.2.3 OneloTa OCF design tool

IoTivity is a reference implementation of OCF specification. OCF define the connectivity requirements to improve interoperability between the billions of devices making up the Internet of Things (IoT). This enables the devices to communicate in a similar way specifying the same properties between the devices.

OCF specification defines a set of core Device Types and their required Resource Types. A Resource is the minimal interoperable component in OCF. It has a URI and a collection of Properties.

<sup>24</sup> <https://hub.docker.com/u/iotivity/>

Creating data models (resource models) for OCF IoT Devices is done by oneloTa<sup>25</sup> which is an open web-based tool created by the Open Connectivity Foundation (OCF) to encourage the design of interoperable device data models for the Internet of Things. Only a certain number of devices are defined by OCF specification and for each there is only one core definition. Derived data models can provide alternative device definitions, but they must be unambiguously tied to a core definition. This allows new devices to be quickly built using existing devices where possible, and guarantees interoperability between devices that use other data models already in the database. The created resource needs to pass an approval process in order to be made available in the oneloTa repository.

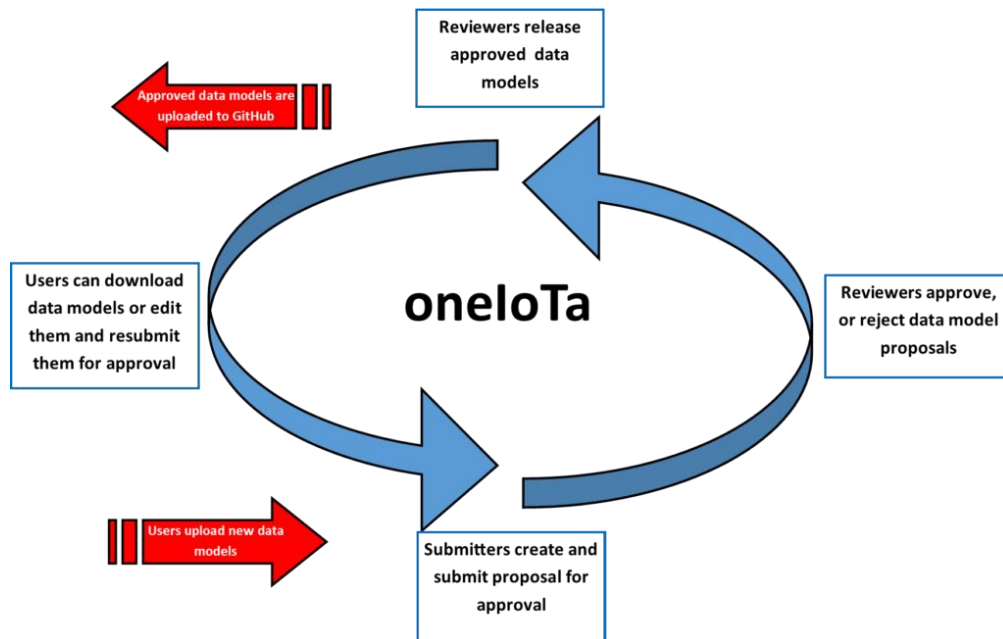


Figure 57: oneloTa data models development process.

#### 4.2.6.2.4 Provisioning Manager

Provisioning Manager (PM)<sup>26</sup> is a tool of IoTivity that could act as a security administrator of IoT devices in its IP subnet. Provisioning Manager has two major roles: Ownership Transferring and Security Management of owned devices.

When new device is introduced in the IP subnet, Provisioning Manager takes the ownership of the new device and provisions security information such as credential and access control policy to manage new device securely. If PM doesn't take ownership and provide proper security policy to the newly introduced device in its IP subnet, the new device might be under control of unwanted subjects and perform undesirable operations such as turning on the light during midnight and ignoring user's commands. All security functionality operate using CBOR data (.dat files).

#### 4.2.6.2.5 IoTivity simulator

IoTivity provides a tool <[https://wiki.iotivity.org/iotivity\\_simulator](https://wiki.iotivity.org/iotivity_simulator)> that can simulate 1) OCF resources and 2) the functionality of an OCF client, which aims to help developers with testing during development and before purchasing the real hardware. The tool is written in

<sup>25</sup> <https://openconnectivity.org/developer/oneiota-data-model-tool>

<sup>26</sup> <https://wiki.iotivity.org/provisioning>

Java as an Eclipse plugin and provides two perspectives. However, in ACTIVAGE physical hardware will be deployed and IoTivity Simulator will only be used for testing reasons.

OCF resources can be simulated by using Resource model definition (RAML) files or created by using GUI wizards. The simulated resources are able to handle requests that are received and send appropriate responses to clients. When the simulator resource server receives any GET/PUT/POST/OBSERVE requests, “Service Provider Perspective” shows the log messages with the request information and sends appropriate responses as shown in figure below.

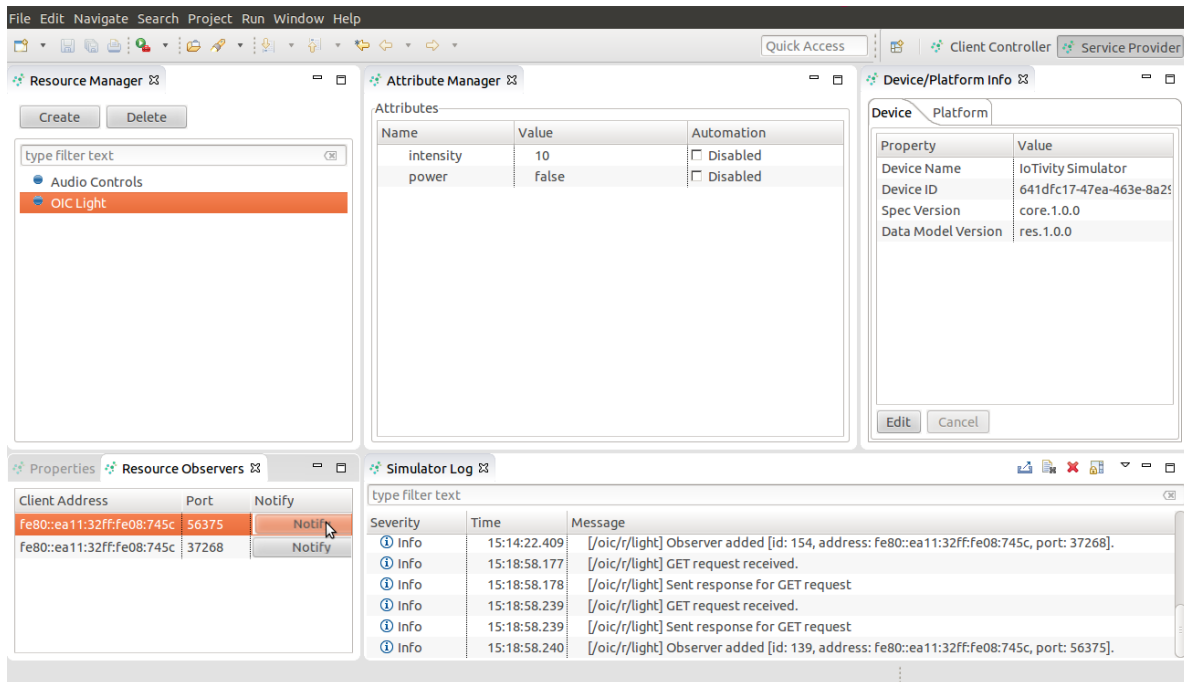


Figure 58: Service Provider Perspective of IoTivity Simulator

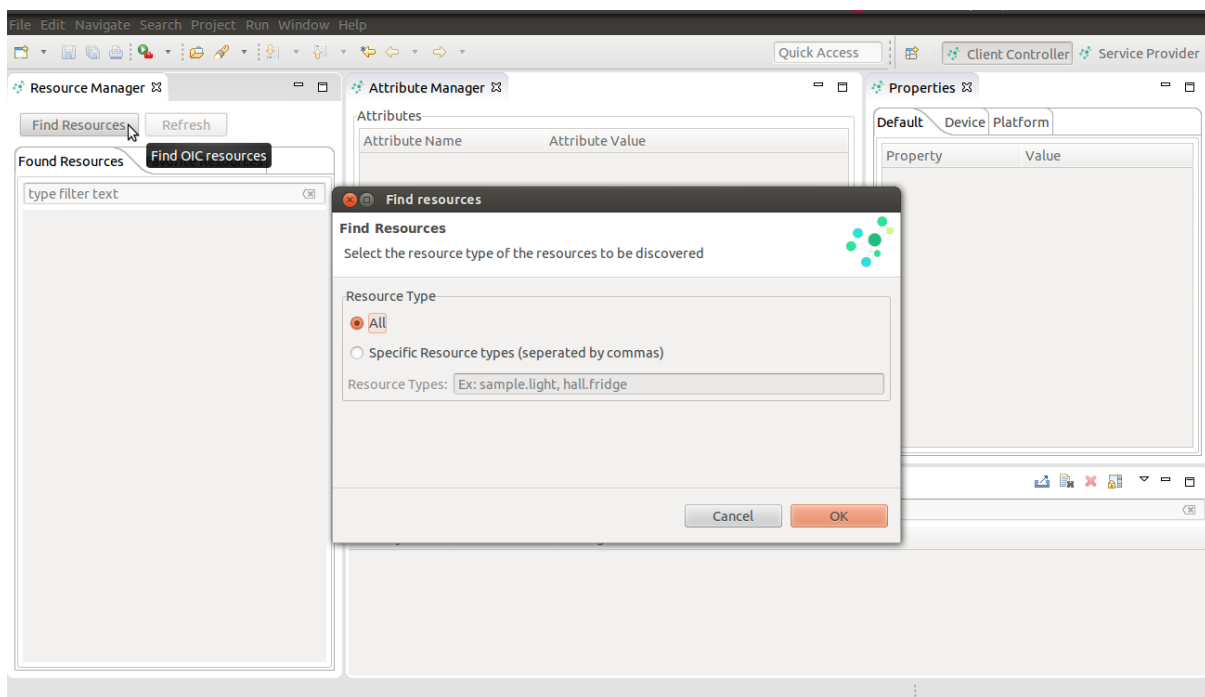


Figure 59: Client Controller Perspective of IoTivity Simulator.



The OCF client can also be simulated. It has the following functionalities: find resources of certain types in the given network, provides support for observing resource changes and provides support for sending automatic requests (GET/PUT/POST) to remote resources with the help of remote resource RAML file. Below screenshot shows the “Client Controller Perspective” view.

### 4.2.6.3 Mapping between IoTivity and ACTIVAGE development tools

Part of the IoTivity development tools described above correspond to specific ACTIVAGE development tools, described in Section 4.1. This mapping is presented in Table 20. Some of the mapped tools may be used within the AIOTES infrastructure, possibly with some modifications or generalizations. Table 20 also summarizes which tools can be generalized to be used within AIOTES, or are too specific to be included.

Table 20: Mapping between IoTivity and ACTIVAGE development tools.

IoTivity development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
<a href="#">CoAP - HTTP Proxy</a>	Not mapped with AIOTES		
<a href="#">Cloud connection tools</a>	Data Lake tools / <a href="#">ACTIVAGE data model workbench</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		The functionalities of the IoTivity cloud connection tools will form a conceptual basis for the implementation of the cloud-based Data Lake functionalities.	
<a href="#">oneloTa OCF design tool</a>	Semantic Interoperability Layer tools / <a href="#">ACTIVAGE Ontology Explorer</a> & <a href="#">Device semantics editor</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		The general idea of the oneloTa OCF design tool is directly relevant to the ACTIVAGE ontology-related development tools. The design, source code and interface of oneloTa can be used as reference for the design of the ontology explorer and device semantics editor tools in AIOTES.	
<a href="#">Provisioning manager</a>	Not mapped with AIOTES		
<a href="#">IoTivity simulator</a>	Not mapped with AIOTES		
<a href="#">Support tools</a> (documentation, wiki, source code samples)	<a href="#">Support</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		The IoTivity support tools (documentation, wiki, code samples, etc.) will be used as part of the overall AIOTES support material.	
<a href="#">Source code samples</a>	IDE / <a href="#">Code generator</a> IDE / <a href="#">Source code templates</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		IoTivity source code samples can be used to design source code templates for IoTivity, which will be used as part of the code generator and code templates ACTIVAGE development tools.	

## 4.2.7 SeniorSome

SeniorSome development tools can be used for Activage but for reuse of tools in AIOTES the main tools are the API:s that are available in the gateway level/Bridge level and in the cloud-based REST interface. The tools are mainly industry standard tools that can be used due to open interfaces provided as API:s for developers in Activage.

The SeniorSome API's can be used in Services layer: development, deployment, analytics, data, with AIOTES API and for Semantic interoperability layer: broker. The SeniorSome development tools set is based the usage of API definitions, examples and descriptions. This API set can be used as a part of AIOTES development tools where needed.

The primary tools that are used are tools like Android studio and Eclipse so any development project format shall follow the form used in these solutions.

The mapping of the tool to AIOTES and Activage the Section 4.2.7.6 shall provide this information. As a reference the tools are also explained in the following list:

### 4.2.7.1 Support

**Support:** (Tools for providing documentation and instructions about using the AIOTES development tools.)

- The SeniorSome API documents in <https://api.seniorsome.net>. Including information about how to use and with short examples.

### 4.2.7.2 Integrated Development Environment

**Integrated Development Environment (IDE):** (Tools for facilitating the creation of new applications.)

- SeniorSome can be developed for example with Android Studio, Eclipse and others like Swift-tools. The scripting parts can be developed in an environment that the coder chooses. Examples are provided only in one like Android Studio.
- As a higher level tool the SeniorSome Backend can be used for service creation/development.

### 4.2.7.3 Data- and visual analytics tools

**Data and visual analytics tools:** (Tools for facilitating the introduction of data analytics and visual analytics in an application.)

- SeniorSome dashboard/analytics-tool is available for the SeniorSome service. the dashboard can be used through the API:s.

### 4.2.7.4 Data Lake tools

**Data Lake tools:** (Tools for facilitating access to the data available through the Data Lake.)

- For the Data Lake seniorSome offers an API connection to SeniorSome stored data.

### 4.2.7.5 Semantic Interoperability Layer

**Semantic Interoperability Layer (SIL) tools:** Tools for facilitating access to the Semantic Interoperability Layer ontologies.

- SeniorSome API:s can be used for semantic interoperability.
- The Seniorsome database service can be used for storing rules/rulesbases for ontologies and mappings.

### 4.2.7.6 Mapping between SeniorSome and ACTIVAGE Development Tools

Part of the SeniorSome development tools described above correspond to specific ACTIVAGE development tools, described in Section 4.1. This mapping is presented in Table 21. Some of the mapped tools may be used within the AIOTES infrastructure, possibly with some modifications or generalizations. Table 21 also summarizes which tools can be generalized to be used within AIOTES, or are too specific to be included.

Table 21: Mapping between SeniorSome and ACTIVAGE development tools.

SeniorSome development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
SeniorSome API:s	Support, Bridge, Data/Visual Analytics tools /Data Analyser, Semantic Interoperability layer	Yes	No
		<b>How?</b>	<b>Why?</b>
		By using the API definition and adding this to the Activage tools and development process.	
SeniorSome Backend	Data/Visual Analytics Tools / Feature/Result viewer / Device control, Semantic Interoperability layer	Yes	No
		<b>How?</b>	<b>Why?</b>
		through the API definitions and/or Bridged connectivity. Partly can be used as a ui/interface for the AIOTES where needed.	
Documentation	Support, API development, Semantic Interoperability Layer tools /	Yes	No
		<b>How?</b>	<b>Why?</b>
		As a support means for AIOTES compliant development.	
Source Code & Binaries	Support	Yes	No
		<b>How?</b>	<b>Why?</b>
		The SeniorSome Documentation can be used as part of the overall AIOTES support material.	

### 4.2.8 Summary of existing tools

This section presents a synthesized set of tables providing a quick overview of the development tools for each of the platform designated to be interoperable with AIOTES framework. This section main purpose is to provide an interested developer with a vision of the different tools provided regarding the platform and, at the same time, serve as a listing of the desirable functionality over the AIOTES framework.

#### 4.2.8.1 Platform-specific development tools comparison

General information about each platform is depicted in Table 22. Aspects like the main programming language, license or where to find related documentation with the platform might serve an interested user to deploy one or another platform

Table 22: High level platform overview

Platform	Programming	Open	License	API documentation	Wiki documentation
----------	-------------	------	---------	-------------------	--------------------

	language	source			
universAAL	Java (plain, OSGi, Android) JavaScript	Yes	Apache 2.0	Yes <a href="https://github.com/universAAL">https://github.com/universAAL</a>	- Yes <a href="https://github.com/universAAL/platform/wiki">https://github.com/universAAL/platform/wiki</a>
SOFIA2	Java,C, Javascript (Multilanguage)	Yes	Apache 2.0	Yes <a href="http://sofia2.com/desarrollador_en.html#documentacion">http://sofia2.com/desarrollador_en.html#documentacion</a>	- Yes <a href="http://sofia2.com/desarrollador_en.html#documentacion">http://sofia2.com/desarrollador_en.html#documentacion</a>
OpenIoT	Java	Yes	LGPL V3.0	Yes <a href="https://github.com/OpenIoTOrg/openiot/wiki/Documentation">https://github.com/OpenIoTOrg/openiot/wiki/Documentation</a>	- Yes <a href="https://github.com/OpenIoTOrg/openiot/wiki">https://github.com/OpenIoTOrg/openiot/wiki</a>
SensiNact	Java	Yes	Eclipse License v1.0	Yes <a href="http://sensinact.ddns.net/swagger-api/index.html#/">http://sensinact.ddns.net/swagger-api/index.html#/</a>	- Yes - <a href="http://wiki.eclipse.org/SensiNact">http://wiki.eclipse.org/SensiNact</a>
FIWARE	Multilanguage	Yes	Apache 2.0	Yes <a href="http://www.fiware.org/developers/">http://www.fiware.org/developers/</a>	- Yes <a href="https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Main_Page">https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Main_Page</a>
IoTivity	C, C++, Java	Yes	Apache 2.0	Yes <a href="https://www.iotivity.org/documentacion">https://www.iotivity.org/documentacion</a>	- Yes <a href="https://wiki.iotivity.org/iotivity_simulator">https://wiki.iotivity.org/iotivity_simulator</a>
Senior Some	Not available	Not Available	Not available	Not available	Not available

Table 23 represents the way in which a developer can build and compile its code over a specific platform. Information about the IDE and its functionalities are shown.

Table 23: Development over platform

Platform	Provides IDE	IDE	How the IDE is provided?	What functionalities are offered by the IDE?
universAAL	Yes	AAL Studio over Eclipse	via plugins	<ul style="list-style-type: none"> <li>– wizards for creating projects,</li> <li>– build tools for simplifying building and launching of applications,</li> <li>– modeling and transformation tools for making the development more efficient.</li> </ul>
SOFIA2	Yes	Eclipse	via plugins	<ul style="list-style-type: none"> <li>– ontologies management,</li> <li>– KPs/APPs management,</li> <li>– token management ,</li> <li>– rules management,</li> <li>– predefined queries management,</li> <li>– send SAPP messages,</li> <li>– API manager</li> </ul>
OpenIoT	Yes	Eclipse	via plugins	<ul style="list-style-type: none"> <li>– API for logging in and out,</li> <li>– Token validity check</li> <li>– Access control utility methods</li> </ul>
SensiNact	Yes	SensiNact Studio over Eclipse	Uses Eclipse tools to perform continuous integration	Eclipse Jenkins runs a compilation of the sensiNact Gateway every day, the resulting compilation are kept in the download area of Eclipse

FIWARE	No	-	-	-
IoTivity	Yes	Eclipse	via plugins	Enables the simulation of services, OCF resources and OCF clients
SeniorSome	Not available	Not Available	Not available	Not available

Table 24 represents the existing tools to development helping. The tools are classified in different types: samples of code, dependency manager, code generator, build manager, service composition builder or GUI Tool for management and configuration.

Table 24: Development helping tools

Platform	Samples of code	Dependency manager	Code generator	Build manager	Service Composition Builder	GUI Tool for management and configuration
universAAL	Yes	Yes – Maven	Yes -AAL Studio plugins	Pax Runner osgim, Maven	Yes, SPARQL SPARQL Tester	Yes, AAL Space Monitor
SOFIA2	Yes	Yes - Maven	Yes	Not specified	Yes	Provides visualization tools but not specified whether the requirements are matched
OpenIoT	Yes	Not specified	Yes – Request definition tool	Yes, Request definition tool	Yes, Request definition tool	Yes, Request definition tool
SensiNact	Yes	Uses Maven	Yes	Uses Maven and Jenkins	Yes – SensiNact Studio	Yes – SensiNact Studio
FIWARE	Yes	Different components use different technologies. Some provides other no	Different components use different technologies. Some provides other no	Different components use different technologies. Some provides other no	No	Fiware-lab could be used to deploy and configure new components. No for development
IoTivity	Yes	Uses - Maven	Yes	docker can be used to create containerized packages	Yes - IoTivity simulator	Yes - IoTivity simulator
SeniorSome	Not available	Not Available	Not available	Not available	Not available	Not available

Table 25 represents existing semantic platform in IoT Platforms. It differs if IoT Platforms provides semantic tools and semantic enhanced. Also, other functionalities are shown it, if they exist.

Table 25: Semantic ready platform

Platform	Semantic enhanced	Semantic tools	Offered Functionalities
universAAL	Yes	Yes	<ul style="list-style-type: none"> <li>– Simplify the process of creating ontologies for use on universAAL,</li> <li>– Lower learning threshold.</li> <li>– Reduce effort required (time)</li> <li>– Limit error-prone activities.</li> <li>– Reuse in universAAL and for other platforms (representations)</li> </ul>

SOFIA2	Yes	Yes - Ontologies Management tool, API Management tool	<ul style="list-style-type: none"> <li>– Creating, modifying and delete an ontology and a ontology group.</li> <li>– Searching an ontology and ontology group following some criteria.</li> <li>– Finding and subscribing to an ontology and ontology group.</li> <li>– Subscription to an ontology.</li> <li>– Autorization to one ontology or group of ontologies</li> </ul>
OpenIoT	Yes	APIs, Request Definition GUI, Common utils and libraries	The console pane (bottom pane) provides workspace validation information (problems/warnings) as well as a debug preview of the generated SPARQL code for the designed service.
SensiNact	Provider/service/resource generic datamodels	Provides APIs	<p>Provides query methods over devices based on sensiNact data model</p> <p>Provides dedicated AHA service providers specialized for DS6 specified AHA functions</p>
FIWARE	No	-	-
IoTivity	Yes	OneloTa OCF design tool	Data models creation in order to reach interoperability between devices and platforms
SeniorSome	Not available	Not Available	Not available

### 4.2.8.2 Mapping between platform-specific development tools and proposed ACTIVAGE development tools

For each of the tools that are described as necessary in AIoTES, from Table 26 to Table 30 a mapping is made with the tools provided by each platform.

Table 26: Support tools

Platform	Documentation	Wiki	Tutorials	Code samples	Discussion forum	Training
universAAL	Yes	Yes	Yes	Yes	Not specified	Not specified
SOFIA2	Yes	Yes	Yes	Yes	Not specified	Not specified
OpenIoT	Yes	Yes	Yes	Yes	Not specified	Not specified
SensiNact	Yes	Yes	Yes	Yes	Not specified	Not specified
FIWARE	Yes	Yes	Yes	Yes	Not specified	Not specified
IoTivity	Yes	Yes	Yes	Yes	Not specified	Not specified
SeniorSome	Not available	Not Available	Not available	Not available	Not available	Not available

Table 27: IDE tools

Platform	Code generator	Code templates	Service composer
universAAL	Not specified	Not specified	Not specified
SOFIA2	Yes - source code templates	Yes - source code templates	Yes - source code templates
OpenIoT	No tool matching AIoTES requirements	No tool matching AIoTES requirements	No tool matching AIoTES requirements
SensiNact	Not specified	Yes – source code templates	Yes – sensiNact Studio
FIWARE	Yes - source code templates	Yes - source code templates	Yes - source code templates

IoTivity	Yes - IoTivity source code samples	Yes - IoTivity source code samples	Yes - IoTivity source code samples
SeniorSome	Not available	Not Available	Not available

Table 28: Data/visual analytics tools

Platform	Data manipulator	Data analyzer	Feature/result viewer	Visualization explorer
universAAL	Not specified	Not specified	Not specified	Not specified
SOFIA2	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
OpenIoT	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SensiNact	Historical statistics agent	Historical statistics agent	Technosens E-lio manager web application	Not specified
FIWARE	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
IoTivity	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SeniorSome	Not available	Not Available	Not available	Not available

Table 29: Data Lake tools

Platform	ACTIVAGE data model workbench	Metadata storage explorer
universAAL	Not specified	Not specified
SOFIA2	No tool matching AIOTES requirements	No tool matching AIOTES requirements
OpenIoT	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SensiNact	Not specified	Not specified
FIWARE	No tool matching AIOTES requirements	No tool matching AIOTES requirements
IoTivity	Yes - IoTivity cloud connection tools	No tool matching AIOTES requirements
Platform	ACTIVAGE data model workbench	Metadata storage explorer

Table 30: Semantic Interoperability Layer tools

Platform	ACTIVAGE ontology explorer	Query translator	Device semantic editor	Service semantics editor
universAAL	Not specified	Not specified	Not specified	Not specified
SOFIA2	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
OpenIoT	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SensiNact	Not specified	Not specified	Not specified	Not specified
FIWARE	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
IoTivity		No tool matching AIOTES requirements	Yes - oneIoTa OCF desing tool	No tool matching AIOTES requirements
SeniorSome	Not available	Not Available	Not available	Not available

## 4.2.9 Mapping between development tools requirements and modules

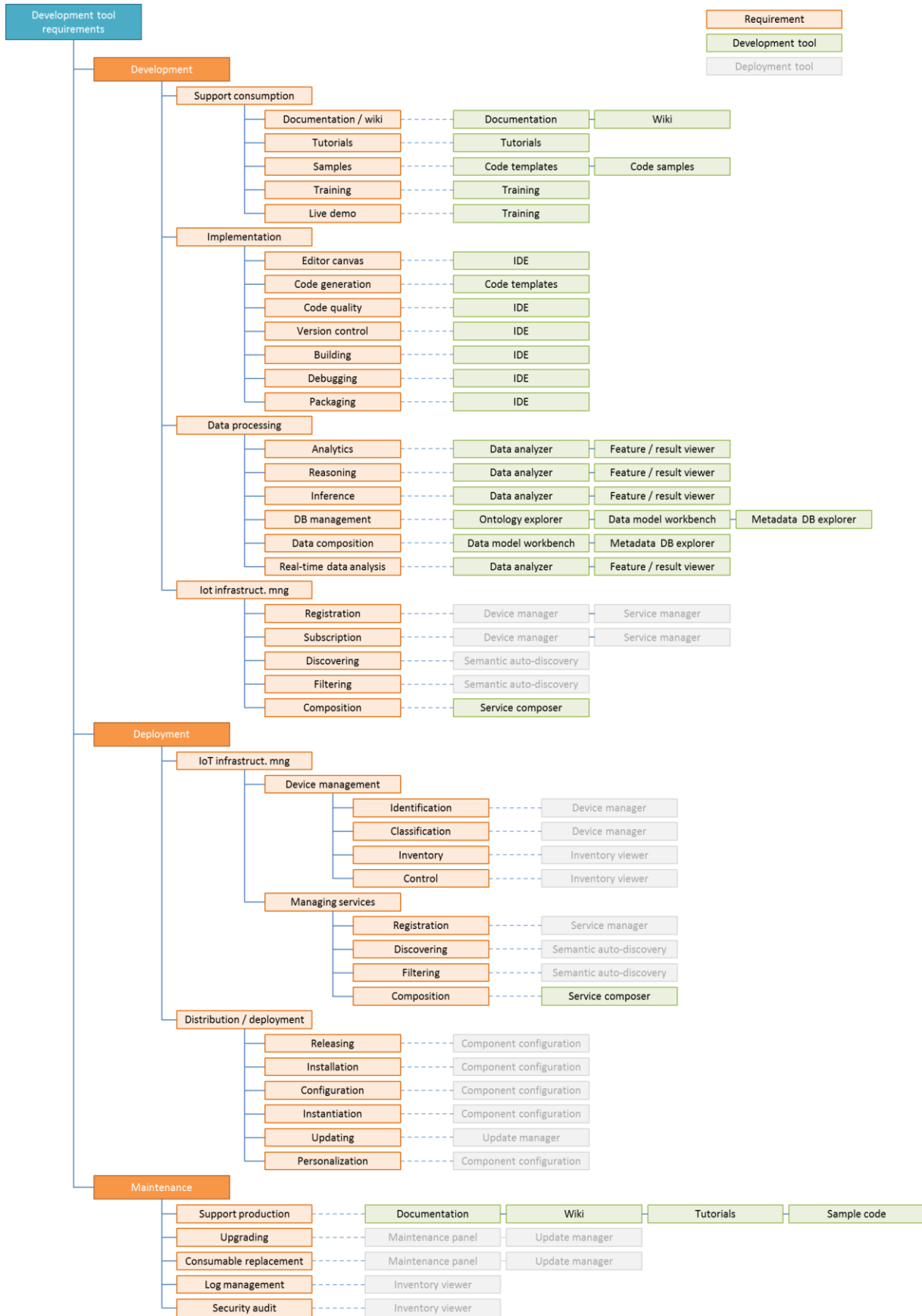


Figure 60: Mapping between requirements (orange) to the ACTIVAGE development tools (green).



The development tools, as described in the sections above, cover the development tools requirements outlined in Section 2.1. Figure 38 provides the mapping between requirements and modules. Orange boxes denote requirements, while green boxes denote the corresponding ACTIVAGE development tools. Part of the requirements is mapped to deployment tools (grey boxes), which are described in Section 5.1.

## 4.3 Tool development plan

The development tools will be presented to developers using the planned information channels of the project:

- The ACTIVAGE web page, where a dedicated section will inform about the existence of the development tools.
- During the dissemination phase of the project, making public the availability of the tools in forums, conferences, etc. Web forums of the platforms included in the ACTIVAGE architecture (Fiware, UniversAAL, etc.), where developers can be informed about these new tools.

Finally, the new GIT repositories containing source code and examples will be public and therefore, indexed by major search engines, making much easier their access for external developers.

Next, the plan to deliver the development tools before they are effectively accessible by external users.

### 4.3.1 Planning

All the aforementioned development tools will be available at the end of month 30, after tasks 4.1 and 4.2 have been finished. More in detail, the working periods of the different tools will be as follows:

Table 31: Planning

Tool	Start (month)	End (month)
<b>Semantic Interoperability Layer tools</b>	2	20
<b>Data Lake tools</b>	2	16
<b>Data / visual analytics tools</b>	12	22
<b>Integrated Development Environment (IDE)</b>	18	30
<b>Support</b>	2	30

This distribution ensures that all developer tools are available by the specified date.

# 5 Deployment tools

## 5.1 Architecture

The ACTIVAGE deployment tools allow IoT site administrators and application developers to register IoT components and applications to the overall IoT ecosystem and allow deployers to discover already existing ones, thus facilitating the actual deployment of IoT applications in the deployment sites.

The ACTIVAGE deployment tools offer web-based means for developers to register their components (devices, applications, etc.) to the AIoTES. They also offer cloud-based means for the semantic discovery of already registered components, in order to support the overall deployment process. The deployment tools are part of the ACTIVAGE application tools, which operate at the higher level of the AIoTES architecture, along with development tools and data analytics. The positioning of the deployment tools within the overall ACTIVAGE architecture is depicted in Figure 61.

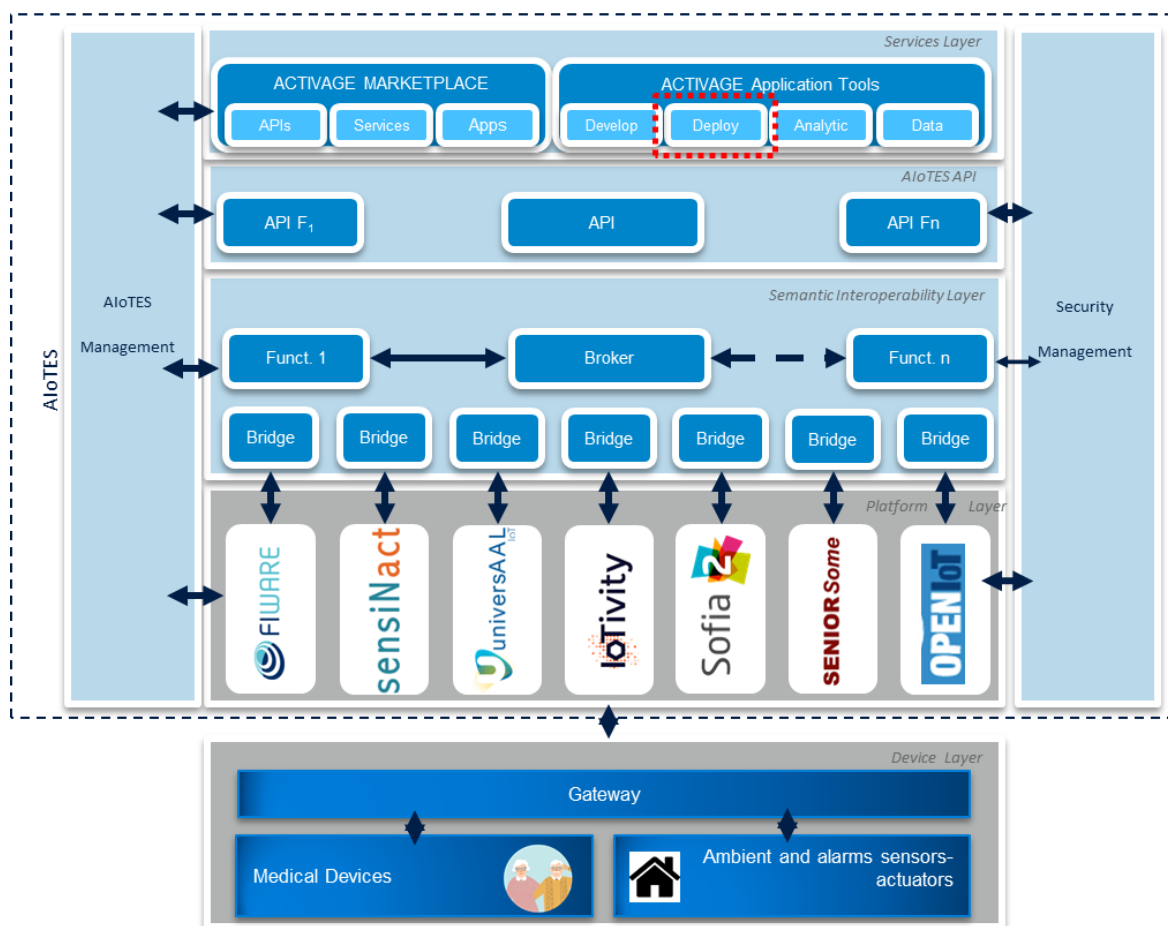


Figure 61: Positioning of the ACTIVAGE deployment tools component within the overall ACTIVAGE architecture.

Components that can be registered and deployed can roughly be separated in the following categories:

- Devices (sensors, actuators, etc.)
- Infrastructure (gateways, servers, etc.)
- Applications (software that runs on top of the infrastructure, using information from the devices)

The functionalities offered by the ACTIVAGE deployment tools are summarized in Table 18. The developer or IoT site administrator can register new components and modify his/her registered components. The developers may be anyone creating IoT applications and components, either from inside or outside ACTIVAGE, wishing to make their components available through the ACTIVAGE marketplace. The deployer can use the semantic discovery tools to search for existing components registered by the community of IoT developers, in order to deploy them to the actual deployment site. Configuration and maintenance functionalities are also provided, for the proper installation and operation of the deployed application.

Table 32: The functionalities offered by the ACTIVAGE deployment tools.

Functionality	Description
<b>register</b>	The developer of a new component can register it to the ACTIVAGE AIoTES, in order to be discoverable by deployers (or other developers, for combining components for new applications) and facilitate the deployment at a deployment site.
<b>edit</b>	The developer can edit the information regarding an already registered component.
<b>delete</b>	The developer can delete an already registered component.
<b>discover</b>	The deployer can search for existing registered components meeting his/her needs, either by searching for a specific component or for semantically similar components, through semantic queries.
<b>deploy</b>	The deployer can proceed to the actual deployment of a registered component in the pilot site.
<b>commissioning</b>	The deployer can set up and configure the component in the deployment unit, specifically adding the information required for the correct functioning of the system, which the system cannot automatically detect (such as the position of devices within the home).
<b>benchmarking</b>	Determine if a component is working appropriately in a secure environment. A deployer can determine whether the component deployer is working properly or not based on the device status and resource consumption. Benchmarking should provide functions to retrieve memory consumption, computation consumption and input/output traffic flow.
<b>inventory</b>	Keep track of the components as a whole: where they are, who was the last person to operate them, when the last update was, and what is their status (operational, storage, under maintenance, end of life, etc.). There is traceability for every component.
<b>update</b>	Keep the software binaries and configuration up to date, specifically for remote and batch operations over many deployment units.
<b>maintenance</b>	Keep the components working in optimal conditions. Some components will require consumables (like batteries, or ink), and/or part replacements; the deployer needs to know, and be notified when this is required.

The above functionalities suggest the following workflow for the deployment of a component at a deployment unit:

- The developer/administrator registers a new component to the ACTIVAGE AIoTES. In case of an application component, it could be an application developed using the ACTIVAGE development tools described in Section 4.1.
- The developer/administrator can modify or delete the registered component, as needed.

- When a deployer needs to deploy a component at a specific deployment unit, he/she uses the deployment tools discovery functionalities to search for components meeting his/her needs.
- Once the desired component has been discovered, it can be deployed at the actual deployment unit.
- The deployer uses the commissioning functionalities, in order to configure the deployed application at the specific deployment unit.
- The deployer uses the maintenance functionalities (benchmarking, inventory, update, maintenance), in order to ensure the proper operation of the application.

All functionalities of the ACTIVAGE deployment tools are offered through a cloud-based platform, which is based on similar platforms developed for other European projects, such as the In Life and Cloud4All projects.

Component registration, along with edit/delete functionalities, is offered through appropriate web forms. Regarding the discovery of a registered component, the ACTIVAGE deployment tools offer semantic query functionalities. The developer can search for existing components by providing queries regarding the semantics of the desired devices or functionalities. For instance, a deployer may wish to search for sensors for motion detection or for applications providing behavioral monitoring of an individual. The deployment tools discovery functionalities can use this query and search for components which are semantically similar to the desired ones, e.g. PIR<sup>27</sup> motion sensors and sound-based motion sensors, or appliance usage and indoor localization applications, respectively for the above examples.

In order to perform such semantic queries, the deployment tools are directly connected to the ACTIVAGE interoperability layer. Architecturally, all registered components are viewed as assets, whether they are hardware or software ones. The semantic interoperability layer maintains ontologies and semantic mappings for all types of registered components (sensors, devices, cloud servers, applications, etc.). Once a new component is registered, the component-related ontologies and data models are updated accordingly. When a user of the deployment tools submits a discovery query, the corresponding ontologies are used, in order to search for semantically similar components.

As already mentioned, the registration and discovery services of the deployment tools are available to the user through a cloud-based web graphical user interface (GUI). However, they are also exposed through a web application programming interface (API), in order to be used by the ACTIVAGE development tools, as described in 4.1.

The high-level architecture of the ACTIVAGE deployment tools component and its connection to the other ACTIVAGE components is depicted in Figure 62.

The actual deployment of a component at a deployment unit (e.g. a specific house) involves the parametrization and configuration of the component, for the needs of the actual unit, as well as ensuring it is executed in a secure environment, it is regularly updated, etc. The configurations needed by the ACTIVAGE deployment sites will guide the design of the ACTIVAGE deployment tools, so that the registered components and the associated semantic models include all the necessary information.

An overview of the ACTIVAGE deployment tools is depicted in Figure 63. They are divided in the following categories:

- **IoT infrastructure management tools:** Tools for registering devices and services to the AIOTES ecosystem, as well as for semantically discovering and testing them.

---

<sup>27</sup> Passive Infrared sensor

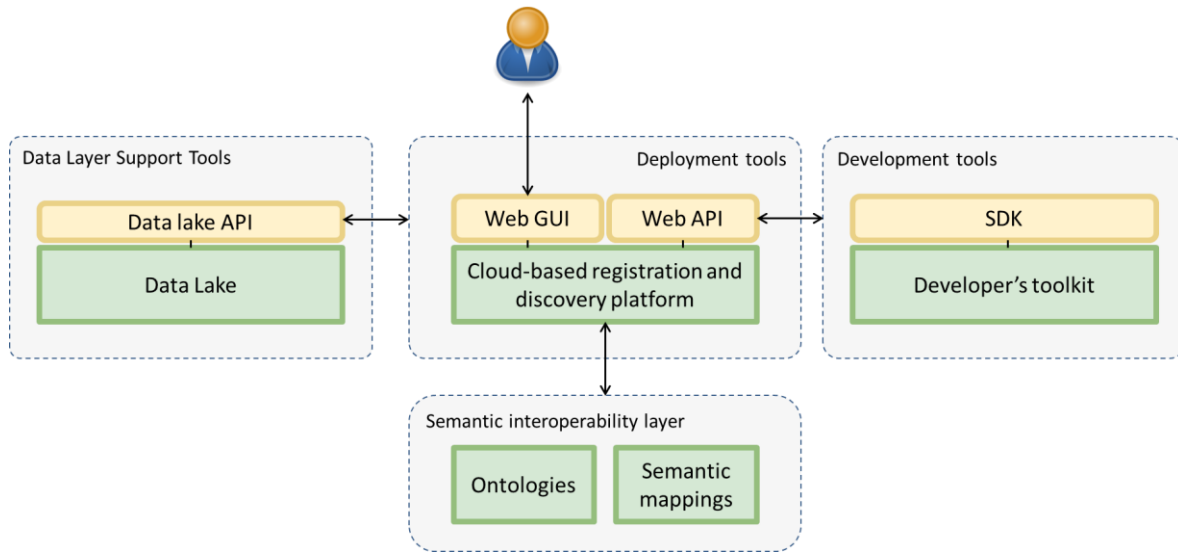


Figure 62: Architecture of the ACTIVAGE deployment tools component and its connection to the other ACTIVAGE components.

- **Deployment management tools:** Tools for component configuration and management of installations in deployment units.

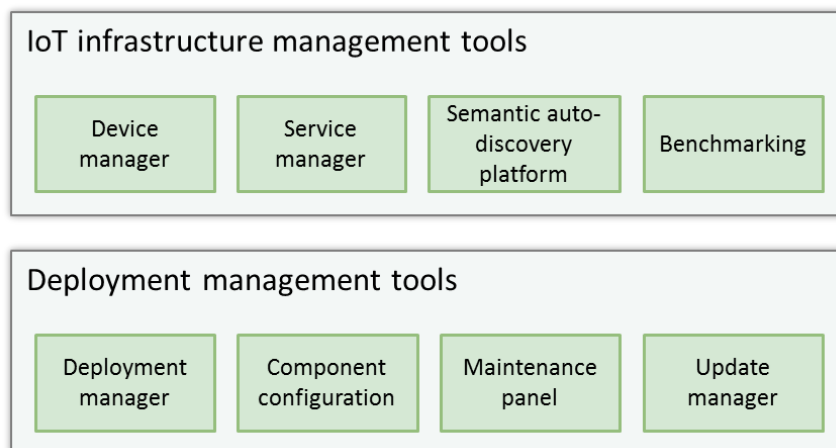


Figure 63: The ACTIVAGE deployment tools.

In the following sections, the ACTIVAGE deployment tools are described in detail.

### 5.1.1 IoT infrastructure management tools

The IoT infrastructure management tools provide utilities for managing the components of the IoT infrastructure, i.e. the devices and the developed services and applications, in order to facilitate their deployment in an actual deployment unit. The IoT infrastructure management tools consist of the following, as also depicted in Figure 64.

- Device manager
- Service manager
- Semantic auto-discovery platform
- Benchmarking

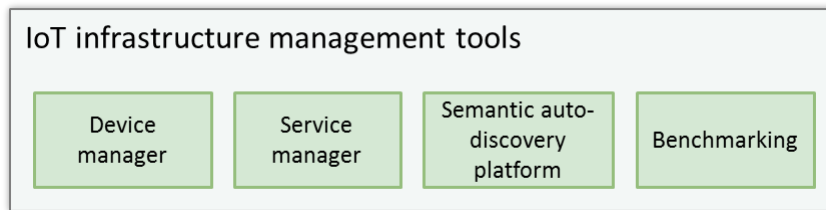


Figure 64: The IoT infrastructure management deployment tools.

The IoT infrastructure management deployment tools are described in detail in the following sections.

### 5.1.1.1 Device manager

The device manager deployment tool is a Graphical User Interface (GUI), through which an IoT device can be registered into the AIOTES and its characteristics be edited/updated/deleted when necessary. The registration of a device involves specifying its functionality, its type (sensor, actuator, etc.), the type of data it collects, the type of commands/parameters it accepts, etc. Proper registration of a device in the AIOTES also involves specifying the semantics of the device’s characteristics, i.e. which ACTIVAGE data model ontology/attribute it belongs/corresponds to. This is important for later discovery of the device by application developers, as well as for the proper translation of the device’s data into the naming conventions of the unified ACTIVAGE data model, in order for its data to be available through the Data Lake. The device manager tool offers forms through which the user can insert the required characteristics and edit or delete them later, if an update is needed. It also provides fields through which one can specify those parameters of the device that should be configurable when it will be deployed in a specific deployment unit (e.g. configurable operational range, depending on the deployment unit’s area). The device manager tool is connected to the Semantic Interoperability Layer of the AIOTES, as depicted in Figure 65, in order to have access to the device ontologies and semantic attributes.

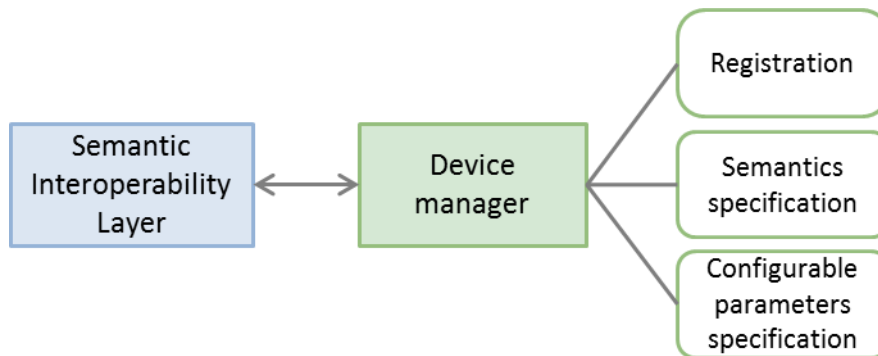


Figure 65: Functionalities and communication of the device manager deployment tool.

#### Usage

The device manager tool is offered as a Web-based form, through which the deployer can insert the details about a device to be registered, or open the information for an already registered device, to edit it. The form fields correspond to the data needed for a device to be properly registered:

- Device type (sensor, actuator, etc.)
- Type of data it collects
- List of parameters it accepts

- List of commands it supports
- Which device entity it corresponds to, in the ACTIVAGE ontology (the available entities are displayed through a drop-down menu). The corresponding entities are ones that have been inserted to the ACTIVAGE ontology through the [Device semantics editor](#) development tool.
- How the device parameters and commands are semantically mapped to the attributes of the corresponding ontology entity.

### 5.1.1.2 Service manager

The service manager deployment tool is similar to the device manager tool described above, but it regards developed services instead of devices. Through the GUI of the service manager tool, the developer of a service or application can register it to the AIOTES, in order for it to be later discoverable and composable by other developers. The registration of a service in the AIOTES involves specifying its functionality, its inputs and outputs, the types of data it needs or exports, etc. Service registration also involves specifying the semantics of the service's functionality, inputs, outputs, etc., so that it is added to the corresponding ontologies of the AIOTES SIL. The service manager provides forms through which the developer can provide all this information, as well as update it when necessary. Similar to the device manager tool, the service manager tool also allows the developer to specify which parameters should be configurable for the deployment of the service in a specific deployment unit (e.g. configurable number of temperature sensors used by the application). The service manager tool is connected to the AIOTES SIL, as depicted in Figure 66, in order to have access to the service-related ontologies and semantic mappings.

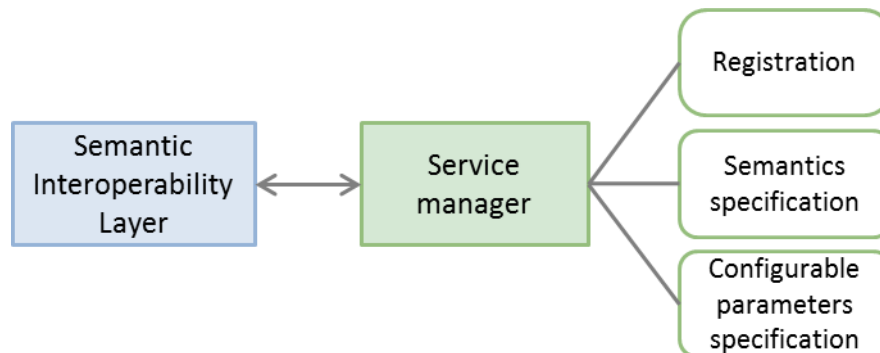


Figure 66: Functionalities and communication of the service manager deployment tool.

#### Usage

The service manager tool is offered as a Web-based form, through which the deployer can insert the details about a service to be registered, or open the information for an already registered service, to edit it. The form fields correspond to the data needed for a service to be properly registered:

- Functionality description
- List of input parameters it accepts
- List of output parameters it produces
- Types of data needed
- Types of data exported
- Which service entity it corresponds to, in the ACTIVAGE ontology (the available entities are displayed through a drop-down menu). The corresponding entities are ones that

have been inserted to the ACTIVAGE ontology through the [Service semantics editor](#) development tool.

- How the service parameters (inputs, outputs, etc.) are semantically mapped to the attributes of the corresponding ontology entity.

### 5.1.1.3 Semantic auto-discovery platform

The semantic auto-discovery platform allows the deployer to discover AIOTES components (devices and services) that meet certain semantically-specified criteria. The semantic auto-discovery platform offers a GUI through which the deployer can formulate a semantic query for devices or services with desired functionalities and characteristics. The auto-discovery platform communicates with the AIOTES SIL, in order to discover which devices and services semantically match the criteria submitted by the deployer, and retrieves relevant components. This basic mode of operation is enhanced with a second mode of operation, similar in concept to content-based search engines. In this second mode, the deployer can use an existing device or service as the query, and ask the tool to retrieve devices or services with semantically similar characteristics to the query one. The functionalities and communication of the semantic auto-discovery platform are depicted in Figure 67.

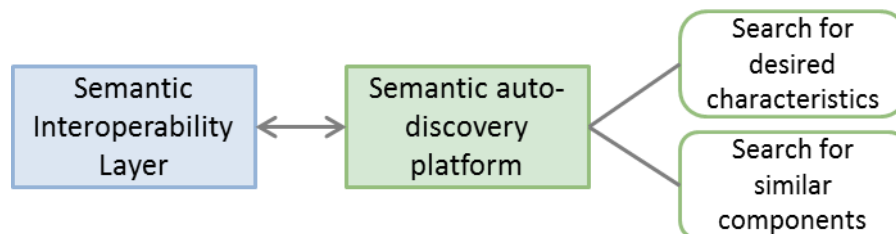


Figure 67: Functionalities and communication of the semantic auto-discovery platform deployment tool.

#### Usage

The semantic auto-discovery platform is offered through a Web-based Graphical User Interface. The deployer can insert the characteristics of a desired service, in order to discover semantically similar services. The desired characteristics can be inserted in one of two ways:

- Through a form, where the deployer can manually insert the characteristics of the desired device/service, such as types of inputs/output, parameters, etc.
- Through the selection of another device/service and selecting a “discover similar devices/services” option.

The list of discovered devices or services is presented to the deployer in a list view, from which he/she can select one to view further details.

### 5.1.1.4 Benchmarking tool

The benchmarking tool allows deployers to determine whether an application or service is working properly or not. Benchmarking should provide functions, in a REST-based way, to retrieve performance values such as: memory consumption, computation consumption and input/output traffic flow; and status values like correct configuration in terms of security and privacy, both key points in ACTIVAGE. At the same time, a graphical user interface must be provided allowing non-technical users to visualize the current status of the services deployed in its property. In order to perform the performance validation operations the interaction with the deployment management tools is required, in the same way that the interaction with the



deployment technology used. For the security and privacy validation the interaction with the Security and Privacy ACTIVAGE tools is required. A more detailed study of the relationships between the benchmarking module and Security and Privacy ACTIVAGE tool will be provided in next releases. In the same way, the choice of a concrete technology for deploying new services in the system will help us to know better the way in which performance information can be gathered. The benchmarking tool is connected to the Semantic Interoperability Layer, in order to retrieve the services supported by the AIOTES. The functionalities and communication of the benchmarking deployment tool are depicted in Figure 68.

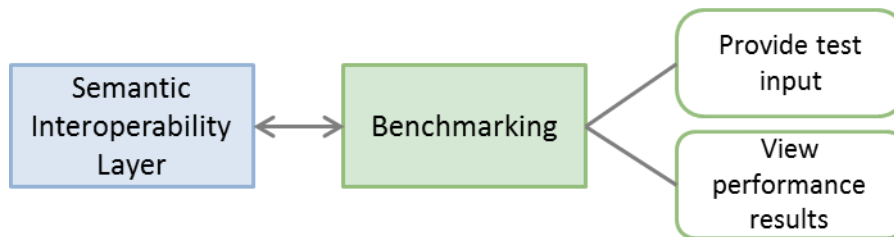


Figure 68: Functionalities and communication of the benchmarking deployment tool.

### Usage

The benchmarking tool offers a Graphical User Interface, through which the deployer select a service to benchmark, can provide test input to the service being benchmarked, and view the results of performance analysis, after the service has been executed.

## 5.1.2 Deployment management tools

The deployment management tools handle the actual deployment of a component (device or service) in a deployment unit, and the management of the deployment. They cover aspects of component configuration, traceability and maintenance and consist of the following tools, as also depicted in Figure 69.

- Deployment manager
- Component configuration
- Maintenance panel
- Update manager

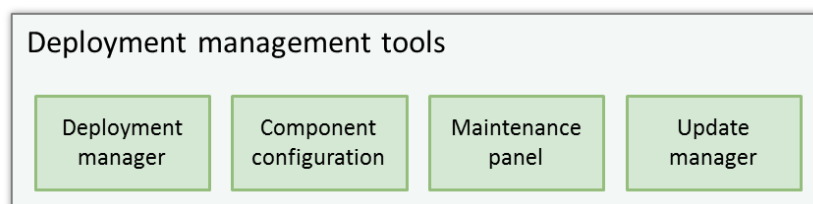


Figure 69: The deployment management tools.

The deployment management tools provide management functionalities for the deployer of an application at a specific deployment unit, e.g. a house. They are similar in nature, and in correspondence, to the deployment management functionalities of the AIOTES Management Toolkit of Task T5.3, described in Deliverable D5.2 “Support and training plan for deployment of AIOTES”, although the latter provides management functionalities for the administrator of a whole Deployment Site. They are also similar to the capabilities offered by the AIOTES management dashboard, presented in Deliverable D5.1 “Integration plan and

operational framework”, which provides functionalities for the management of the whole ACTIVAGE ecosystem.

### 5.1.2.1 Deployment manager

The deployment manager tool provides a graphical interface through which the deployer can create/edit a specific deployment installation (e.g. in a specific home) and have an overview of the deployment inventory, along with its installed devices and services. Through the deployment manager, the deployer can edit/view the component installation characteristics, such as the locations where devices are installed, their current operational status, etc. The deployer can also view the change history regarding the component, such as who was the last person to operate a device or service, when a device or service was last updated, when maintenance activities were last performed, etc. Such kind of deployment-specific metadata are stored in the Semantic Interoperability Layer, while the raw data collected by the deployed devices are stored in the platform-specific databases and are accessible through the Data Lake. Thus, the deployment manager communicates with the Semantic Interoperability Layer, as depicted in Figure 70, in order to retrieve all deployment-related metadata.

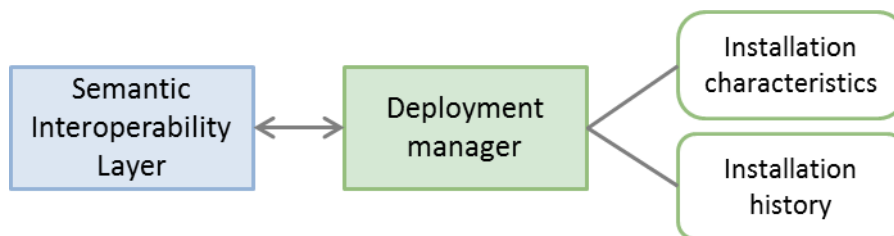


Figure 70: Functionalities and communication of the deployment manager tool.

#### Usage

The deployment manager is a Web-based graphical user interface, through which the deployer can create a new deployment installation. Its main interface is a tree-like structure, showing the components (devices, services, etc.) of a deployment. The deployer can perform the following actions, regarding a deployment installation:

- Create a new deployment installation
- Add devices/services in the installation. The devices/services can be added to the tree-like structure by selecting among the devices/services registered in AIOTES.
- View/insert/edit installation-specific information for the devices/services of an installation, through edit forms.
- Configure selected devices/services, through the [component configuration tool](#)
- View maintenance information for selected devices/services, through the [maintenance panel tool](#).
- Perform update operations for selected devices/services, through the [update manager tool](#).

### 5.1.2.2 Component configuration

The component configuration deployment tool is a GUI through which the deployer of a component (device or service) can provide appropriate values for all configurable parameters that are necessary for the deployment of a component in an actual deployment unit. Example of configurable parameters include the operational range of a device, which may be adjusted according to the deployment unit’s area, or the number of temperature

sensors used by an application, which may be adjusted according to how many sensors are available in the deployment unit. The component configuration is connected to the Semantic Interoperability Layer, in order to store the deployment-specific configuration. It is also connected to the deployment manager tool (see below), from which the deployer can select the device to configure. The functionalities and communication of the component configuration deployment tool are depicted in Figure 71.

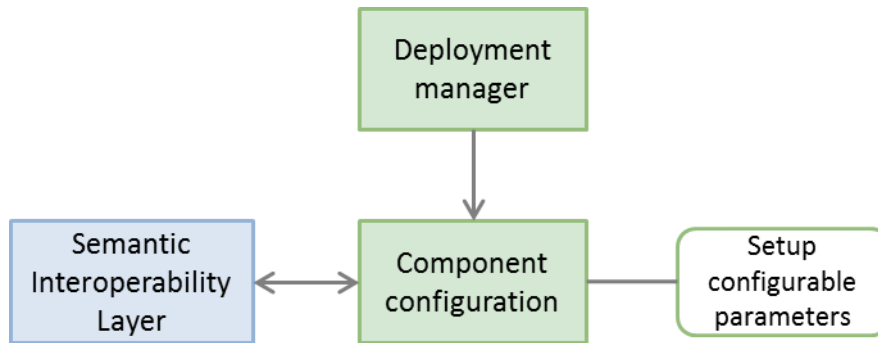


Figure 71: Functionalities and communication of the component configuration deployment tool.

### Usage

The component configuration tool is offered as a form where the deployer can insert and submit the configuration parameters for a specific device in a deployment. The component configuration tool is connected to the [deployment manager](#), from which the deployer can view the whole deployment installation and select devices to configure.

### 5.1.2.3 Maintenance panel

The maintenance panel deployment tool provides a graphical interface in order to facilitate the deployer in performing maintenance activities in a deployment installation. Through the maintenance panel's interface, the deployer can view the operating status of all components (devices and services) installed in a deployment unit. Operating status includes possible malfunctions of devices or services, battery levels, etc. The maintenance panel also provides a notification service, in order to provide notifications to the deployer whenever a change in the operating status of a component happens. The maintenance panel communicates with the Semantic Interoperability Layer, as depicted in Figure 72, in order to retrieve all deployment-specific information about the devices and services installed. It also communicates with the deployment manager, from which the deployer can select the installation components for which to open the maintenance panel.

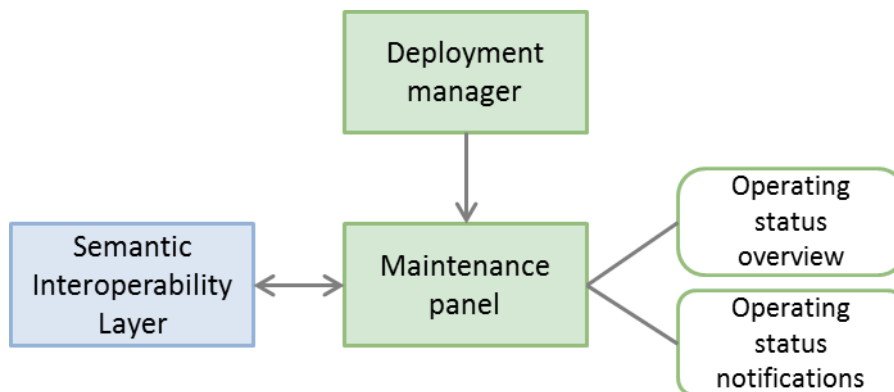


Figure 72: Functionalities and communication of the maintenance panel deployment tool.

## Usage

The maintenance panel tool offers a Graphical User Interface, through which the deployer can view the maintenance information for the selected device/service, and control the operation of the notification mechanisms, i.e. turn on or off notifications for specific devices/services.

### 5.1.2.4 Update manager

The update manager deployment tool provides a graphical user interface which facilitates the deployer in updating the installed components (devices and services) when new versions are available. The update manager GUI displays the versions of the devices and services installed in a specific deployment unit, and shows notifications if new versions are available for each component. Using the interface, the deployer can select a specific component and perform the update, by e.g. downloading and installing a new version of a service, or being directed to relevant online stores for ordering new versions of equipment. The update manager communicates with Semantic Interoperability Layer, as depicted in Figure 73, in order to retrieve all relevant metadata regarding the installed component versions. It also communicates with the deployment manager, from which the deployer can select the devices/services for which to open the update manager.

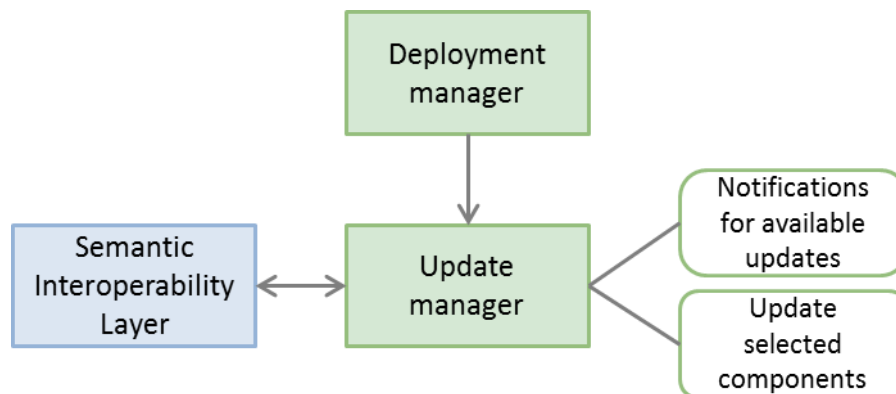


Figure 73: Functionalities and communication of the update manager deployment tool.

## Usage

The update manager tool offers a Graphical User Interface (GUI), through which the developer can view the version status of the selected devices or services. The GUI offers links to new versions of devices/services and allows the deployer to directly perform an update, from within the update manager, whenever possible.

## 5.1.3 Mapping between deployment tools requirements and modules

The deployment tools, as described in the sections above, cover the deployment tools requirements outlined in Section 2 provides the mapping between requirements and modules. Orange boxes denote requirements, while green boxes denote the corresponding ACTIVAGE deployment tools. Part of the requirements is mapped to development tools (grey boxes), which are described in Section 4.1.

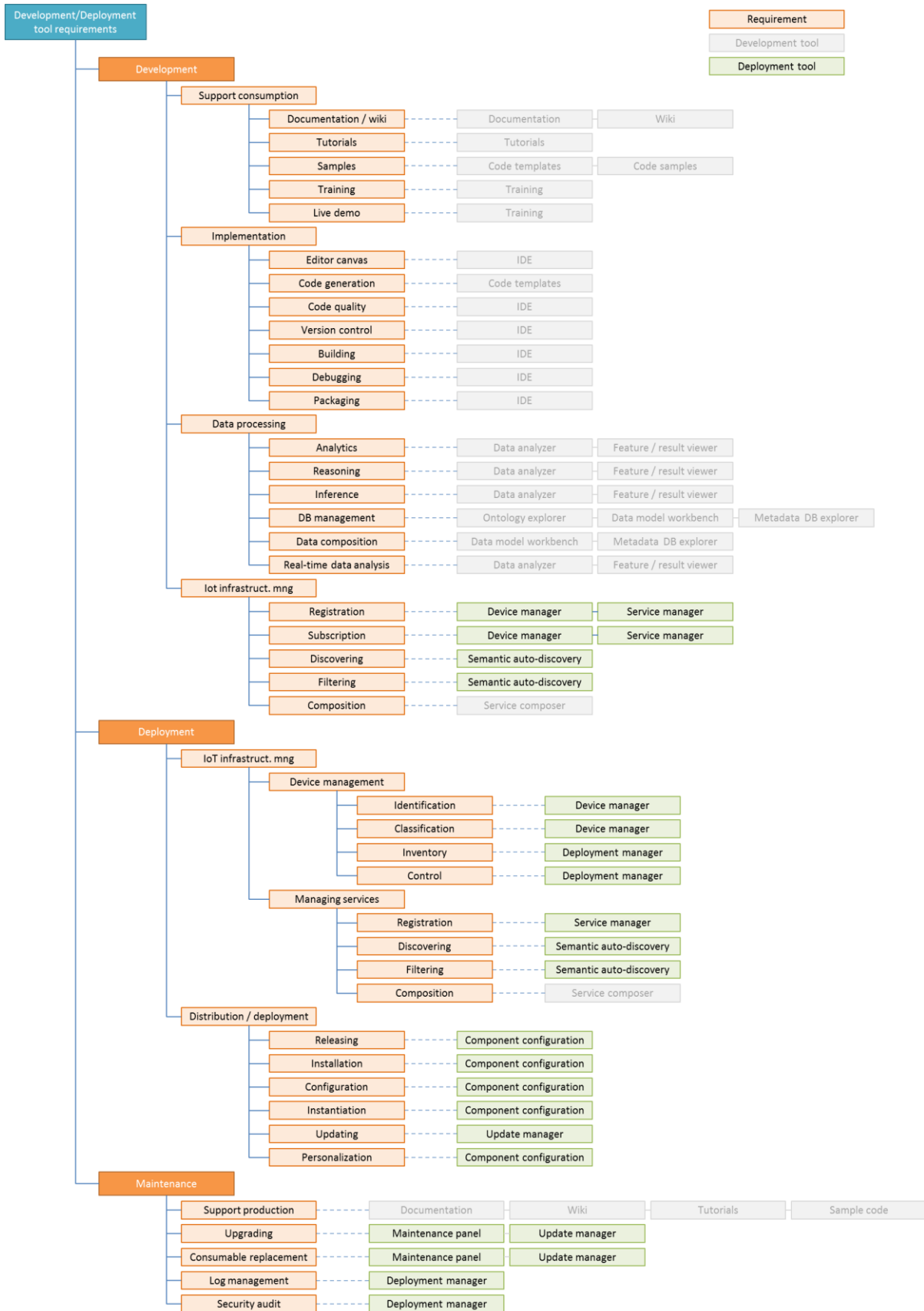


Figure 74: Mapping between requirements (orange) to the ACTIVAGE deployment tools (green).

## 5.2 Available deployment tools supported by the ACTIVAGE IoT platforms

### 5.2.1 Platform independent Available Deployment tools

#### 5.2.1.1 Docker

Docker<sup>28</sup> is software container platform that encapsulates applications to run and manage them side-by-side in isolated containers to obtain better performance and compute density.

Docker allows to package an application with all of its dependencies into a standardized unit for software development. Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries; to sum up, anything that could be installed on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

These containers can communicate with each other through a Docker network specifying the direction and port, and the Docker tool can handle the lifecycle of the containers in a way that this packages of software run isolated on a shared operating system being started, ran or stopped when needed.

Despite other virtualization methods or machines, containers do not build a full operating system, instead only libraries and settings required to make the software work as needed.

#### 5.2.1.2 Virtual Machine

A virtual machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.

Specialized software, called a hypervisor, emulates the PC client or server's CPU, memory, hard disk, network and other hardware resources completely, enabling virtual machines to share the resources. The hypervisor can emulate multiple virtual hardware platforms that are isolated from each other, allowing virtual machines to run Linux and Windows Server operating systems on the same underlying physical host. Virtualization limits costs by reducing the need for physical hardware systems. Virtual machines more efficiently use hardware, which lowers the quantities of hardware and associated maintenance costs, and reduces power and cooling demand. They also ease management because virtual hardware does not fail. Administrators can take advantage of virtual environments to simplify backups, disaster recovery, new deployments and basic system administration tasks.

#### 5.2.1.3 OSGi based deployment

According to the OSGi Alliance<sup>29</sup>, the OSGi technology is a set of specifications that define a dynamic component system for Java. These specifications enable a development model where applications are (dynamically) composed of many different (reusable) components. The OSGi specifications enable components to hide their implementations from other

---

<sup>28</sup> <https://www.docker.com/>

<sup>29</sup> <https://www.osgi.org/>

components while communicating through services, which are objects that are specifically shared between components. This surprisingly simple model has far reaching effects for almost any aspect of the software development process.

OSGi reduces complexity by providing a modular architecture for today's large-scale distributed systems as well as small, embedded applications. Building systems from in-house and off-the-shelf modules significantly reduces complexity and thus development and maintenance expenses. The OSGi programming model realizes the promise of component-based systems. The following list contains a short definition of the main concepts of OSGi:

**Bundles** – Bundles are the OSGi components made by the developers.

**Services** – The services layer connects bundles in a dynamic way by offering a publish-find-bind model for plain old Java objects.

**Life-Cycle** – The API to install, start, stop, update, and uninstall bundles.

**Modules** – The layer that defines how a bundle can import and export code.

**Security** – The layer that handles the security aspects.

**Execution Environment** – Defines what methods and classes are available in a specific platform.

These specifications are then implemented by different frameworks, such as:

– **Framework OSGi Knopflerfish**

Knopflerfish <sup>30</sup>is the leading universal open source OSGi Service Platform. Led and maintained by Makewave, Knopflerfish delivers significant value as the key container technology for many Java based projects and products.

– **Apache Felix**

Apache Felix <sup>31</sup>is a community effort to implement the OSGi Framework and Service platform and other interesting OSGi-related technologies under the Apache license.

– **Eclipse Equinox**

The Equinox project<sup>32</sup> is to be a first class OSGi community and foster the vision of Eclipse as a landscape of bundles. As part of this, it is responsible for developing and delivering the OSGi framework implementation used for all of Eclipse.

Different groups also provide additional component, and component implementations to build ever more complex Java/OSGi applications.

– **Pax Runner**

Pax Runner <sup>33</sup>is a tool to provision OSGi bundles in all major open source OSGi framework implementations (Felix, Equinox, Knopflerfish, Concierge).

– **Apache Karaf**

Karaf <sup>34</sup>is a lightweight, powerful, and enterprise ready container powered by OSGi. By polymorphic, it means that Karaf can host any kind of applications: OSGi, Spring, WAR, and much more.

---

<sup>30</sup> <https://www.knopflerfish.org/>

<sup>31</sup> <http://felix.apache.org/>

<sup>32</sup> <http://www.eclipse.org/equinox/>

<sup>33</sup> <https://ops4j1.jira.com/wiki/spaces/paxrunner/overview>

<sup>34</sup> <http://karaf.apache.org/>

### – Apache ACE

Apache ACE <sup>35</sup> is a software distribution framework that allows you to centrally manage and distribute software components, configuration data and other artifacts to target systems. It is built using OSGi and can be deployed in different topologies.

## 5.2.2 universAAL

The universAAL platform is a multi container platform, allowing its modules, as well as the middleware, to be installed in different containers. Currently there are 2 containers offered: OSGi and Android. And there are tools to help in the deployment of both.

Additionally To the tools for deployment, there are tools to customize, configure and set up each deployment. Pax Runner

### 5.2.2.1 Distributions

Distributions are ready made containers, and tools for said containers.

All these tools offer the function to deploy the container, as well as component configuration. Their use is limited to applications using the same container, thus the usage of these tools in AIOOTES support tool set is limited to this factor.

#### 5.2.2.1.1 OSGi Container

universAAL uses the Pax Runner framework to provide runnable instances of universAAL over OSGi. Within universAAL community there are templates and documentation to configure custom runnables.

The AAL Studio tool offers an extension of the open source project Pax Runner for Eclipse. It provides a user interface for managing provision of OSGi bundles on Eclipse. This tool is very useful for deployers, so people creating packages of AAL Applications and platform components can really benefit from this extension. The Pax Runner plugin extension is used for running and debugging uAAL applications that have been developed within Eclipse with AAL Studio. It extends the original Pax Runner plugin, in order to provide a more user friendly interface and a better support for the latest version of Eclipse and OSGi runtimes. To create a start-configuration for universAAL

This tool parses the ".launch" configuration file of the project that is selected in the Project/Package explorer of Eclipse, sorts the bundles according to the start level that they belong and displays them in the tree structure as above. The developer may select which bundle or level should be included or started in the OSGi runtime. Moreover, a new button ("Add Level") has been added for adding levels in the configuration.

Like the Pax runner, karaf is a tool which provides an OSGi environment. But the power of karaf relies on the extensive tool set for deployment, provisioning and other container operations which are installable in each container. universAAL has provided<sup>36</sup> the configuration files and other customizations of the karaf container for a more reliable, robust and pleasant deployment experience.

---

<sup>35</sup> <http://ace.apache.org/>

<sup>36</sup> <https://github.com/universAAL/distro.karaf/wiki>



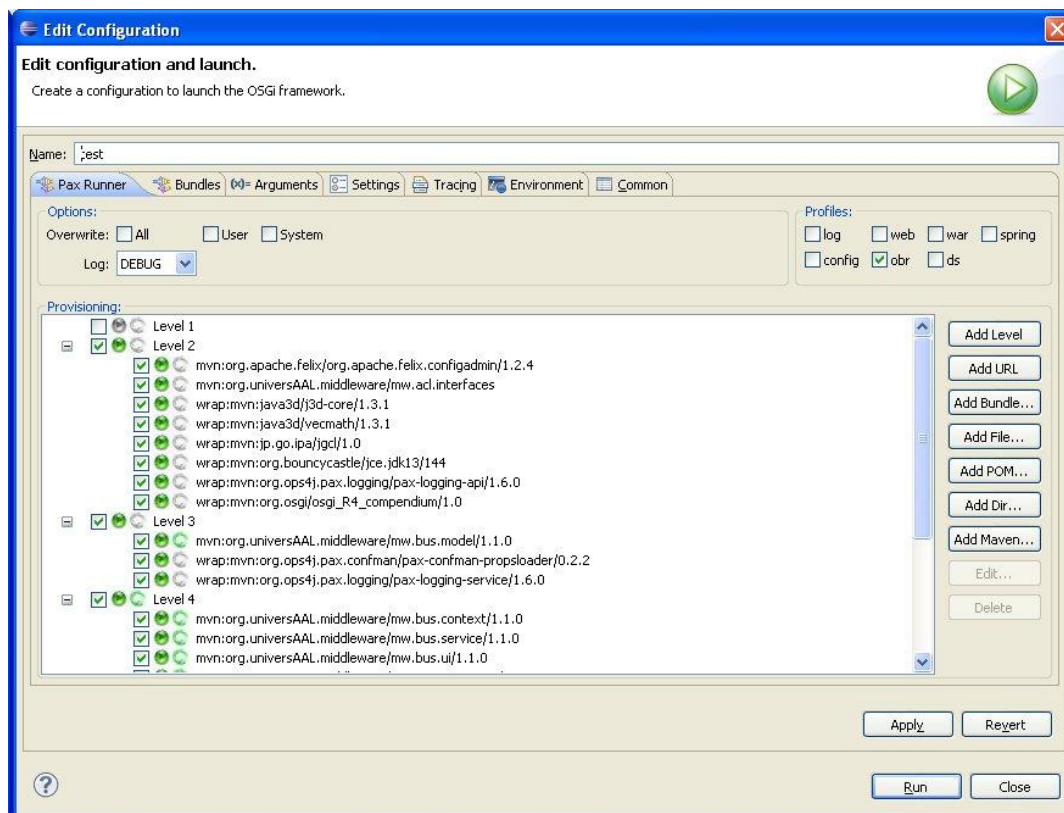


Figure 75 Run configuration of Pax Runner for Eclipse

### 5.2.2.1.2 Android universAAL Middleware App

To provide easy installation of universAAL middleware there is an universAAL middleware app<sup>37</sup> which makes it easy. Then adding modules to this container is as easy as installing universAAL-android ready apps on the system.

### 5.2.2.2 Runtime tools

There is a suite of deployment tools which run on the deployed instance of the container running universAAL. These offer main deployment operation facilities. Here is a small list of the tools:

- **Log Monitor:** The log monitor is a graphical viewer of all the universAAL events; it helps debug deployments by showing the internal operations of the system, as well as showing graphical representations of the exchanged messages between modules.
- **Makro Recorder:** The Makro recorder is useful to create sequence of events by recording the events and services generated in the devices in the deployed space, and being able to later replay them. The main purpose of the Makro recorder is to enable deployers and even end users, to set up rules and automatic responses.
- **Sparql tester:** The sparql tester is a tool which enables deployers to issue sparql queries to the data lake in the space. Thus giving full control over configurations held in that space.

<sup>37</sup> <https://github.com/universAAL/nativeandroid/wiki>

- **Security Profile Management:** a simple tool that helps create and manage user credentials.
- **AAL Space Monitor:** a tool that displays graphically the status of the current AAL Space, showing nodes and their capabilities.
- **Ace server:** a software module, part of karaf suite, which enables automatic update of bundles in a batch of deployments.

Some of these tools are also used for developers while testing, thus you may recognise some of them from Section 4.2.1.2.4.

### 5.2.2.3 Deploy Management & uCC

The universAAL platform initially planned for a digital store, the uStore, which offered support for a market place of software, hardware, services, human resources as well as combinations of these.

As a client for this framework the universAAL Control Center (uCC) was developed. It enabled end users buy, hire, download and install easily AAL apps and services in their AAL space. For this to work the universAAL middleware implemented a deployment management framework which analysed the space's nodes resources, as well as the application (described in xml) to determine which bundles should be installed in which nodes, and whether there could be replication. This function would only work on karaf based deployments.

This system is very interesting to be included in AIOTES since it will facilitate the user management of their services. This concept brings the user perspective to the deployment operations. Sadly this tool is too universAAL specific to be imported in AIOTES support suite.

An extension of the uCC's functionalities is planned. The new functions include:

- User Profile and Security management
- Log Monitor and the capability of recovering remote node logs
- Functional manifest explorer (configuring access rights, etc...)
- Deploy Management improvement; visual tool to deploy uAAP packages, and select the nodes where each module should be installed
- AALSpace physical definitions editor, for defining shapes and positions of objects inside a home or deployment.
- Advanced Configuration Editor
- Ontological explorer: look and browse ontologies, concepts, properties, etc. in the AAL Space
- Graphical Resource Editor (GRE): edit graphically RDF graphs, with special graphic aids for particular resource types (such as locations or shapes).
- AAL space visual statistics
- Connection manager: configure where the space is connected.
- Batch Deployment Manager: to deploy, configure and upgrade many homes/deployments at a time.
- Advanced container function control.
- Visual reasoner, and orchestrator management.

This tool is also usable for developers, thus it has been mentioned as Section 4.2.1.2.6.2.

### 5.2.2.4 Mapping between universAAL and ACTIVAGE deployment tools

Table 33: Mapping between universAAL and ACTIVAGE deployment tools.

universAAL development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?	
Log Monitor	Maintenance Panel / operating status notifications	Yes	No
		<b>How?</b>	<b>Why?</b>
			This tool is too universAAL specific. Although it could be used to observe events and calls coming from the AIOTES universAAL Bridge.
Makro Recorder.	Service Manager / semantics specification	Yes	No
		<b>How?</b>	<b>Why?</b>
		The makro recorder can be used to record actions to manually specify semantical statements. Both recording and replay can be performed thanks to the AIOTES universAAL Bridge. This tool needs maintainence.	
Sparql Tester	Maintenance Panel	Yes	No
		<b>How?</b>	<b>Why?</b>
			This tool is used primariliy to provide an accessible interface for developers to the SparQL database backend, and test or organize database operations. This functionality could be too low level to be included in an ACTIVAGE deployer tool set.
Security Profile Management	Not mapped with AIOTES		
AAL Space Monitor	Maintenance Panel	Yes	No
		<b>How?</b>	<b>Why?</b>
			This tool is too universAAL Specific, it relates to the uSpace, describing all reachable IoT gateways with universAAL middleware.
Deploy Management & uCC	Device Manager; Service Manager; Deployment Manager; Component	Yes	No
		<b>How?</b>	<b>Why?</b>
			This tool is too universAAL Specific. It is relative to universAAL modules and

Configuration; Maintenance Panel; Update manager		applications.
-----------------------------------------------------------	--	---------------

### 5.2.3 SOFIA2

SOFIA2 provides a software management tool to define software versions and configurations of applications deployed as well as listing the assigned and created configurations.

Only the users with the role “Administrador” and “Colaborador” will have access to this menu.

Bear in mind the following considerations:

- A KP can have multiple software applications and each application can have multiple software configurations. Only one software configuration can be active at a time.
- Once a configuration is active and assigned to a KP and Instance, it cannot be changed. When you modify an application or softwareconfiguration, the version will increase in one and will be saved as disabled by default.
  - By default, when a configuration or software application is modified o cloned it will be saved as disabled.
- The software configurations and the software applications may be cloned with their assignments, by default they will be saved as disabled.
- Software applications can be turned off but can not be removed from the system. And can only be modified if they haven´t been assigned to any KP o Instance of KP.

With this management massive updates can be made that affect a large number of KPs and/or instances of KPs.

The main capabilities of that tool are Management of Software Configuration and Assignment of Software in KPs.

#### Management Software Configuration

An administrator can see all the applications and software configurations created in the System. At the top of the screen, there is a field that allows you to filter by name of the software application. There is a filter by “user” only available for users with Administrator role (cf. Figure 76).

In the list we can see the name of the application, version, description and if is activated or not. The list also includes the option to view the details of the softwre configuration or to modify if the software application has not been assigned.

In the first part of the screen is a button “New Configuration” to redirect us to another screen to register a new software application and configuration.

La UI consists of two main parts, “SW Management Inf.” and “Configuration Information”.

In the first part we will indicate the details of software management:

- Name of the application: The system will warn us if we try to create software application with a name that already exists.
- Active: It will indicate us if we want to activate the application or not.

- Application: Allows selecting the war with the software used by clients.
- Description: It is to include on the software application.

**My SW Configuration**

Form

Application Name  User

**Configurations List**

Search:

Application Name	Version	Description	Owner	Active	Options
AplicacionTest	1	Aplicación para probar la subida de ficheros	sofa	<input type="checkbox"/>	
prueba	1		sofa	<input type="checkbox"/>	
SIMPLESTAT	1	simple stat	sofa	<input type="checkbox"/>	
SIMPLESTAT	4	simple stat	sofa	<input type="checkbox"/>	
SIMPLESTAT	3	simple stat	sofa	<input checked="" type="checkbox"/>	
SIMPLESTAT	5	simple stat	sofa	<input type="checkbox"/>	
SIMPLESTAT	2	simple stat	sofa	<input type="checkbox"/>	

Showing 1 to 7 of 7 entries

Figure 76: A Sofia2 deployment tool for selecting active applications per user

In the configuration data part, we will indicate the software configuration properties (for example: kp, instance of kp, connection token, ip, port, etc.)

### Assignment of Software in KPs

Using this option we can list application assignments and software configuration for KPs and Instances of KPs.

From this function we can set specific assignments to a KP and Instance of KP, as well as performing massive assignments such as assign the same software configuration to all instances of a KP, selecting “\*”. Or establishing a software configuration for all the KPs and

Instances of KP with “\*”. An assignment can be removed using the icon .

## SW Assignment to KPs

### Form

### List of Assignment

Search:

Kp	Kp Instance	Application - Version	Conf. Version	User	Ops.
StatKP *		SIMPLESTAT- V1	Version -2	sofia	

Showing 1 to 1 of 1 entries

### 5.2.3.1 Mapping between deployment tools requirements and modules

Table 34: Mapping between SOFIA2 and ACTIVAGE deployment tools.

FIWARE development tool	Corresponding ACTIVAGE development tool(s).	Can the tool be used in AIOTES?
Management Software Configuration tool	Not mapped with AIOTES	
Assignment of Software in KPs	Not mapped with AIOTES	

## 5.2.4 OPENIOT

OpenIoT is categorised as a middleware platform which means there is no prebuilt application for end users. This also means that there is no deployment tool available for OpenIoT applications. It is up to the developers to use the provided APIs to build their own applications and deployment tools.

## 5.2.5 SensiNact

As sensiNact is deployed in the DS6 deployment site, the related deployment tools are those developed and used in this deployment site for the needs of the ACTIVAGE project.

### 5.2.5.1 Deployment tools around sensiNact

The DS6 ISE comprises three separate panels, with the first two for a residential and individual population, while the third is for a specialized collective residence. In this configuration, the DS6 has two different set-ups :

- In institution, the IoT suite is quite generic, with a high reproducibility for each rooms in each building. In addition, application are unique and can be deployed with very little personalization across all users. Finally, the cloud environment is existing, with identified procedures for deployment of new services and support for these services. These procedures are defined upon a medical context with well defined path for experimentation.
- In individual housing, the IoT suite has to be tailored in order to match the local constraints and opportunities. Indeed, each house is unique in both its design and its occupancy, devices shall be chosen given these differences. Cloud and application will be implemented on a new infrastructure, on which software components has to be deployed given WP3 recommendations.

#### 5.2.5.1.1 Device deployment tools

Prior to Activage developments, IoT devices are deployed manually. They are paired on-site to the gateway and if necessary between themselves. Their deployment is ideally made by an integrator in order to calibrate the different sensors given the service requirements.

In panel 1 and 2, a specific deployment procedure in two step will be used for the deployment, and a specific tool will be developed in order to assist this procedure. First step will be an audit in order to establish a summary recognition of the place: number of rooms, placing of the rooms, approximate sizes, location of some devices such as electrical meter, type of heating, etc. This information will be used for the preparation of the IoT suite in which each device will be pre-located in its future placing. Once the IoT suite is properly prepared, the second step consist of deploying the devices on-site and validate the overall deployment by testing individually each device and make sure data is properly forwarded to the gateway.

The device development tool planned for panel 1 and 2 will be based on a tablet for any installer or professional care-giver. It will be designed to be intuitive and easy to use by an untrained. The goal is to prevent any mismatch of the IoT kit toward the final user nor any misconfiguration of the devices to reach the objective service.

The device deployment tool will report on:

- In the audit / pre-deployment phase:
  - the approximate ground-plane of the housing, with approximate size and disposition of the room together.
  - equipment already in place, whether they are connected or not.
- In the deployment phase:
  - exact location of the device to be deployed
  - safety instruction if necessary to deploy the device
  - installer notice and user manual of the device in deployment

- specific instruction if needed, such as calibration
- security provisioning, if available
- on-site test of the device

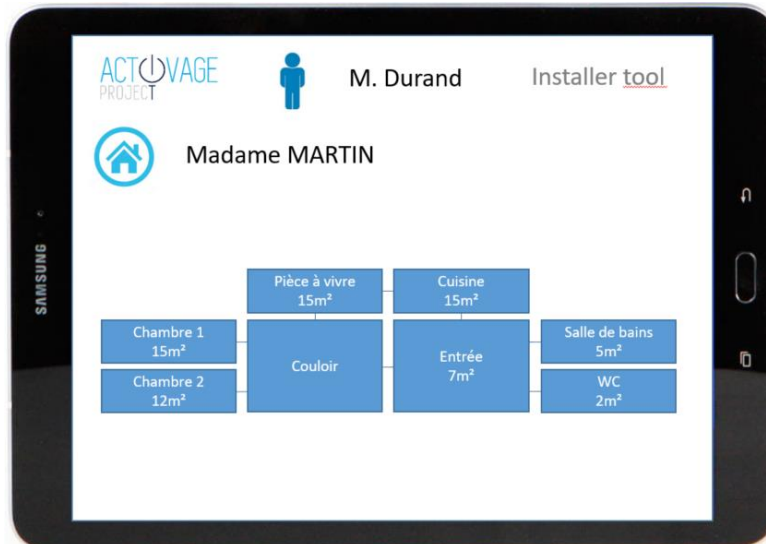


Figure 77. Installer tool mock up: discovering the ground plane of the installation



Figure 78. Installer tool mock up: discovering the existing asset, in this case the water counter and valve

The same tool can be further extended to the maintenance phase in order to access to the history of the home as well as to device's diagnostics.

In panel 3, device deployment is less exposed to human risks of errors due to its reproducibility, each room having the same configuration. The first step (audit) will be made once manually for the building and then a generic connected solution can be prepared for each room. The deployment of the devices can be made by or with support of the building technical manager using a generic deployment plan.



### 5.2.5.1.2 Gateway deployment tools

The gateway is a raspberry pi. The image that is flashed is already pre-configured with software such as middleware, firewall and tools needed for the functioning of the gateway. SensiNact is deployed using the repository online available on eclipse. It is then configured by developers in order to implement the corresponding bindings and rules running at the gateway level.

Existing deployment methods for the gateway, such as prebuild image and generic software containers that can be remotely configured, are sufficient enough at the scale of the deployment site. Thus, an issue is to handle the provisioning of security information such as unique identification and authentication data. At the time of this deliverable, open discussion are carried out in order to address this issue, with either a hardware dedicated device or with a specific tool.

### 5.2.5.1.3 Cloud deployment tools

As the DS6 ISE comprises three panels, it distinguishes two different architectures which on one side is defined by the CEA for panel 1 and 2 and on other side by Korian for panel 3. For this last, Korian has provided the server and granted access for developers in order to deploy tools and middleware. As for the gateway sensiNact is deployed using the eclipse platform and configured by the developers. This installation requires to be done once for the server and will be available for all of the building Korian benefits.

The Studio Web application embedded in the sensiNact platform can be used to check the status of deployed devices and services. This web application make it possible to localize devices through the ‘Navigator’ panel (cf. Figure 79). This Studio Web navigator gives the status of both installed devices and deployed services.

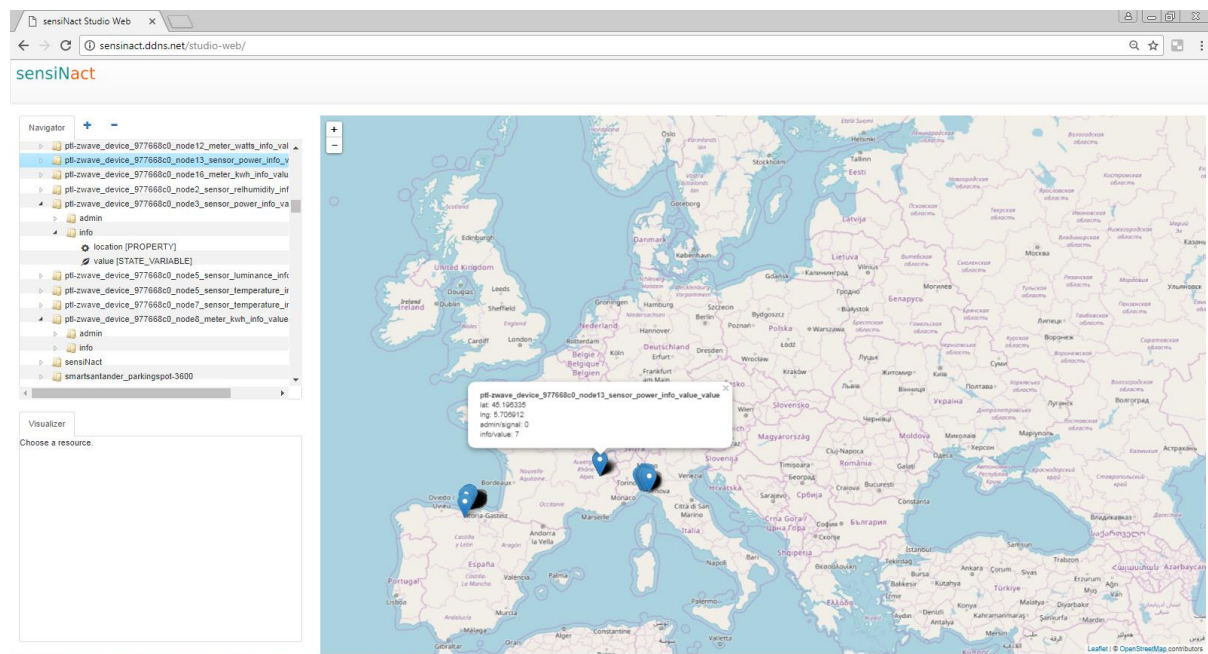


Figure 79: screenshot of the sensiNact Studio Web device and service navigator

### 5.2.5.14 Application deployment tools

There are four kinds of application provided by Technosens. One is for the resident called “e-resident” deployed on a tablet or smartphone such as the one for the family “e-lio family”. There are available on the play store. One interface of management is already deployed on the cloud. The last application is deployed as an .apk which is an android application for tablet or smartphone for nurses.

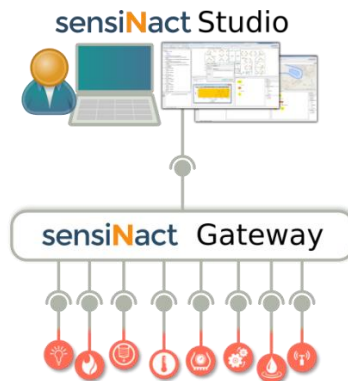


Figure 80: sensiNact studio tool overview

The sensiNact studio standalone software delivered with the sensiNact platform, as a tool on top of the sensiNact gateway API (see Figure 80), provides facilities for gateway monitoring and application creation.

sensiNact studio is a Eclipse-RCP based software containing views to monitor available devices, and an editor to create applications (see Figure 81).

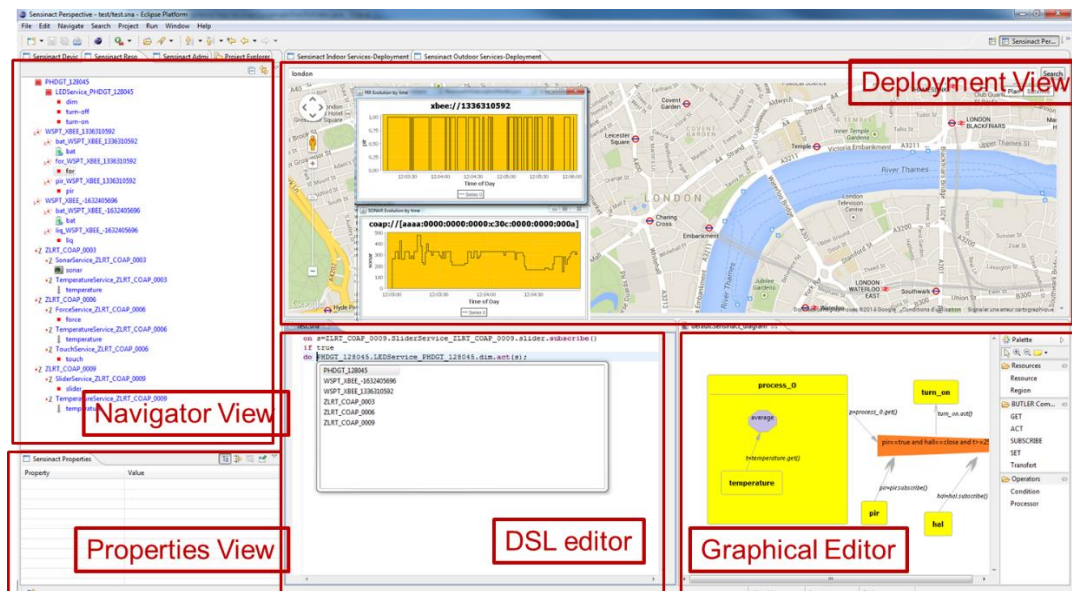


Figure 81: sensiNact studio view and editor components

Through the DSL<sup>38</sup> editor, it is possible to program Event-Condition-Action scripts (called “applications”) in a dedicated language.

<sup>38</sup> Domain Specific Language

```

ON presence=PIRService.pir.subscribe()
IF presence==true
DO LightService.lightOn.act();
ELSE
DO LightService.lightOff.act();

ON presence=during(PIRService1.pir.subscribe()==true,
                    PIRService2.pir.subscribe()==true,
                    3)
IF presence==true
DO LightService.lightOn.act();
ELSE
DO LightService.lightOff.act();
    
```

Figure 82: samples of DSL script in sensiNact studio

Auto-completion, type checking, validation features are provided by the DSL editor (see Figure 83).

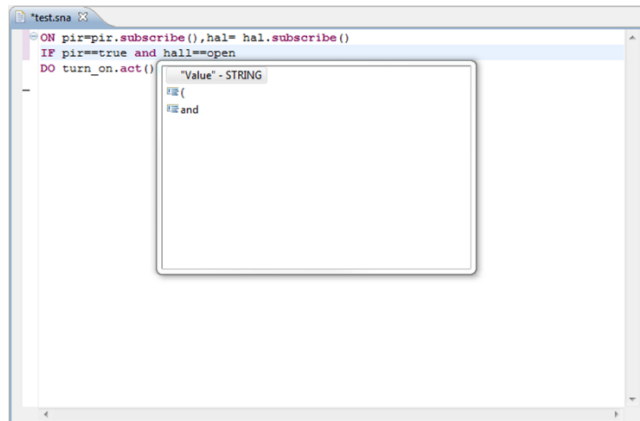


Figure 83: auto-completion in sensiNact studio DSL editor

### 5.2.5.2 Mapping between deployment tools requirements and modules

Table 35: Mapping between DS6/sensiNact and ACTIVAGE deployment tools.

sensiNact deployment tool	Corresponding ACTIVAGE deployment tool	Can the tool be used in AIOTES?	
sensiNact Studio and studio web	IoT infrastructure management tools / Device and Service manager	Yes	No
		<b>How?</b>	<b>Why?</b>
		The web application will be used as device managing tool at deployment phase in DS6, and also may be used as monitoring elements in the AIOTES management module.	
Installer tool	Distribution/deployment Component configuration and update	Yes	No
		<b>How?</b>	<b>Why?</b>
		The Installer tool can be shared to other DS with minor adaptations	

### 5.2.6 FIWARE

As described in Section 0, FIWARE offers Generic Enablers, which are a set of building blocks that ease creation of smart Internet Applications. So as to use GEs by means of their specific APIs, it is necessary to deploy a dedicated GE instance. So as to do that, there are two available options.

On the other hand, there is a dedicated section in the FIWARE Catalogue dedicated to provide access to available instances of each Generic Enabler. This section is located in the Instances tab of the catalogue entry where the instances are freely available without restrictions.

The second option consists of creating your own dedicated instances and using it. So as to create the instances, it should be followed the instructions provided in the tab "Creating Instances" of the corresponding Generic Enabler section in the FIWARE Catalogue. For each particular generic enabler instance it is offered a step-by-step guide to deploy the instance.

It must be pointed out that FIWARE does not provides any tool for management, installation or maintenance. For the deployment of the FIWARE platform, Docker and/or Virtual Machine can be used. This two tools are platform independed and has already been described in Section 5.2.1.

### 5.2.6.1 Mapping between deployment tools requirements and modules

Table 36: Mapping between FIWARE and ACTIVAGE deployment tools.

FIWARE deployment tool	Corresponding ACTIVAGE deployment tool	Can the tool be used in AIOTES?	
<a href="#">Docker</a>	Platform independent Available Deployment tools / Docker	Yes	No
		<b>How?</b>	<b>Why?</b>
		Docker allows to package up any ACTIVAGE application or application needed by ACTIVAGE with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.	
Virtual Machine	Platform independent Available Deployment tools / Virtual Machine	Yes	No
		<b>How?</b>	<b>Why?</b>
		For better performance, durability and host ability ACTIVAGE tools can be build as Virtual Machines.	

## 5.2.7 IoTivity

### 5.2.7.1 Easy Setup

One deployment tool of IoTivity is [Easy Setup](#). Easy Setup is a primitive service layer developed using native platform and IoTivity APIs for making UI-less unboxed devices to be easily connected to the end user's IoTivity network seamlessly, thus enabling the devices to be part of the IoTivity network in a user friendly manner. Specifically, user can transfer a bunch of essential information to the unboxed devices in easy setup phase, which the information includes: WiFi AP connection information needed for the device to connect to Home AP and device configuration settings. Additionally, user can provide a [cloud](#) access information to the devices so that they can register them to an IoTivity cloud server (CoAP Native Cloud) and user can access them via IoTivity cloud even from a distance.

There are three types of roles defined in Easy Setup for various devices involves in Easy Setup method. These roles are; Enrollee, Mediator and Enroller and their architecture view is depicted with below diagram:

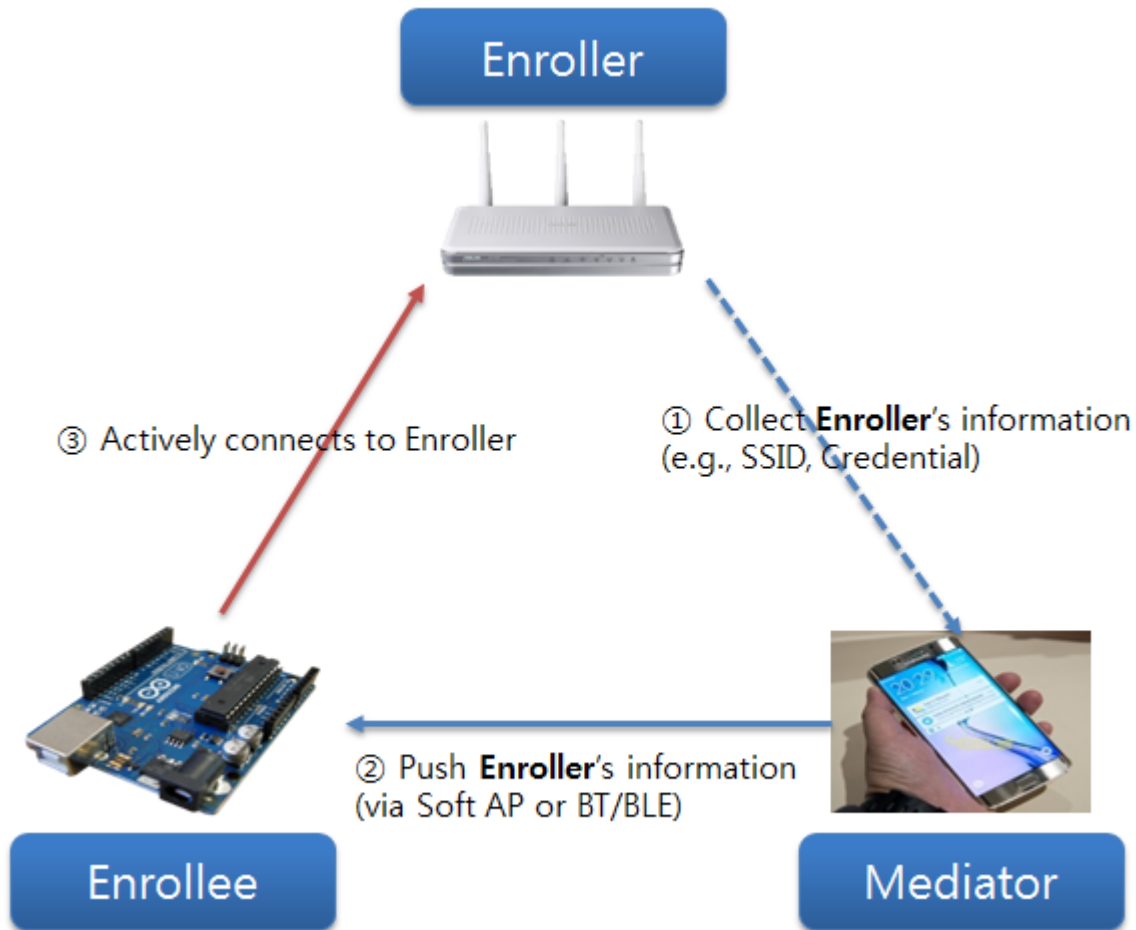


Figure 84: IoTivity Easy Setup

Easy Setup is used to transfer data through the Mediator (e.g. a smartphone) to the enrollee. After running the application Enrollee receives WiFi Properties, SSID, password, Auth type, Cloud properties etc.

### 5.2.7.2 Device Management

The IoTivity device management is a deployment tool developed by CERTH within the context of ACTIVAGE, in order to handle device registration and update. Although it has been used for IoTivity applications, it is implemented as a set of RESTful web services, which can be readily used by any IoT platform within ACTIVAGE. The Device Management tool has not yet been documented elsewhere, thus it is hereby presented in detail, in order to make all relevant information available to the reader.

IoTivity is a reference implementation of OCF specification. OCF specification defines a set of core Device Types and their required Resource Types. A Resource is the minimal interoperable component in OCF. It has a URI and a collection of Properties. The following Properties are mandatory: Resource Type ('rt') e.g. 'oic.r.light', Resource Interface(s) ('if')

e.g. 'oic.if.a', Resource Properties with associated key/value pairs e.g. 'status: binary'. Resources provide operations that comply with the OCF Interaction Model – CRUDN (CREATE, RETRIEVE, UPDATE, DELETE, NOTIFY). Resources can be also mapped to models from non-OCF ecosystems using Derived Models, which describe the mapping of resources between External Models (User-defined) and Native Models (OCF).

For the devices of smart home scenario the related IoTivity Resources have been created, providing the required operations, and are exposed by the IoTivity Server. These resources need to communicate with physical devices or sensors through a gateway in order to update their resource representation. In order to ease the work of development for the communication layer in IoTivity Server and make it more configurable, we have implemented a device management mechanism.

The mapping of the actual hardware with the resources is done through a device management mechanism, which enables the system administrator to register specific information for a device/sensor (e.g. the mac address of a device) that will be used for the communication of the resources with the corresponding devices/sensors.

The device management system holds the devices information in a Relational Database, which is used to update the IoTivity Server with the mapping between physical devices and resources.

A device has several properties. Some of them are mandatory and some are optional, depending on the communication type. The following table shows this in detail.

Table 37: Device parameters according to their communication type

Device Properties	Communication Type			
	Bluetooth	Wifi	Zigbee	ZWave
Device Id	mandatory	mandatory	mandatory	mandatory
Name	mandatory	mandatory	mandatory	mandatory
Type	mandatory	mandatory	mandatory	mandatory
Address	mandatory	mandatory	mandatory	mandatory
Password	optional	optional	optional	optional
Manufacturer	optional	optional	mandatory	mandatory
Model	optional	optional	mandatory	optional

The device management mechanism supports four main functionalities:

- Register a device
- Update a device
- Unregister a device
- View all registered devices

These functionalities are implemented by corresponding operations of the *Registration Service*. The operations are described in the next sub-chapters 6-6.

The definition of the four service operations implemented in Java, is shown below.

```
@Path("/service")
public class Webservice {

    @POST
    @Path("/register")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response generateResponse(RegInfo info) throws Exception {
        WorkflowClass newClass = new WorkflowClass();
    }
}
```

```

        Response response = newClass.parseResponse(info);
        return response;
    }

    @PUT
    @Path("/update")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public ResponseUpdate generateResponse(UpInfo info) throws Exception {
        WorkflowClass2 newClass = new WorkflowClass2();
        ResponseUpdate response = newClass.parseResponse(info);
        return response;
    }

    @DELETE
    @Path("/remove")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public ResponseDelete generateResponse(DelInfo info) throws Exception {
        WorkflowClass3 newClass = new WorkflowClass3();
        ResponseDelete response = newClass.parseResponse(info);
        return response;
    }

    @GET
    @Path("/getall")
    @Produces(MediaType.APPLICATION_JSON)
    public ResponseGet generateResponse() throws Exception {
        WorkflowClass4 newClass = new WorkflowClass4();
        ResponseGet response = newClass.parseResponse();
        return response;
    }
}

```

The IoTivity server requests to get all registered devices at start up. Whenever there is a request for device registration, update or removal, the database and the IoTivity server are updated accordingly by the related services. Apart from the services a dedicated endpoint has been implemented also on IoTivity Server side in order to handle any addition, deletion or update of devices. The following sequence diagram shows this process.

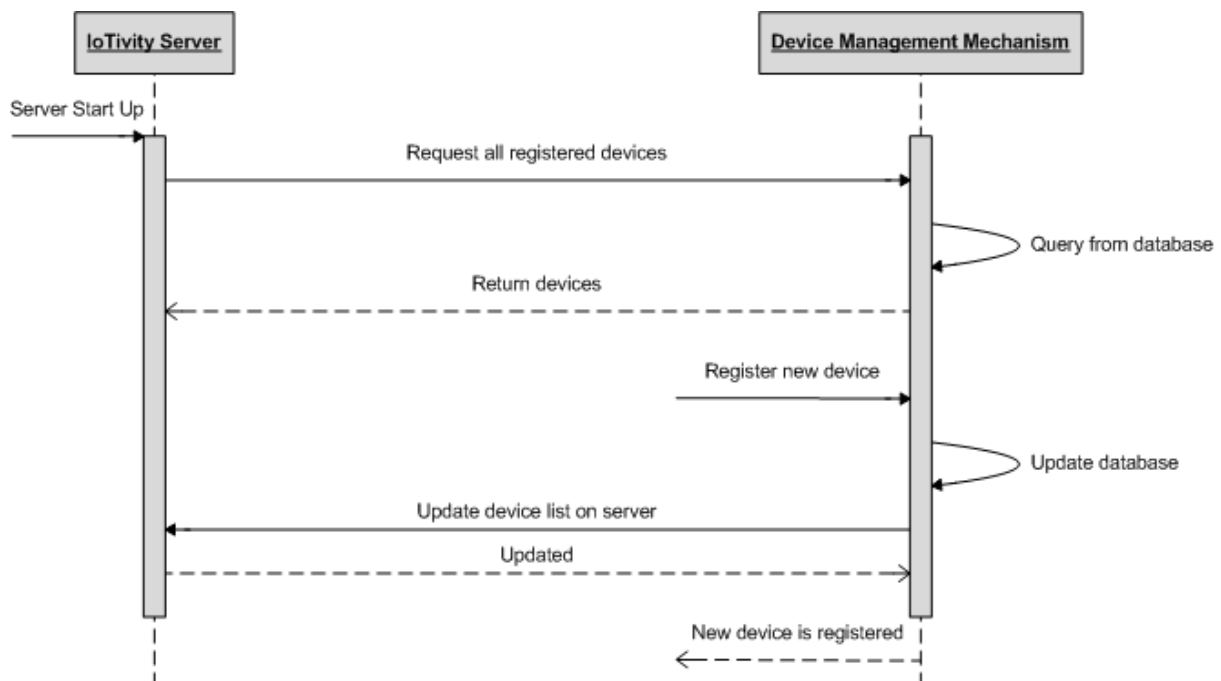


Figure 85: Device Management Mechanism for IoTivity Platform

### Device Registration

When a new device is deployed it needs to get registered before it can communicate with IoTivity Server. The installer or administrator of the device will need to enter the appropriate information to the system.

The sequence diagram below illustrates the device registration procedure that takes place after a physical device deployment and the software components that participate in this process. This procedure takes place upon request.

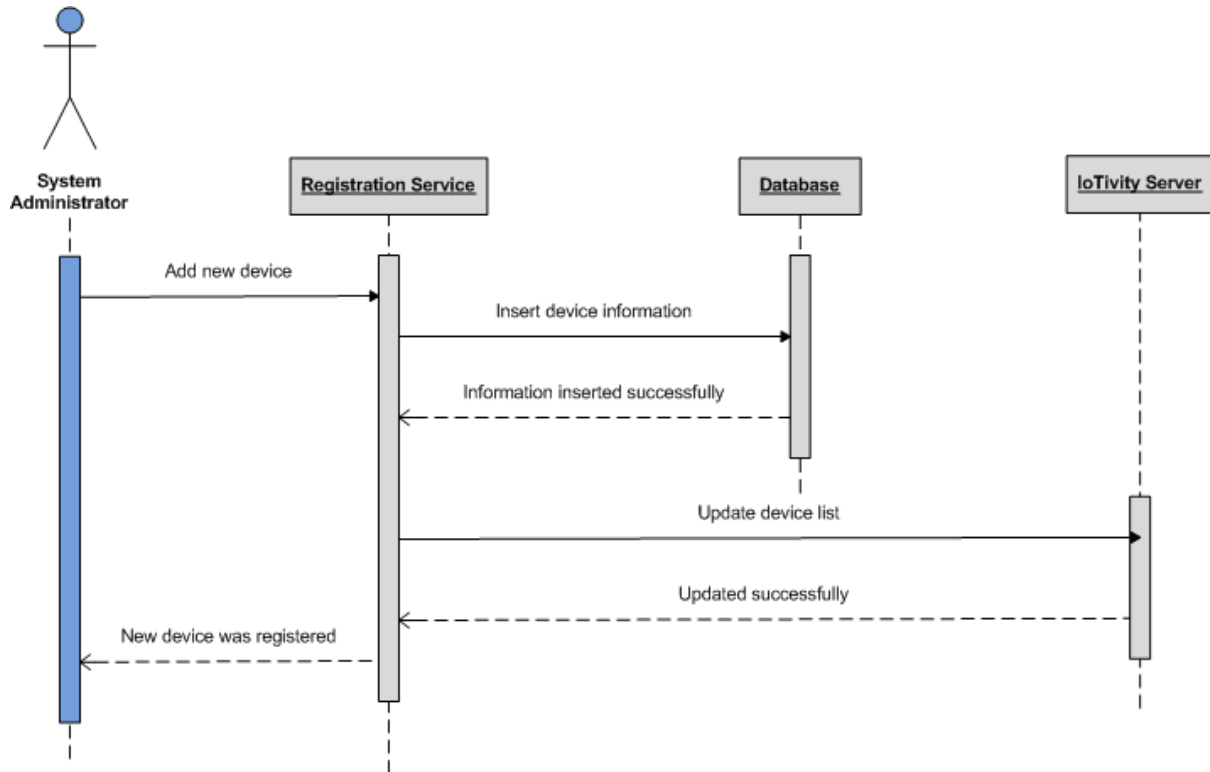


Figure 86: Sequence diagram for device registration for IoTivity Platform

For this functionality, a POST operation is used, with the appropriate parameters:

Resource URL: “/service/register”

Input example

```

{
  "name": "Bloodpressure
monitor",
  "type": "bloodpressure",
  "mac": "00:12:A1:B0:77:AE",
  "password": "123"
}
    
```

Output

```

{
  "message": "The device with name: Bloodpressure monitor was registered."
}
    
```



### Device Update

The sequence diagram below illustrates the steps that take place for the update of an already registered device’s property and the software components that participate in this process. This procedure takes place upon request.

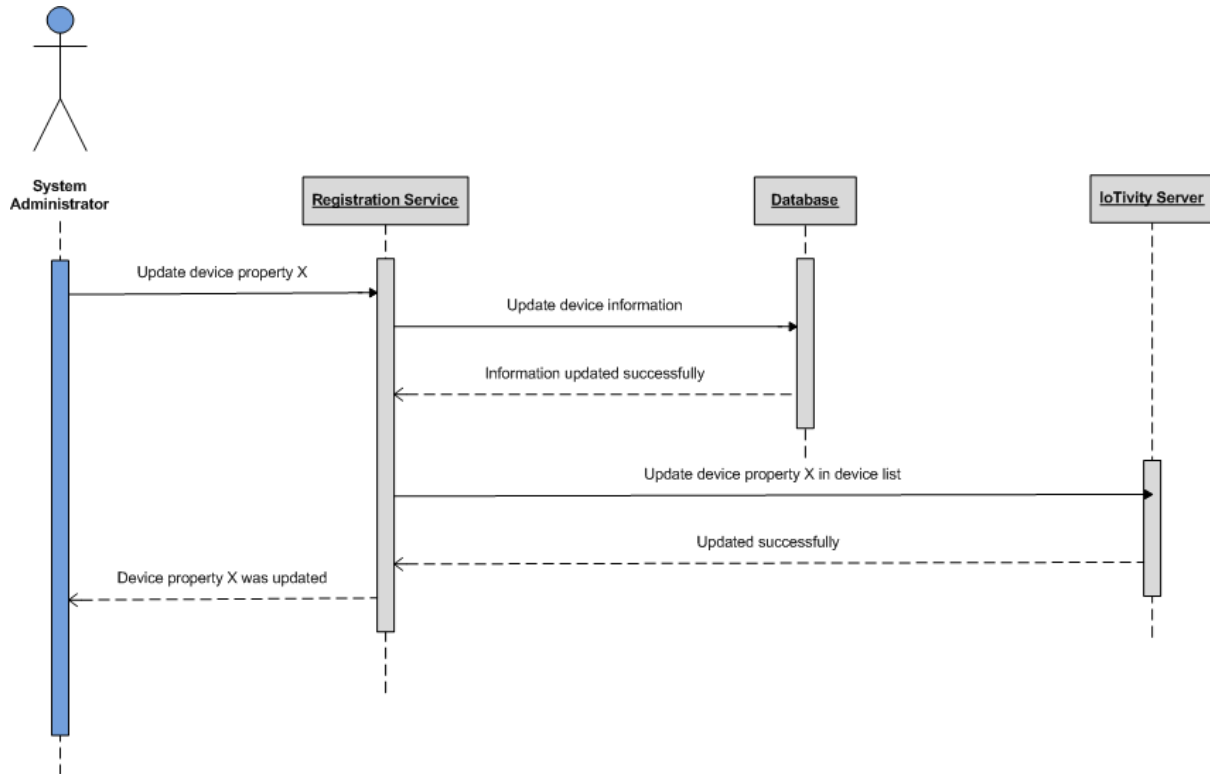


Figure 87: Sequence diagram for device update for IoTivity Platform

For this functionality, a PUT operation is used with the following parameters:

Resource URL: “/service/update”

Input

```

{
  "name": "Bloodpressure monitor",
  "password": "789"
}
    
```

Output

```

{
  "message": "The device with name: Bloodpressure monitor was updated."
}
    
```

### Device Removal

The sequence diagram below illustrates the steps that take place for the removal of an already registered device and the software components that participate in this process. This procedure takes place upon request.

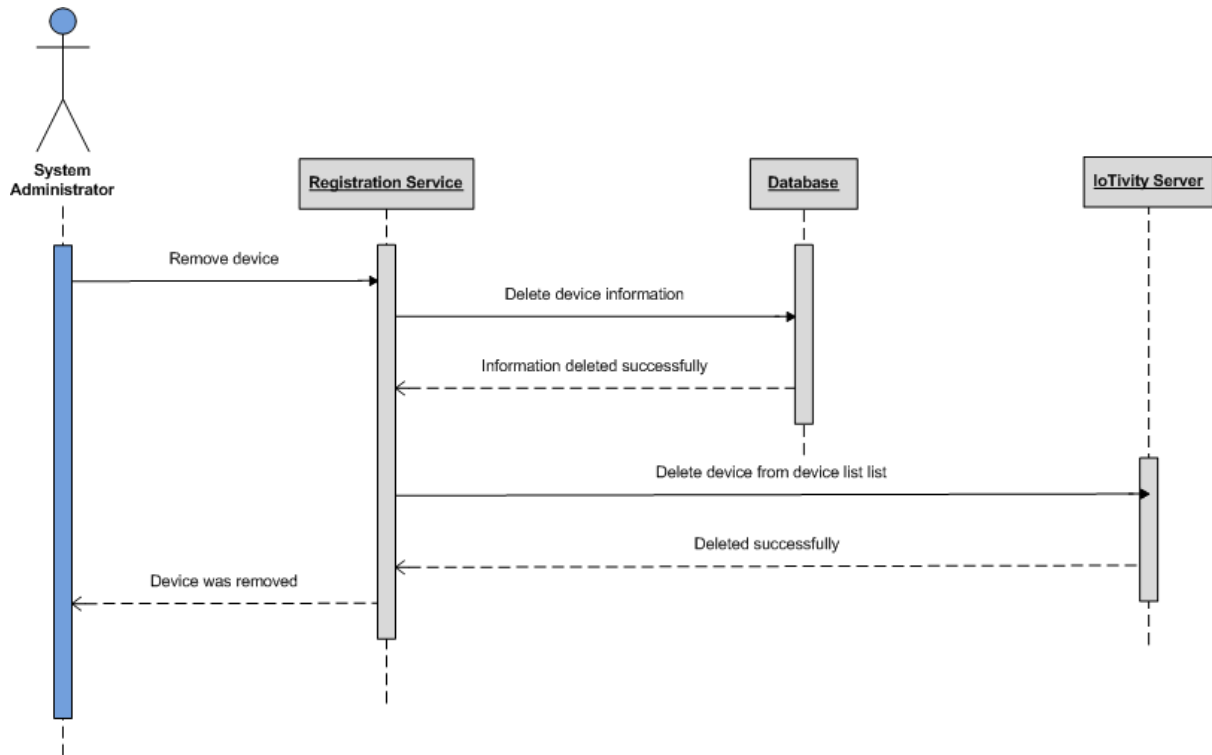


Figure 88: Sequence diagram for device removal for IoTivity Platform

For this functionality, a DELETE operation is used with the following parameters:

Resource URL: "/service/remove"

Input

```
{
  "name": "Bloodpressure monitor"
}
```

Output

```
{
  "message": "The device with name: Bloodpressure monitor was deleted."
}
```

### Retrieve Registered Devices

The sequence diagram below illustrates the steps that take place in order to retrieve the list of the registered devices.

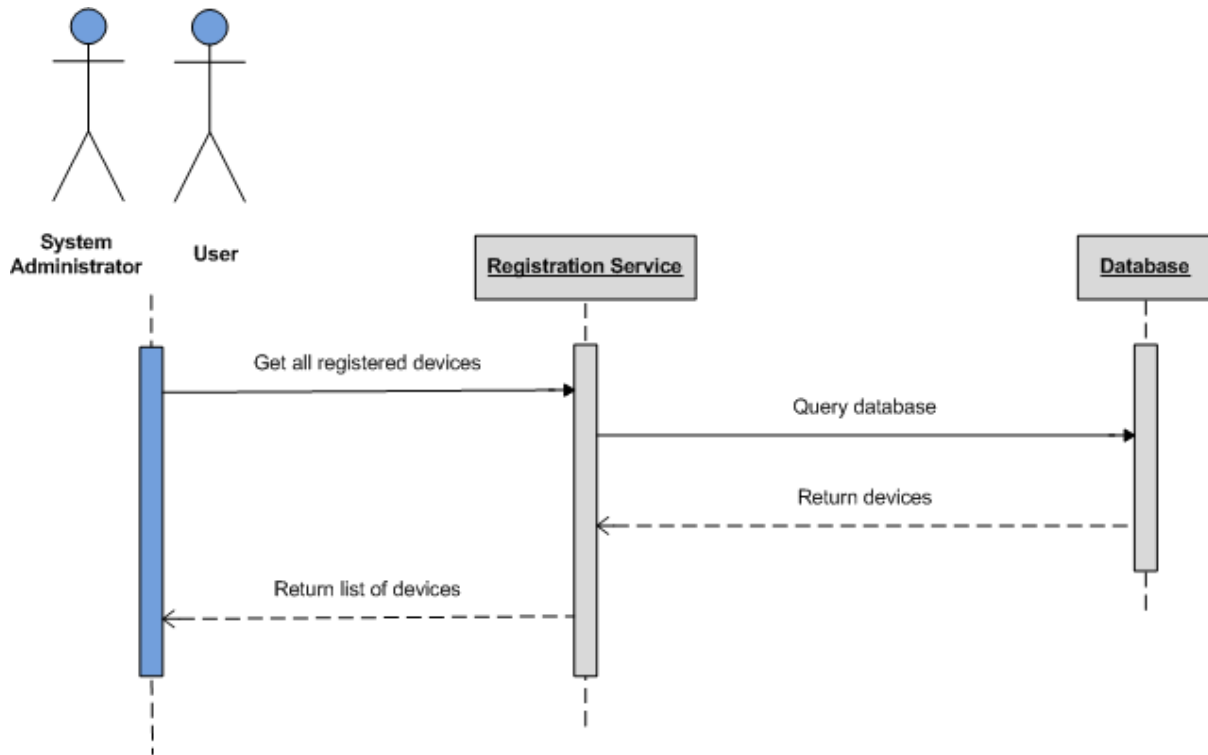


Figure 89: Sequence diagram for getting registered devices for IoTivity Platform

For this functionality, a GET operation is used with the following parameters:

Resource URL: "/service/getall"

Input Example

No Input is required

Output Example

```

{
  "devices": [
    {
      "ip": null,
      "mac": "00:12:A1:B0:77:AE",
      "manufacturer": null,
      "model": null,
      "name": "Bloodpressure monitor",
      "password": "123",
      "type": "bloodpressure"
    },
    {
      "ip": null,
      "mac": "00:12:A1:B1:04:CB",
      "manufacturer": null,
      "model": null,
      "name": "Thermometer1",
      "password": "123",
      "type": "thermometer"
    }
  ]
}

```

### 5.2.7.3 Mapping between deployment tools requirements and modules

Table 38: Mapping between IoTivity and ACTIVAGE deployment tools.

IoTivity deployment tool	Corresponding ACTIVAGE deployment tool	Can the tool be used in AIOTES?	
<a href="#">Easy Setup</a>	Deployment management tools / <a href="#">Component configuration</a>	Yes	No
		<b>How?</b> The tool's user-friendly configuration interface and the functionality for transferring essential information to the devices can be used as conceptual basis for the design of the ACTIVAGE component configuration tool.	<b>Why?</b> The tool offers a C++ SDK, which cannot easily be used to implement ACTIVAGE web-based tools.
Device Management Tool	IoT infrastructure management tools / <a href="#">Device manager</a>	Yes	No
		<b>How?</b> The tool is implemented as RESTful web services, which can be used directly in ACTIVAGE, with few modifications.	<b>Why?</b>

## 5.2.8 SeniorSome

The existing development and deployment tools are the same as described in Section 4.2.8 but without AIOTES connectivity or direct connectivity to protocols described in general sections.

### The mapping to Section 5.1. is:

- Services layer: development, deployment, analytics, data: The development api can be used with different applicable tools.
  - For deploying the SeniorSome backend can be used through the deployment api.

### Aiotes API:

- The AIOTES API is included in SeniorSome API:s.

### Semantic interoperability layer / Broker and Platform layer:

- The Broker can utilize the SeniorSome broker through the SeniorSome API:s like the Broker Api.

The deployment tools mapping to AIOTES is described in the below table. Further informations is maintained at <https://api.seniorsome.net>.

Table 39: Mapping between SeniorSome and ACTIVAGE deployment tools.

Seniorsome deployment tool	Corresponding ACTIVAGE deployment tool	Can the tool be used in AIOTES?	
SeniorSome API-based backend tool	Deployment management tools / <a href="#">Component configuration</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		The tool can best be used for AIOTES through API integration. When using directly it requires a seniorsome compliant device. However through the API configuration other devices like the bridged devices can possibly be deployed.	
Seniorsome Device Management Tool	IoT infrastructure management tools / <a href="#">Device manager</a>	Yes	No
		<b>How?</b>	<b>Why?</b>
		This tool can be used through API-configuration and possibly through an application layer integration.	
Documentation	Support	As a part of the AIOTES support documentation.	

## 5.2.9 Summary of existing tools

This section presents a synthesized set of tables providing a quick overview of the deployment tools for each of the platform designated to be interoperable with AIoTES framework. This section main purpose is to provide a interested developer with a vision of the different tools provided regarding the platform and, at the same time, serve as a listing of the desirable functionality over the AIoTES framework.

### 5.2.9.1 Platform-specific deployment tools comparison

Table 40: Deployment technologies per platform

Platform	Orchestrated deployment	Package technology	Deployment and configuration technology	Accesible service repository	Marketplace	GUI Tool for deployment and configuration
universAAL	Yes	Pax Runner	karaf	Yes, but not specified	UniversAAL Control Center (uCC)	universAAL Control Center (uCC)
SOFIA2	Yes	Not specified	Not specified	Not specified	Not specified	Management Software Configuration
OpenIoT	Yes	docker and virtual machine	new: virtual machines, future: docker	Not specified	Request definition tool	Yes
SensiNact	Not specified	Virtual machine	Not specified	Not specified	Not specified	sensiNact studio
FIWARE	Yes	docker and virtual machine	docker and docker-compose	docker-hub, fiware-lab	fiware marketplace	fiware-lab
IoTivity	Yes	docker	docker-	Not	Not specified	Cloud

			compose	specified		connection tools
SeniorSome	Not available	Not Available	Not available	Not available	Not available	Not available

Table 41: Deployment tools for device configuration

Platform	Provides device deployment and configuration?	Provides status monitoring and software updates on physically deployed devices
universAAL	Not specified	Not specified
SOFIA2	Yes - Management software configuration	Yes - Easy Setup tool
OpenIoT	Yes - Request definition tool	Yes - Request definition tool
SensiNact	No - manual pairing	No - manual pairing
FIWARE	Yes - IDAS and Orion	Yes - IDAS and Orion
IoTivity	Yes - Easy Setup tool	Yes - Easy Setup tool
SeniorSome	Not available	Not Available

### 5.2.9.2 Mapping between platform-specific deployment tools and proposed ACTIVAGE deployment tools

Table 42: IoT infrastructure management tools

Platform	Device manager	Service manager	Semantic auto-discovery platform	Benchmarking
universAAL	Not specified	Not specified	Not specified	Not specified
SOFIA2	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
OpenIoT	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SensiNact	sensiNact Studio and studio-Web	sensiNact Studio and studio-Web	No tool matching AIOTES requirements	No tool matching AIOTES requirements
FIWARE	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
IoTivity	Yes - Device Management tool	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SeniorSome	Not available	Not Available	Not available	Not available

Table 43: Deployment management tools

Platform	Deployment manager	Component configuration	Maintenance panel	Update manager
universAAL	Not specified	Not specified	Not specified	Not specified
SOFIA2	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
OpenIoT	Yes - Virtual machine and docker manager	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SensiNact	Virtual machine	No tool matching AIOTES requirements	No tool matching AIOTES requirements	No tool matching AIOTES requirements
FIWARE	Yes - Virtual machine	No tool matching	No tool matching	No tool matching

	and docker manager	AIOTES requirements	AIOTES requirements	AIOTES requirements
IoTivity	No tool matching AIOTES requirements	Yes - Easy setup	No tool matching AIOTES requirements	No tool matching AIOTES requirements
SeniorSome	Not available	Not Available	Not available	Not available

## 5.3 Development within AIOTES

### 5.3.1 Functional description

The objective of this section is to provide a detailed description of the tools that are going to be developed and presented as deployment tools in the scope of the ACTIVAGE project, all this together with an action plan for the development of the platform based on an analysis of the dependencies between the different modules that compose it. These tools come from the AIOTES framework requirements, the different use cases taken and explained in Section 3.3, the architecture description done in Section 5.1, the deployment tools specification done for each platform in the previous section and of course the attempt on the project partners to advance the needs of third parties developers.

As can be seen in Figure 90 the deployment tools can be separated regarding the IoT infrastructure and services deployed in: the set of IoT services that are going to be deployed in the AIOTES framework southbound layer, and deploy, discovery and manage of services running over the framework. At this point is important to remark that the section describe only tools presented on top of the framework, even when platform specific tools and functions are mentioned or treated the purpose is to provide this capabilities through a suite of centralized and agnostic functions on services. For a better understanding of the different services a brief description is provided:

IoT infrastructure management, compile a set of functions that, even when are platform specific, must be presented in a platform-agnostic way. That is providing a reduced set of functions that can be transformed and processed by the interoperability layer to allow main functionalities, as the registration, discovery or manage of new devices or services, from a platform independently definition to a platform specific function. The tools need cover also functionalities like configuration and security, both considered key points in the project.

Distribution and deployment cover the set of functions that have to be presented as the named semantic auto-discovery platform. This platform must assure a complete support to the overall deployment process and also must handle semantically enhanced discovery of existing services by using semantic query mechanisms on existing and newly created semantic models.

According to the functional descriptions and tools identification provided in Section 5.1 the Figure 90: Deployment tools in AIOTES high level architecture is created trying to put in place the deployment tools in the complete AIOTES architecture. In the following sections each tool will be analysed exposing its concrete functionality and deploy and presentation method of the tool.

The interest, from the project point of view, is not to start from scratch but reuse all valid platform-specific tool developed, at least as much as it can be used. In this way, the previous sections turn out to be extremely important in the planning and deployment of the different tools outlined in Architecture section.

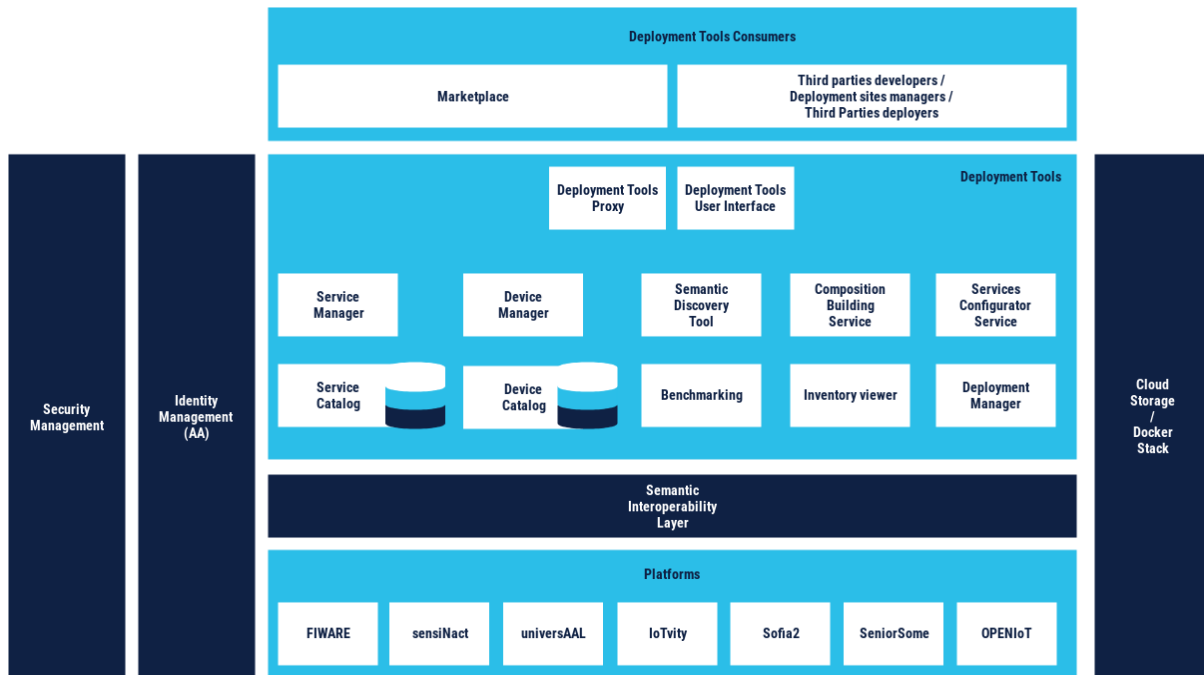


Figure 90: Deployment tools in AIOOTES high level architecture

## 5.3.2 Deployment tools description

### 5.3.2.1 Deployment Tools Proxy

#### 5.3.2.1.1 Functional description

This service provides a unique point of access to the deployment tools functionality. Its purpose is to limit and control the access to certain operations in the system and, at the same time, serves like a mechanism of dependency reduction between the rest of the tools.

Even when the deployment tools could work without this tool, its use provides simplicity by processing the complex operations maintaining this complexity hidden to the end users. The tool may be developed stateless assuring its replicability behind a load balancer in a cloud environment.

#### 5.3.2.1.2 Deploy and presentation method

The service must provide a REST-based API with the main functions discussed throughout this document.

Once the service has been developed and tested the service must be containerized and presented as a docker container ready to be deployable. Together with the tool an API, functionality and deploy documentation is required.

### 5.3.2.2 Deployment Tools User Interface

#### 5.3.2.2.1 Functional description

The user interface should provide a simple, intuitive and friendly way of interaction with the deployment tools for technical and non-technical users. This tool should represent the overall functionalities covered by the deployment tools. Capabilities like:



- a web semantic query engine, with historic, a query helper builder, some samples or change between ontology raw data and a more readable view will be really helpful;
- a building block composer, a visual tool to create and compose services private networks;
- a service status dashboard, that could provide users with services status real time information.

Other functionalities, like services upload, edit and deploy could be provided through forms.

#### 5.3.2.2.2 Deploy and presentation method

As a web development the service will be deployed in a HTTP server, Nginx or Wildfly are possible solutions. This servers will be deployed in docker containers to assure the consistency of the entire system.

In order to retrieve information the user interface will use the Deployment Tools Proxy.

### 5.3.2.3 Device Manager

#### 5.3.2.3.1 Functional description

Device Manager covers the main functionalities related to a device registered in the framework. This service must provide methods to:

- device registration,
- provide device configuration,
- provide semantic specification,
- manage device status and operations.

#### 5.3.2.3.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless microservice container.

#### 5.3.2.3.3 REST-based methods

Device Manager must provide REST-based methods allowing application developers and deployers to:

- Register a new device in AioTES framework,
- Provide custom configuration by device,
- Provide semantic descriptor by device

#### 5.3.2.3.4 Sequence diagrams

For a better understanding of the functionalities provided by the component, the shows Figure 91 the operation flow and components interactions.

#### 5.3.2.3.5 User interface

A first graphical user interface is proposed in Figure 92.

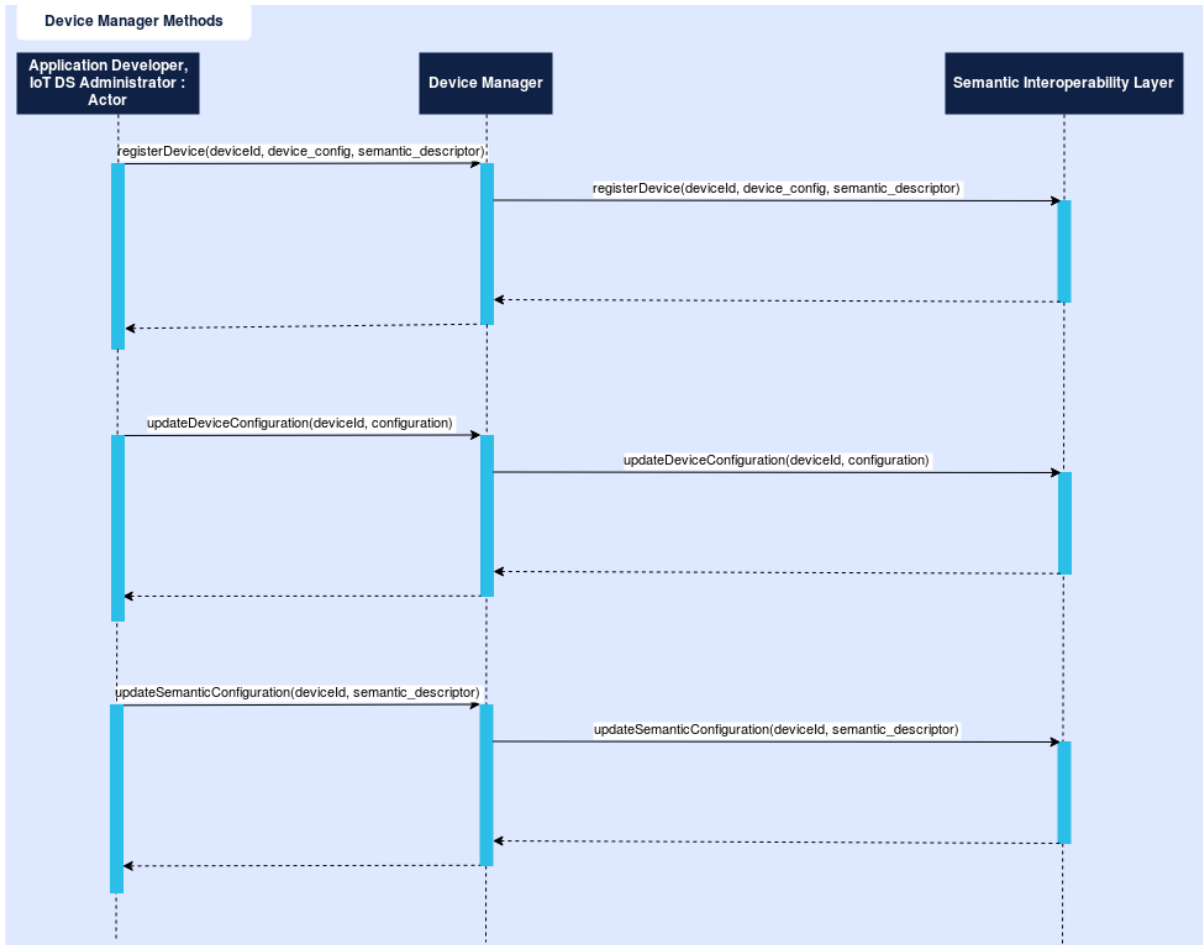


Figure 91: Device manager sequence diagram

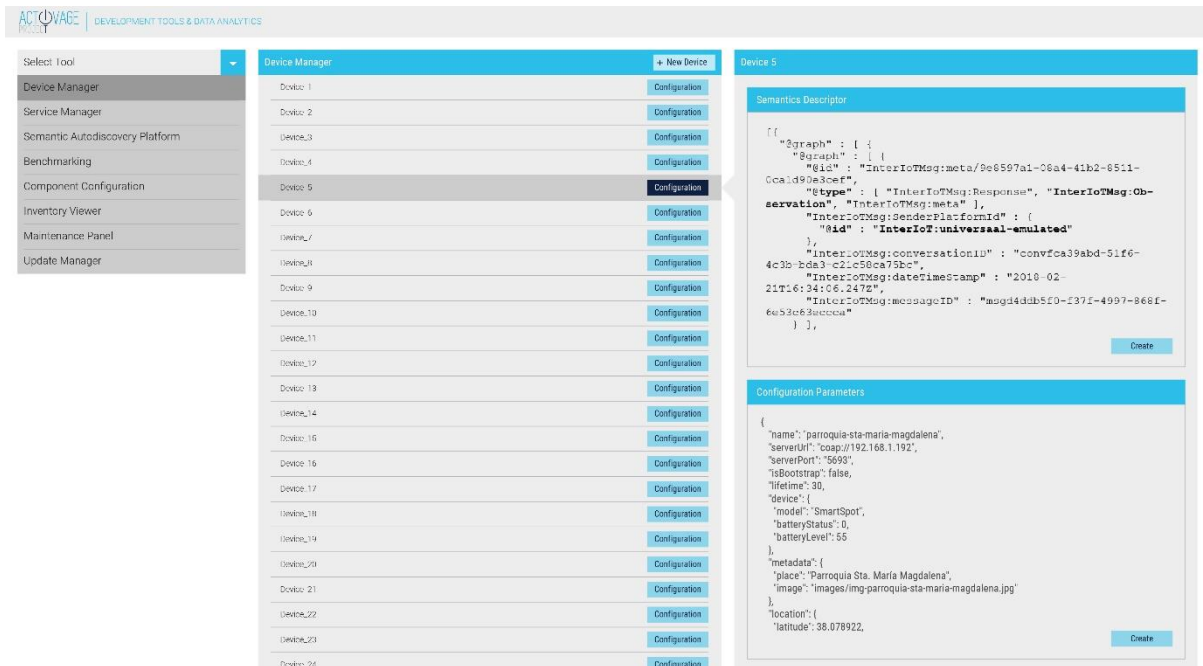


Figure 92: Device manager GUI mockup

## 5.3.2.4 Service Manager

### 5.3.2.4.1 Functional description

Service Manager covers the main functionalities related to a service or application published or not published yet but ready to be upload. This service must provide REST-based methods to:

- upload a service in the catalog,
- publish the service like a ACTIVAGE application or service,
- delete the application,
- configure a service descriptor specifying its semantic discovery or other interesting information,
- release a service new version,
- and retrieve services published information: like property or use statistics.

### 5.3.2.4.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless microservice.

### 5.3.2.4.3 REST-based methods

Service Manager must provide REST-based methods allowing applications developers to:

- Register a new service in the system,
- Provide service configuration in terms of inputs and outputs,
- Provide service semantic descriptor,
- Remove a service,
- Upgrade a service version

### 5.3.2.4.4 Sequence diagrams

For a better understanding of the service manager component and its interactions Figure 93 is presented. This figures describes the methods main flow in terms of components relationships.

### 5.3.2.4.5 User interface

A first graphical user interface is proposed in Figure 94.

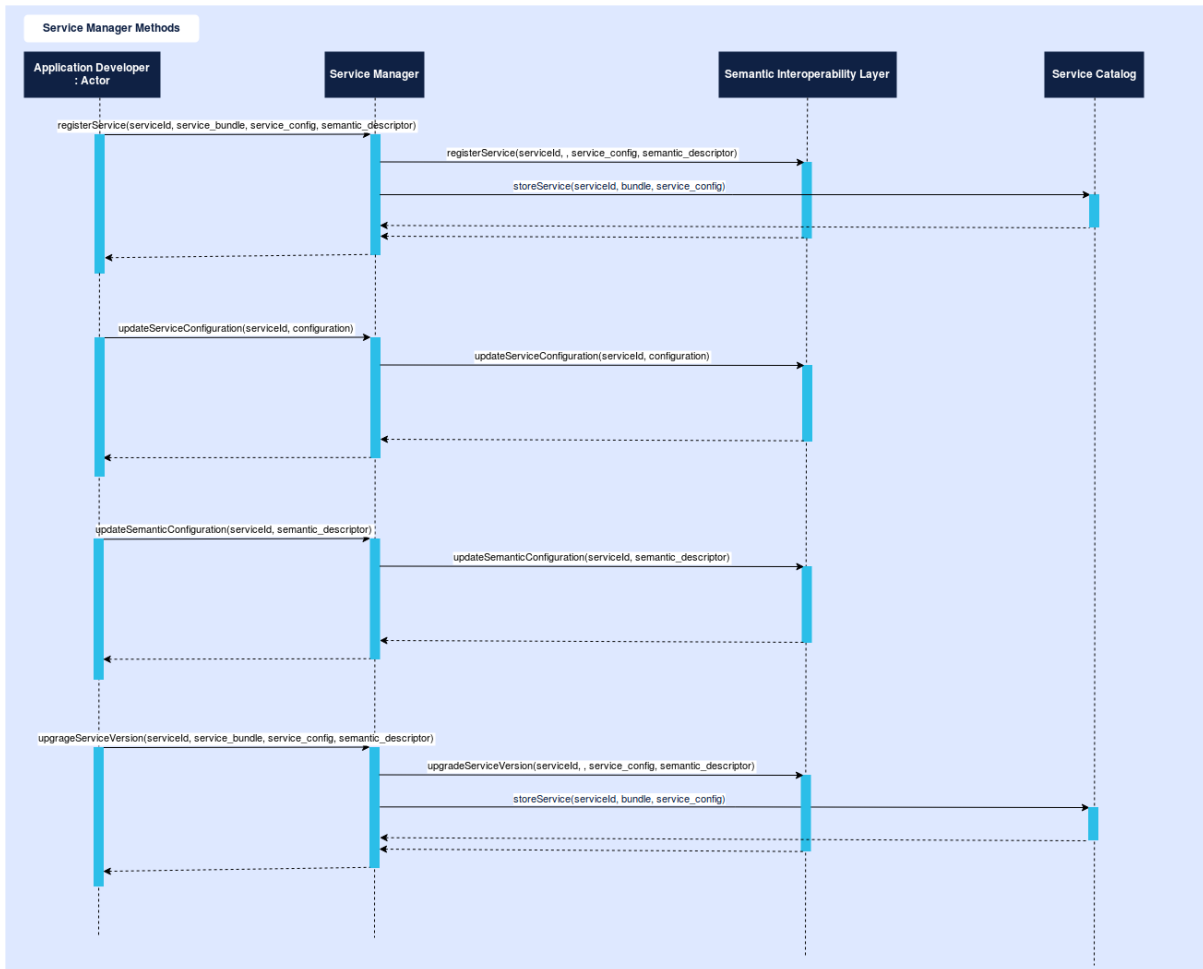


Figure 93: Service manager sequence diagram

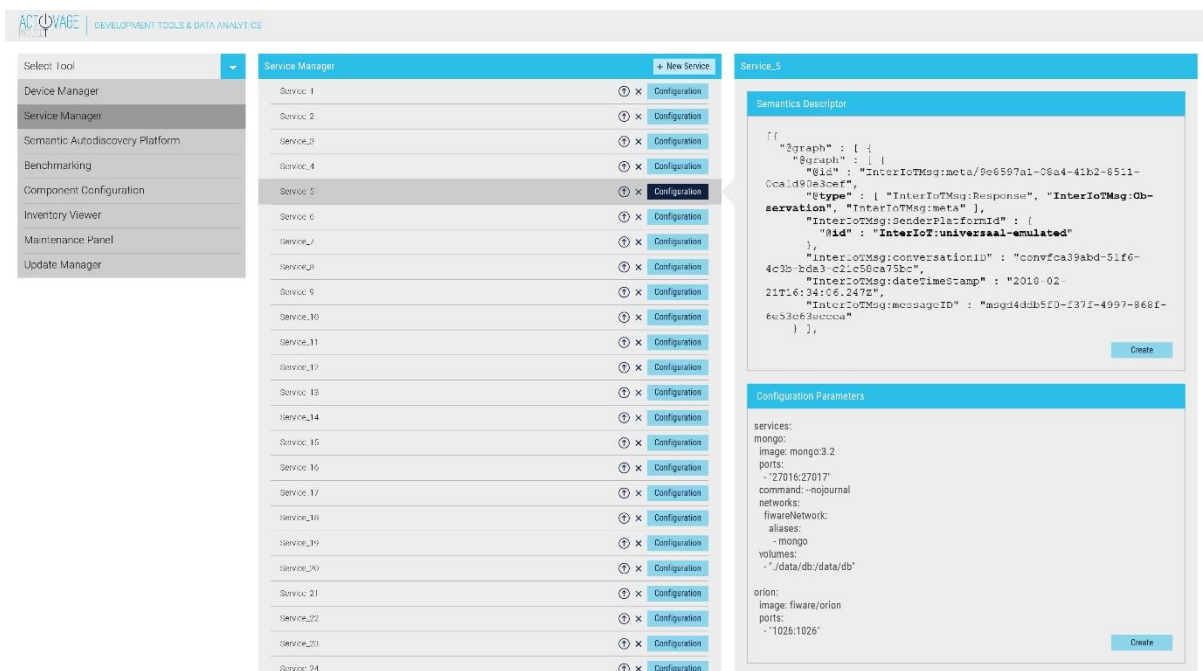


Figure 94: Service manager GUI mockup

## 5.3.2.5 Service and Device Catalog

### 5.3.2.5.1 Functional description

Service Catalog and Device Catalog are abstractions layers over storage systems allowing postpone and change the specific storage technology chosen. The service must provide only a couple of methods to retrieve and store a non-instantiated version of the services registered as ACTIVAGE applications.

Currently, in the Service Catalog case, the technology chosen to store and deploy all the services in the system is Docker, so the Service Catalog must provide this abstraction layer directly over a Docker Registry.

### 5.3.2.5.2 Deploy and presentation method

This service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless microservice deployed inside a docker container. At this point, in the service Catalog case, the real need of develop an abstraction layer over a well defined container (registry is made and containerized by the docker community) is being analysed.

## 5.3.2.6 Semantic Discovery Tool

### 5.3.2.6.1 Functional description

This service can handle semantically enhanced discovery of existing IoT services and applications on the client side by using semantic query mechanisms on existing and newly created semantic models with extra help of software agents.

### 5.3.2.6.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless microservice.

### 5.3.2.6.3 REST-based methods

Semantic auto-discovery platform must allow deployers and application developers to:

- Search for IoT devices and services based on the semantic output required,
- Search for services based on the semantic input and output,

### 5.3.2.6.4 Sequence diagrams

The Figure 95 provides a clear vision of the interactions between components in AIoTES when each method is called.

### 5.3.2.6.5 User interface

A first graphical user interface is proposed in Figure 96.

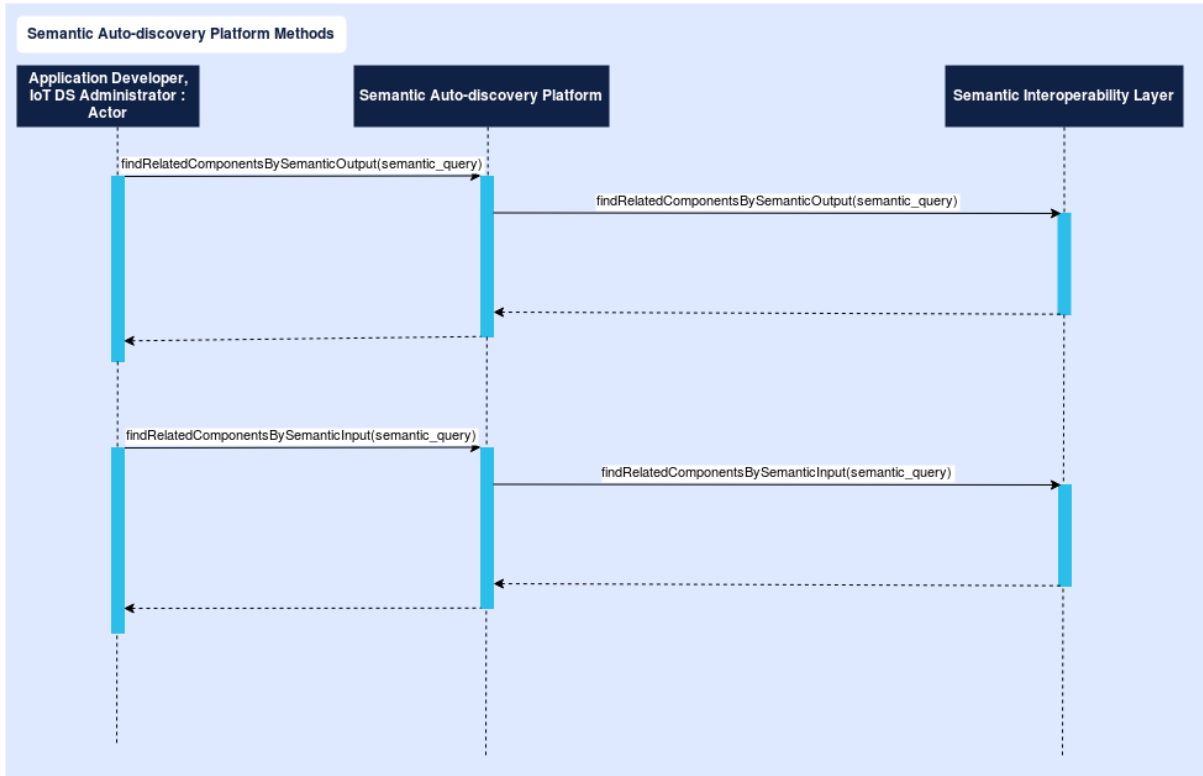


Figure 95: Semantic discovery tool sequence diagram

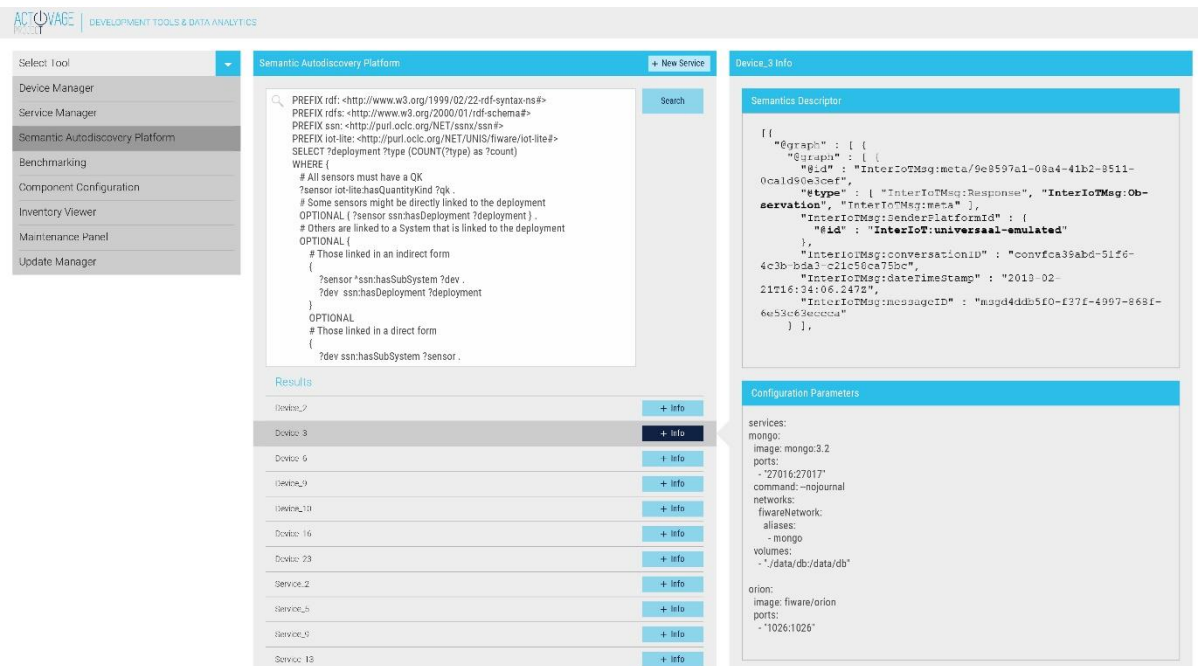


Figure 96: Semantic discovery tool GUI mockup

## 5.3.2.7 Composition Building Service

### 5.3.2.7.1 Functional description

Composition Building Service must be compatible and extend a similar development tool. While in the development tools a graphical service and a couple of functions must allow the composition of services in order to create new applications and services, this tool must add functionality allowing the description of private network services. That is, a small set of services that work together in an isolated scope.

### 5.3.2.7.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless microservice.

The service must assure the storage of the services composition and net descriptions in order to be able to monitor and recreate it on failure conditions

## 5.3.2.8 Benchmarking

### 5.3.2.8.1 Functional description

This service must provide real time information about the state of the services. Data like status, running or stopped; resources consumption or input and output traffic data through the service must be listed.

### 5.3.2.8.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless micro-service.

### 5.3.2.8.3 REST-based methods

Benchmarking must provide a set of well-defined REST-based methods in order to:

- Retrieve service's performance status,
- Retrieve service's security and privacy status.

### 5.3.2.8.4 Sequence diagrams

Figure 97 provides a visual description of the complete operation flow for both capabilities presented.

### 5.3.2.8.5 User interface

A first graphical user interface is proposed in Figure 98.

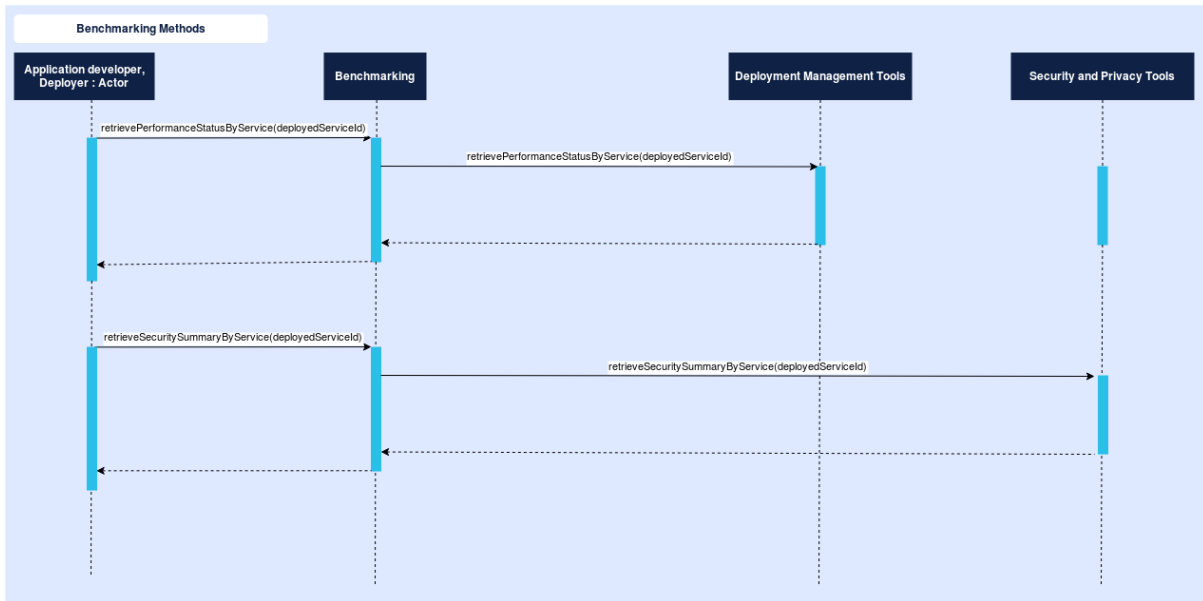


Figure 97: Benchmarking sequence diagram

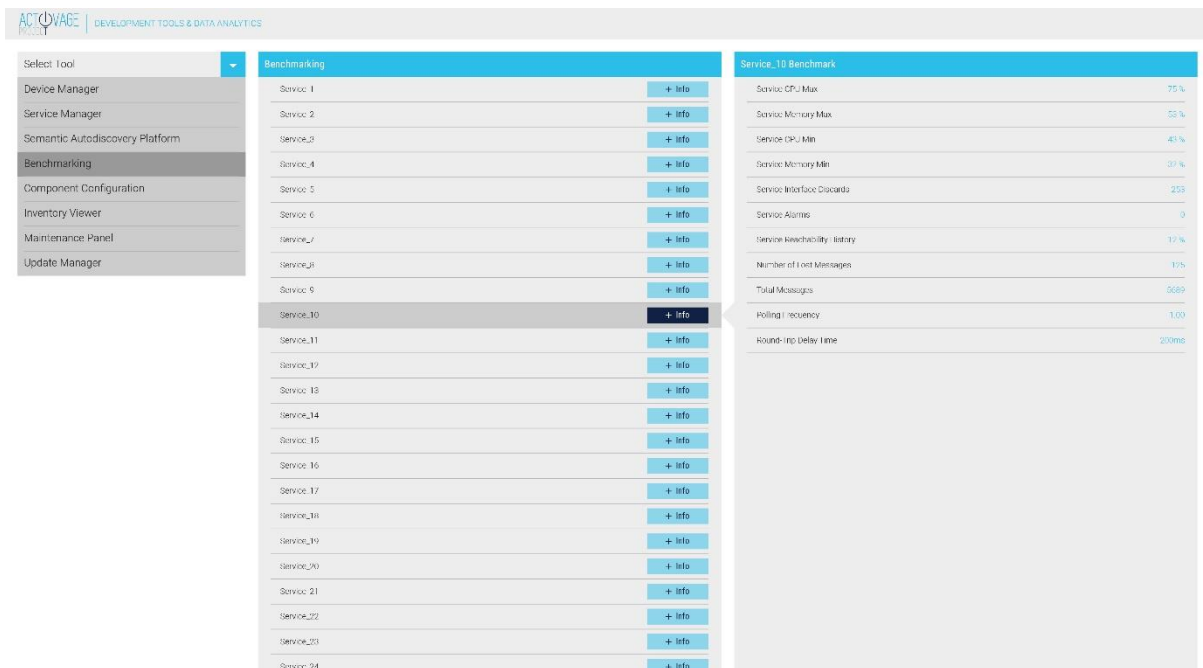


Figure 98: Benchmarking GUI mockup

### 5.3.2.9 Deployment Manager

#### 5.3.2.9.1 Functional description

Deployment Manager must assure the correct services and application deployment and its configuration once they are running. Assuming that the framework is developed and deployed using docker technology, this service must have access to the docker-engine, in order to be able to instantiate, stop or restart any of the containers running.



### 5.3.2.9.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless micro-service.

## 5.3.2.10 Inventory Viewer

### 5.3.2.10.1 Functional description

Inventory Viewer provides a set of well-defined methods to retrieve the current status of the different deployment sites providing services deployed, status of those services and the resource consumption.

### 5.3.2.10.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless micro-service.

### 5.3.2.10.3 Supported functionalities

Inventory viewer must provide a set of well-defined REST-based methods in order to:

- Retrieve deployment site status,
- Retrieve service status,
- Retrieve operation historical by service.

### 5.3.2.10.4 Sequence diagrams

For a better understanding of the flow process, the operations provided by the module are depicted in Figure 99.

### 5.3.2.10.5 User interface

A first graphical user interface is proposed in Figure 100.

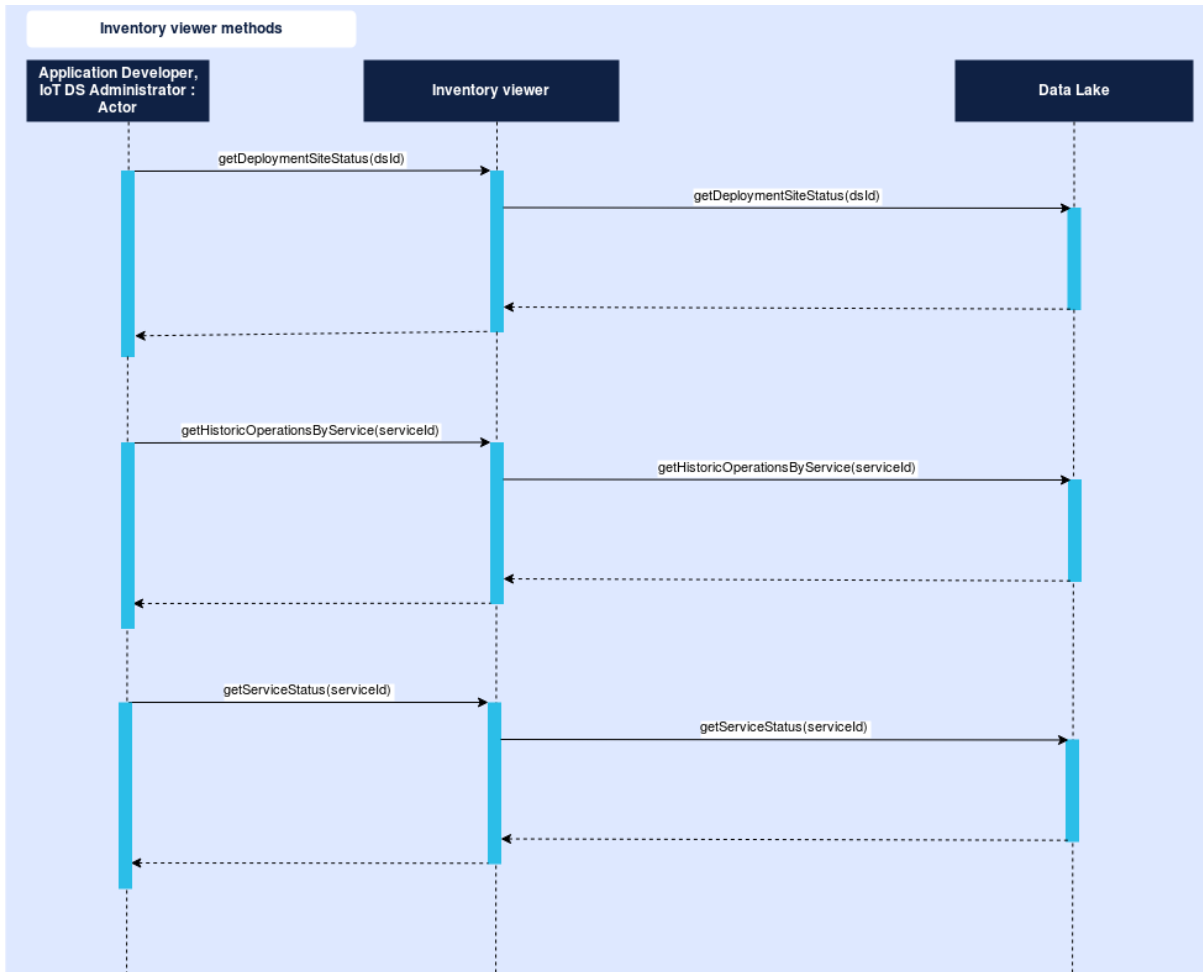


Figure 99: Inventory viewer sequence diagram

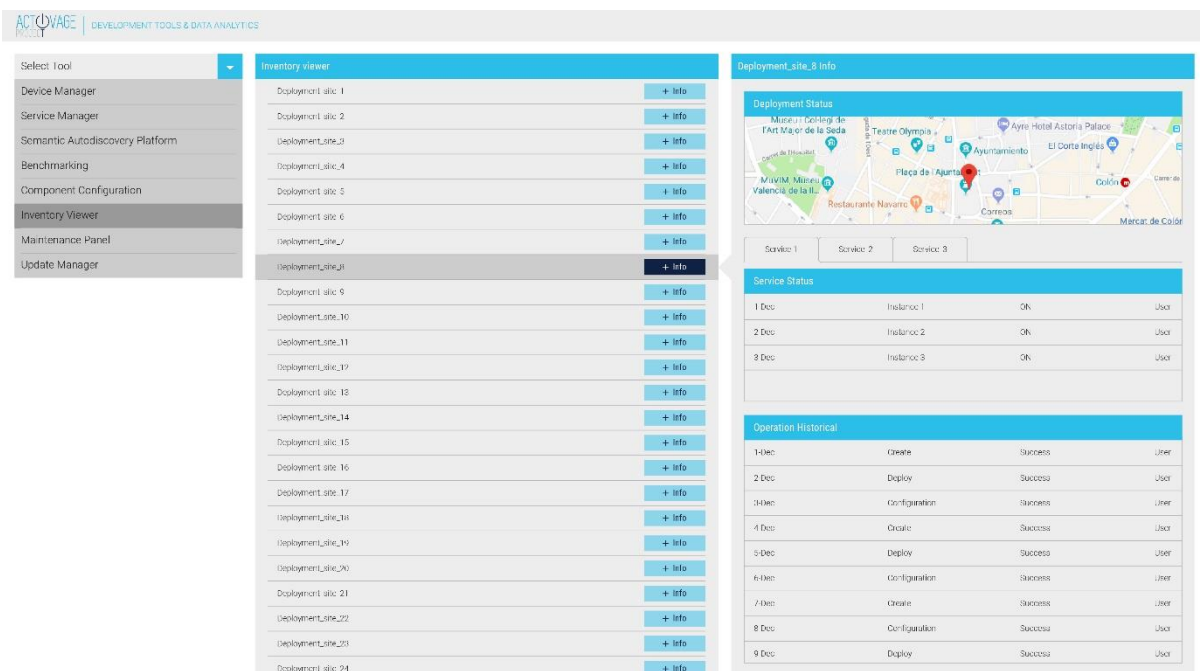


Figure 100: Inventory viewer GUI mockup

## 5.3.2.11 Component Configuration

### 5.3.2.11.1 Functional description

Component configuration must provide a set of methods to be able to configure the service and its interactions with other ACTIVAGE Applications, like Data Lake and Analytics.

### 5.3.2.11.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless micro-service.

### 5.3.2.11.3 REST-based methods

Component configuration module must provide the set of REST-based methods that allows a deployer to:

- Manage deployed services’ configurable parameters,
- Deploy a new service instance,
- Commission a deployed service in order to provide a custom configuration

### 5.3.2.11.4 Sequence diagrams

For a better understanding of the flow process, the operations provided by the module are depicted in Figure 101.

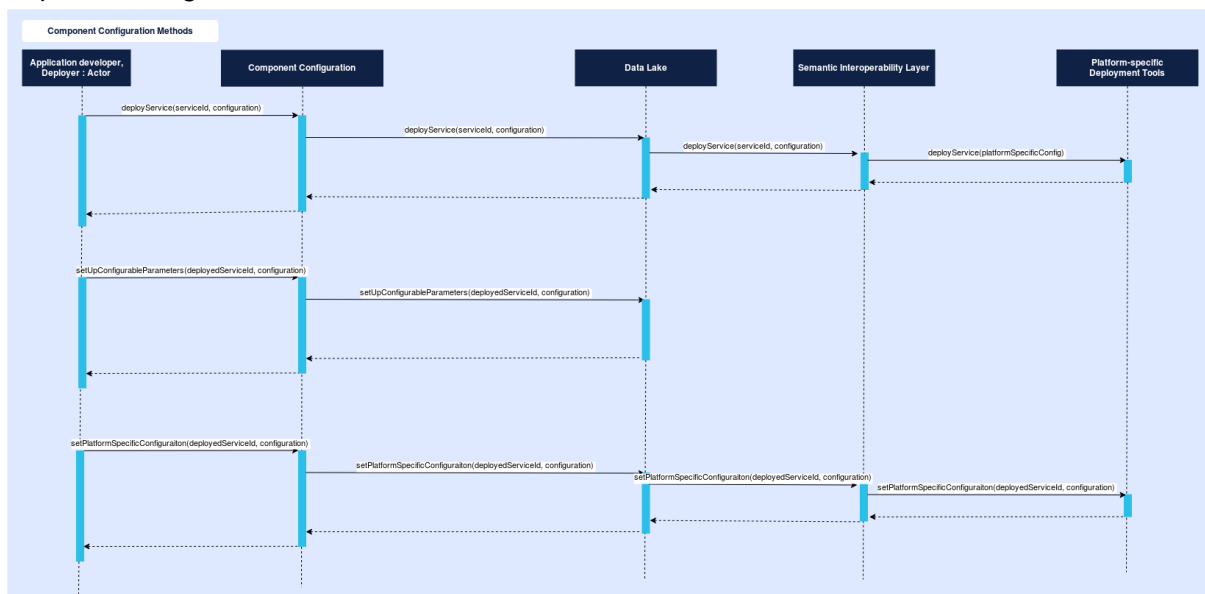


Figure 101: Component configuration sequence diagram

### 5.3.2.11.5 User interface

A first graphical user interface is proposed in Figure 102.

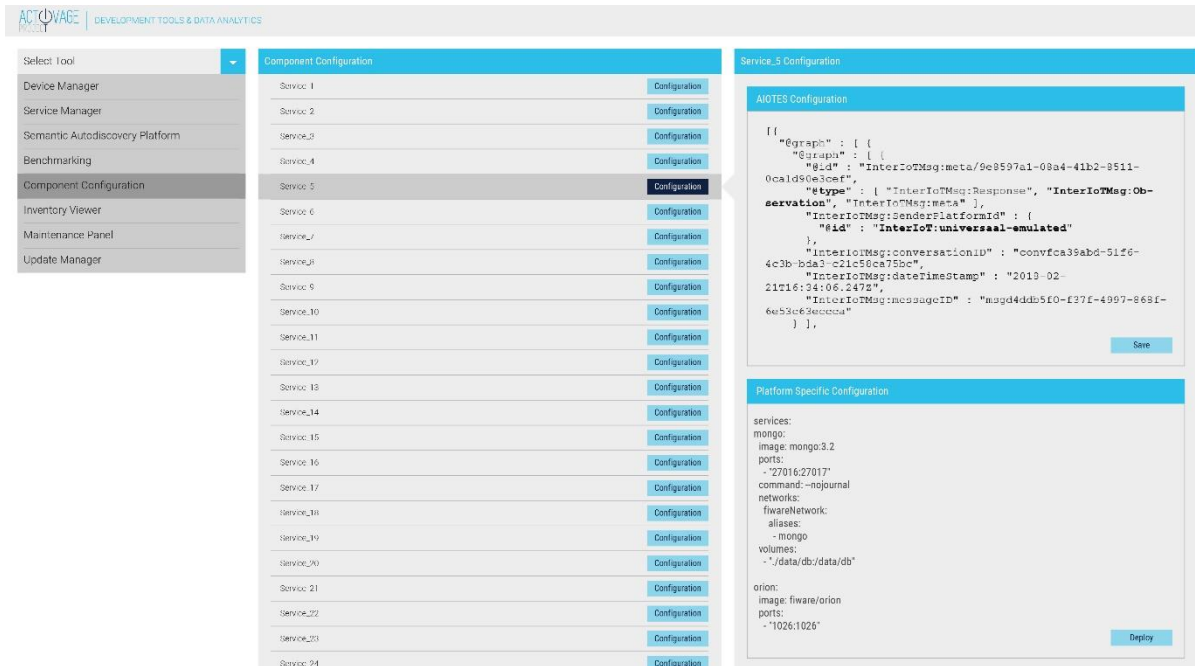


Figure 102: Component configuration GUI mockup

### 5.3.2.12 Maintenance panel

#### 5.3.2.12.1 Functional description

The maintenance panel provides a set of well-defined methods in order to facilitate the deployer the view and status of all components (devices and services) installed in a deployment unit or site.

#### 5.3.2.12.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless micro-service.

#### 5.3.2.12.3 REST-based methods

Maintenance panel must provide a set of well-defined REST-based methods in order to:

- Retrieve services’ operation status by deployment site,
- Subscribe to notification by service

#### 5.3.2.12.4 Sequence diagrams

For a better understanding of the flow process, the operations provided by the module are depicted in Figure 103.

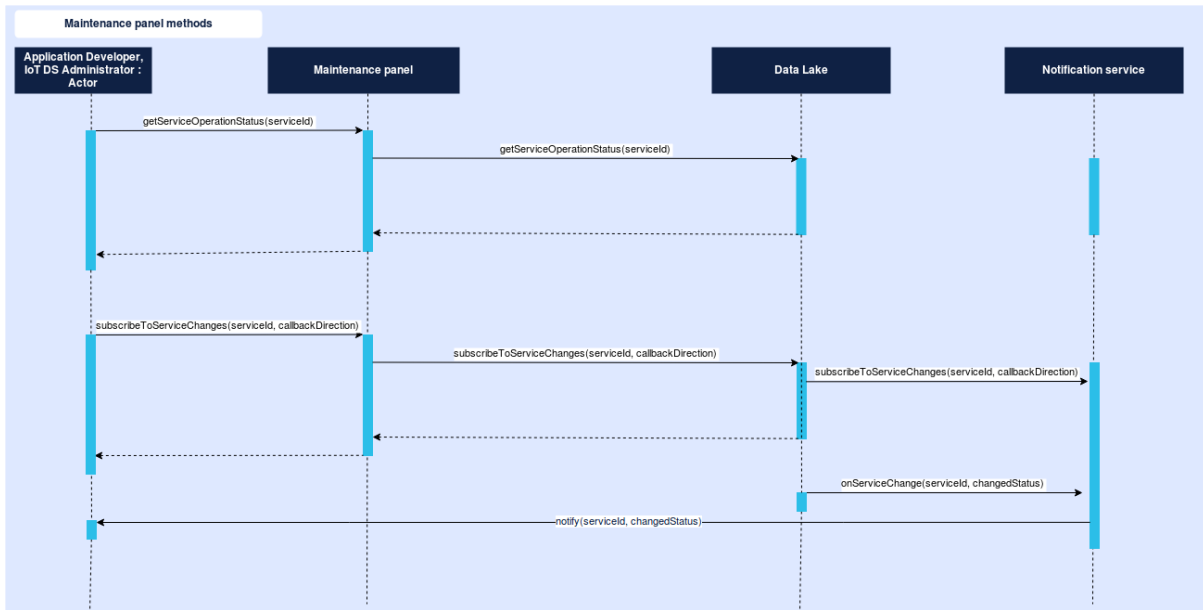


Figure 103: Maintenance panel sequence diagram

### 5.3.2.12.5 User interface

A first graphical user interface is proposed in Figure 104.

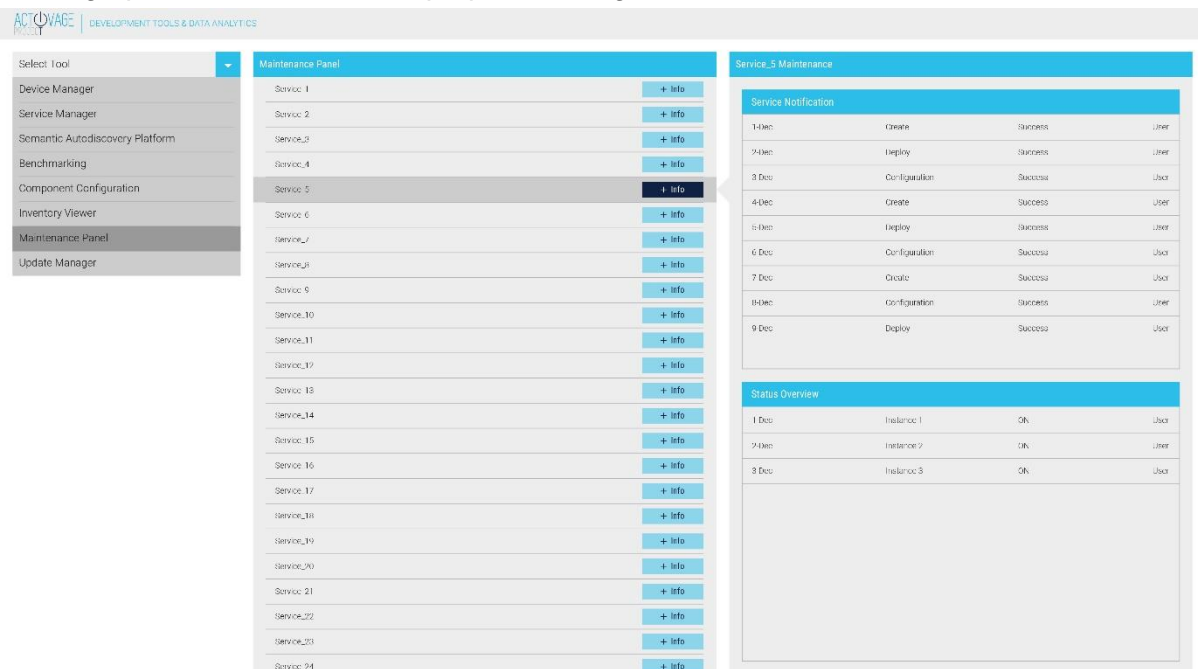


Figure 104: Maintenance panel GUI mockup

### 5.3.2.13 Update manager

#### 5.3.2.13.1 Functional description

The update manager deployment tool provides a set of well-defined methods which facilitates the deployer in updating the installed components and for this provides a notification service that notifies when new versions are available for each component.

### 5.3.2.13.2 Deploy and presentation method

The service must provide a REST-based API with the main functions aforementioned.

The service will be presented as a stateless micro-service.

### 5.3.2.13.3 REST-based methods

Maintenance panel must provide a set of well-defined REST-based methods in order to:

- Update a selected component,
- Notify interested user when a new component version is released.

### 5.3.2.13.4 Sequence diagrams

For a better understanding of the flow process, the operations provided by the module are depicted in Figure 105.

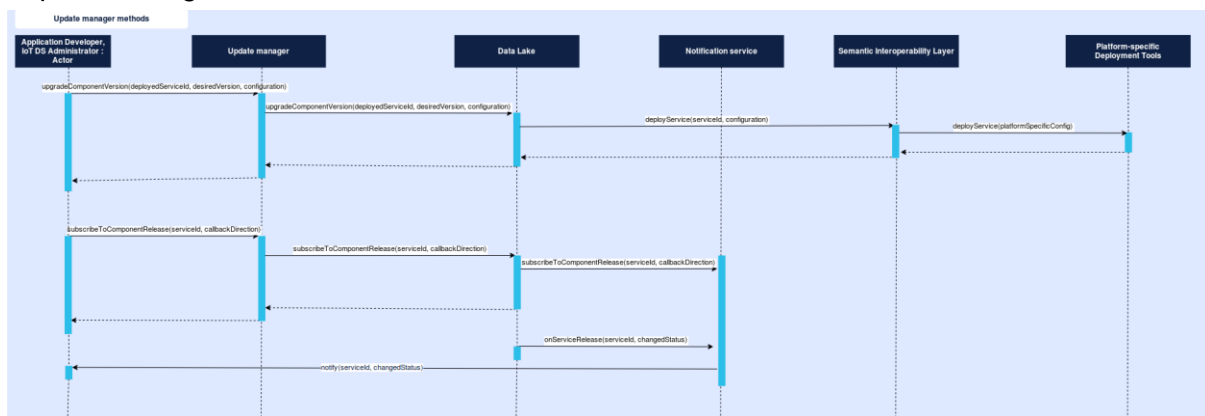


Figure 105: Update manager sequence diagram

### 5.3.2.13.5 User interface

A first graphical user interface is proposed in Figure 106.

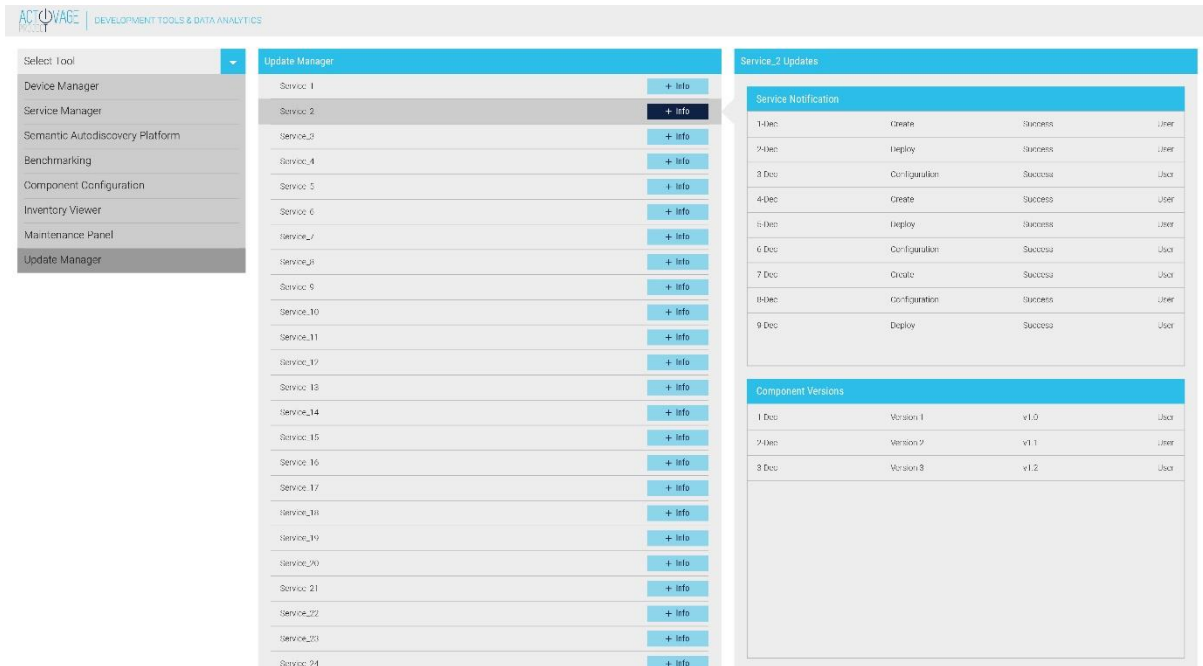


Figure 106: Update manager GUI mockup

### 5.3.3 Interactions between the Marketplace and deployment tools

As already described, deployment tools cover the processes of publishing, configuring, deploying and monitoring services in the AIOTES framework. Meanwhile, the ACTIVAGE Marketplace provides a user-friendly portal for creators/developers, deployers, technicians, stakeholders and end-users to carry out these exact processes of publishing, configuring, deploying, updating and, to some extent, monitoring AIOTES applications.

In this way, the Marketplace serves as front-end, and to be exact a higher-level user application that encapsulates deployment tools. The deployment tools, then, provide some, but not all, the fundamental functionality of the ACTIVAGE Marketplace. For the sake of completeness, first the overall Marketplace functionality is presented here, before examining dependencies on deployment tools.

Figure 107 shows a wireframe/mockup of the Marketplace Home Page, which shows an overview of featured and top rated apps, and the ability to search for apps using various filters and categories (tags). It also allows the current user to edit his profile, billing options, devices etc. An implementation of this wireframe is shown on Figure 108. Selecting an application leads to the Application View, a mockup of which is shown on Figure 109. This view presents screenshots of the application, information on how many installs it currently holds, user comments and replies and most importantly, the ability to buy and deploy it to a selection of the current user’s connected platforms and devices.

While the Marketplace is presented thoroughly in D4.3, this deliverable highlights the interactions between it and the deployment tools. Table 44 briefly lists all functionality, for completeness, with indications of where deployment tools are needed to realize these processes. In all cases, the Marketplaces delegates and takes advantage of the deployment tools REST-based API. These cases are mostly associated with discovering the user’s active connected devices and platforms to deploy to or manage and the service and device managers to actually deploy or update the applications. The rest of the functionality is implemented within the Marketplace itself, including not only the standard front- and back-

end of the web application to store and view the applications but also application search and discovery with the aim to support users and boost application sales.

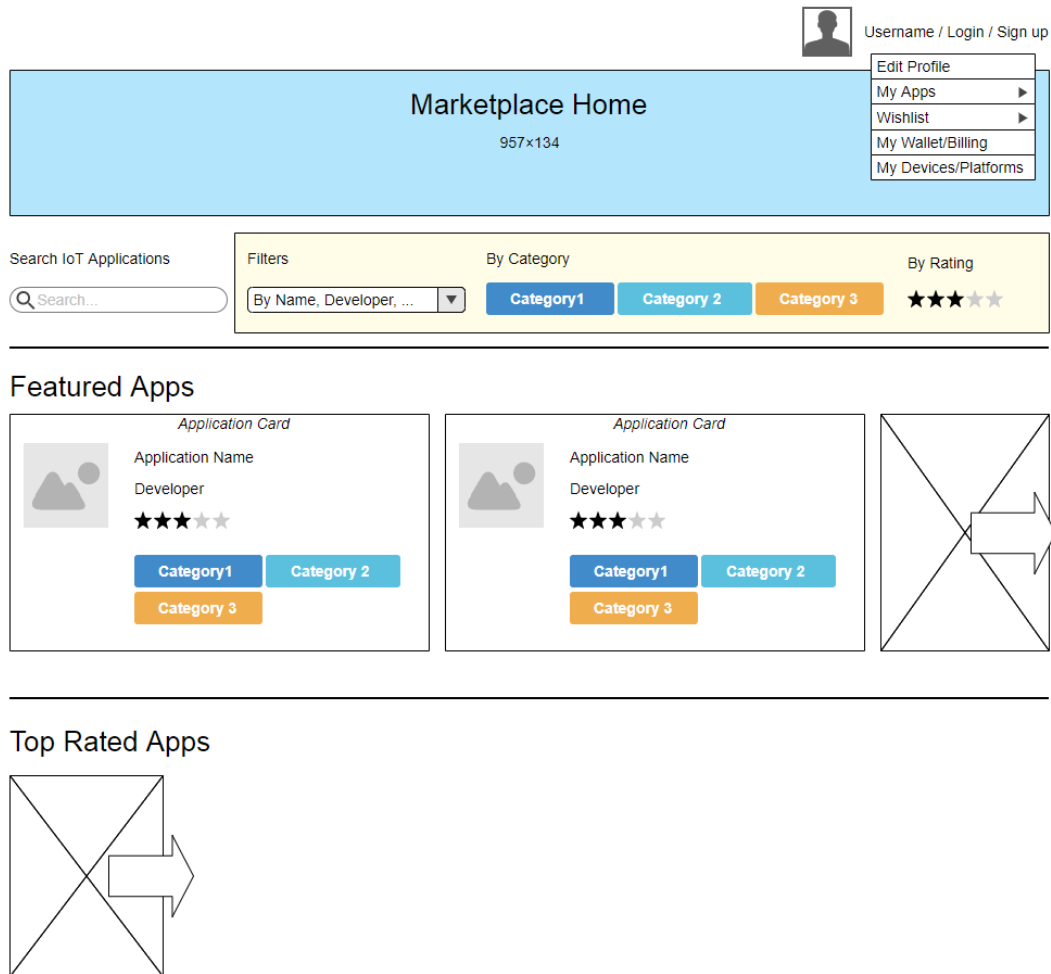


Figure 107: Marketplace Home Page mockup

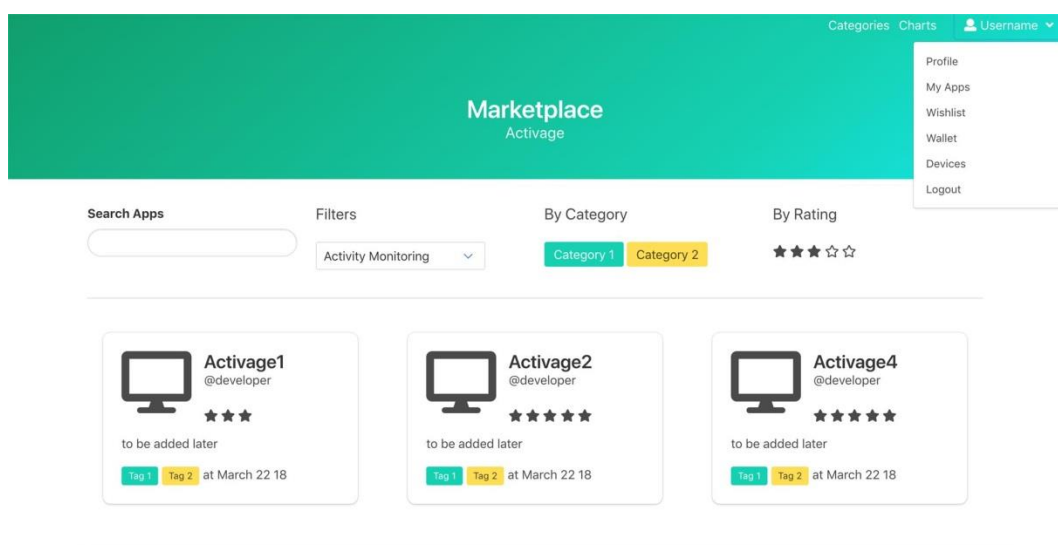


Figure 108: Marketplace Home Page implementation



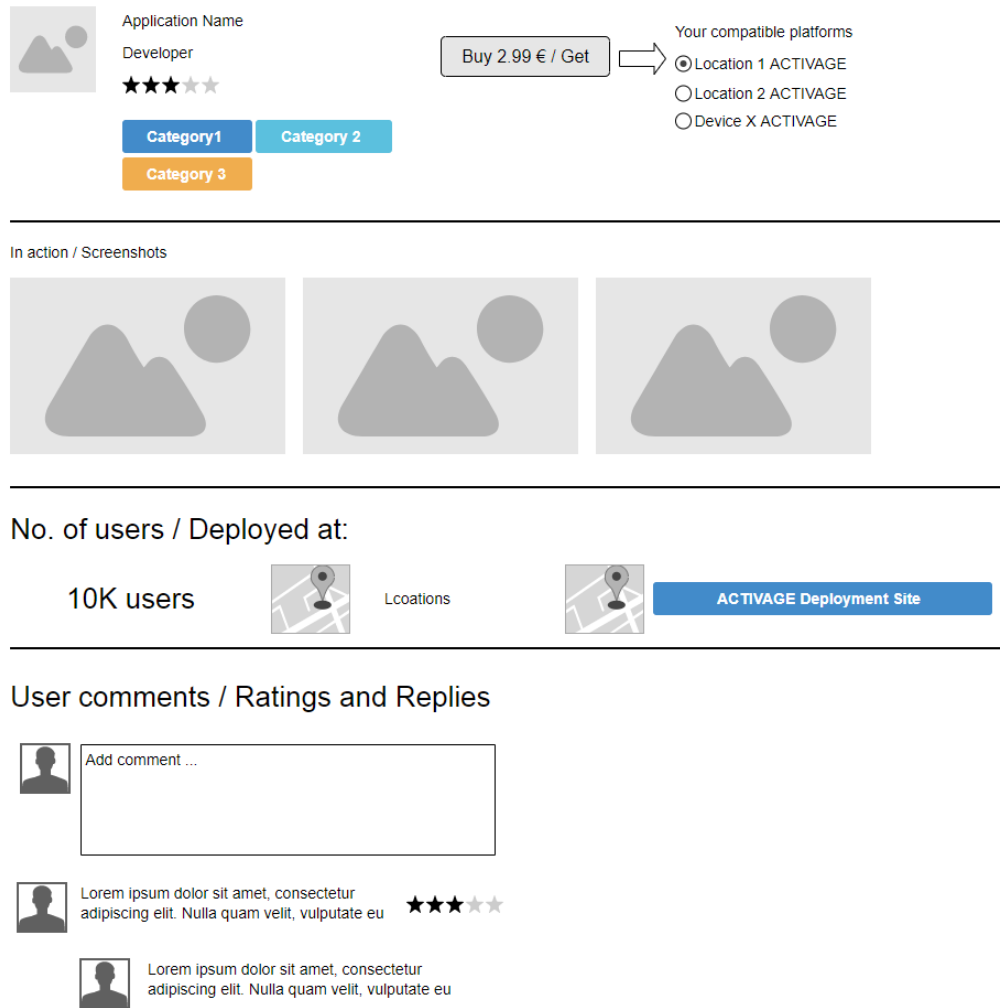


Figure 109: Marketplace Application View mockup.

Table 44: Marketplace functionality and dependencies on Deployment Tools

Functionality	User Role	Description	Deployment Tools
Publish Applications	Developer	Developers can publish their applications provided with a certain well-formed description, tags and features that will best allow for their discovery.	-
Sell Applications	Developer	Developers can receive and track payments, installs and uninstalls.	-
Receive and Reply to Ratings and Comments	Developer	Receive Ratings and Comments on developed applications and respond to them.	-
Search Applications	All	End-users can browse and search the repository of applications by maker, title, tags, category and description as well as discover links to applications similar to the ones the like.	-
Buy Applications	End-user	Pay for application using standard, secure payment methods	-

Deploy Applications	End-user	Download and deploy applications to compatible devices, either locally (i.e. same device the Marketplace is browsed from) or remotely (remote device is associated with the user and has internet access)	All (e.g. Service Manager/Catalog, Device Manager/Catalog, Semantic Discovery etc.)
Update Applications	End-user	Update applications with latest version automatically or after approval	All (e.g. Service Manager/Catalog, Device Manager/Catalog, Semantic Discovery etc.)
Rate and Comment on Applications	End-user	Rate owned applications, comment and get replies	-
Manage User Account	All	Manage user profile, application wishlist, transactions, installed applications and associated devices.	Device Manager/Catalog

## 6 Conclusion / Future Work

This is the first version of deliverable D4.1 that reports on the progress and work done in Task 4.1 and 4.3, aiming at providing a set of development and deployment tools that can facilitate the adoption of ACTIVAGE approach for AHA IoT solutions, and that are part of AIOTES.

One of the main barriers frequently reported for the extended use of IoT platforms in AHA domain, according to the developers' community of the input platforms, is the lack of easy to use development and deployment tools, especially for less technical specialist stakeholders, so the tools defined in this report constitute important assets of ACTIVAGE project to achieve acceptance from external stakeholders and succeed in the goal of enlarging the ecosystem.

The proposed tools cover the whole lifecycle of AHA solutions based on IoT, from design to operation, and the initial steps have advanced the needed identification of use cases and requirements related to development and deployment tools.

After the analysis of the requirements extracted from different sources (i.e. background IoT platforms, deliverable D2.1 on requirements, technical experts from deployment sites...), 23 requirements related to development tools have been defined, organising them in four main categories: support consumption, implementation, data processing and IoT infrastructure management.

Similarly, a total of 14 requirements related to deployment tools has been organised in two main categories, such as IoT infrastructure management, both devices and services, and distribution and deployment.

In the analysis of these requirements it has been taken into account the different level of skills and competences of developers, so the definition of use cases reflects the complexity of the potential users of the tools. In particular, it is proposed to incorporate in the set of tools a new one specially developed for non-technical developers, ClickDigital, a pluggable visual IoT IDE for different IoT platforms, and that will be connected to IoT platforms deployed in ACTIVAGE through AIOTES interoperability layer.

As result of this work, the conceptual architecture of the ACTIVAGE development tools component as well as deployment tools component and its connection to the other ACTIVAGE components have been defined, specially with regard to the AIOTES SIL, data lake and data analytics layers. A total of 21 development tools and 8 deployment tools have been specified, describing their goal, main functionalities and intended usage. The proposed tools address all identified requirements.

One of the cornerstone of ACTIVAGE project is the reusability, from all possible aspects, so the next step once we have defined the needed tools has been to analyse which tools are already provided by the IoT platforms integrated in ACTIVAGE and could form the basis for the set of development tools. The result of this analysis can be viewed in section 5.2.8 and provides an accurate landscape of available tools.

A similar analysis has been done for the deployment tools, this time also including other well-known platform independent tools, such Docker or OSGi based tools. The result of this analysis can be viewed in section 6.2.9 and in this case, it evidences the lack of appropriate tools for deployment that could be potentially reused. As a consequence, a more detailed functional description, including use cases diagrams and mock-up GUI are provided in section 6.3

The next steps regarding the development and deployment tools is to start the implementation work of the different tools according to the agreed plan, summarized in

Section 4.3.1 for the development tools. An initial set of tools will be available at the time of launching ACTIVAGE first open call, as described in D7.1, [36], as they are needed for the external stakeholders participating to develop and deploy their solutions on top of AIOTES.

# References

- [1] D3.1 REPORT ON IoT EUROPEAN PLATFORMS: [ACTIVAGE > 3 ACTIVAGE PROJECT > 014 Project Deliverables > Deliverables Month 3 > Deliverable 3.1 - Report on IoT European Platforms](#)
- [2] [HTTPS://FORGE.FIWARE.ORG/PLUGINS/MEDIAWIKI/WIKI/FIWARE/INDEX.PHP/WELCOME\\_TO\\_THE\\_FIWARE\\_WIKI](https://forge.fiware.org/plugins/mediawiki/wiki/Fiware/Index.php/welcome_to_the_fiware_wiki)
- [3] [WWW.FIWARE.ORG](http://www.fiware.org)
- [4] [HTTPS://WWW.FIWARE.ORG/DEVELOPERS-ENTREPRENEURS/](https://www.fiware.org/developers-entrepreneurs/)
- [5] [HTTPS://CATALOGUE.FIWARE.ORG/](https://catalogue.fiware.org/)
- [6] [HTTP://WWW.OPENIOT.EU](http://www.openiot.eu)
- [7] [HTTPS://GITHUB.COM/OPENIOTORG/OPENIOT](https://github.com/OpenIoTOrg/OpenIoT)
- [8] [HTTPS://GITHUB.COM/GOLLUM/GOLLUM#README](https://github.com/Gollum/Gollum#readme)
- [9] [HTTP://SOFIA2.COM](http://sofia2.com)
- [10] [HTTP://SOFIA2.COM/DESARROLLADOR\\_EN.HTML#DOCUMENTACIO](http://sofia2.com/desarrollador_en.html#documentacio)
- [11] SWAGGER: <https://swagger.io/>
- [12] ORACLE APIARY: <https://apiary.io/>
- [13] GIT: [HTTPS://GIT-SCM.COM/](https://git-scm.com/)
- [14] GIT DOCUMENTATION: [HTTPS://GIT-SCM.COM/DOC](https://git-scm.com/doc)
- [15] UNIVERSAAL IoT, SEMANTIC INTEROPERABILITY FOR RAPID INTEGRATION & DEPLOYMENT: [HTTPS://GITHUB.COM/UNIVERSAAL](https://github.com/universaal)
- [16] SOFIA2 WEB SITE: [HTTP://SOFIA2.COM/HOME\\_EN.HTML](http://sofia2.com/home_en.html)
- [17] OPENIOT – OPEN SOURCE CLOUD SOLUTION FOR THE INTERNET OF THINGS: [HTTP://WWW.OPENIOT.EU](http://www.openiot.eu)
- [18] [HTTPS://GITHUB.COM/OPENIOTORG/OPENIOT/WIKI](https://github.com/OpenIoTOrg/OpenIoT/wiki)
- [19] [HTTPS://GITHUB.COM/OPENIOTORG/OPENIOT/WIKI/DOWNLOADS](https://github.com/OpenIoTOrg/OpenIoT/wiki/downloads)
- [20] [HTTPS://GITHUB.COM/OPENIOTORG/OPENIOT/WIKI/USER-INSTRUCTIONS](https://github.com/OpenIoTOrg/OpenIoT/wiki/user-instructions)
- [21] SENSINACT WEB SITE: [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.SENSINACT](https://projects.eclipse.org/projects/technology.sensinact)
- [22] [HTTP://EDU.FIWARE.ORG/](http://edu.fiware.org/)
- [23] [HTTP://SOFIA2.COM/DOCS/\(EN\)%20SOFIA2-SOFIA2%20CONCEPTS.PDF](http://sofia2.com/docs/(en)%20sofia2-sofia2%20concepts.pdf)
- [24] [HTTP://SOFIA2.COM/DOCS/\(EN\)%20SOFIA2-FIRST%20STEPS%20WITH%20SOFIA2.PDF](http://sofia2.com/docs/(en)%20sofia2-first%20steps%20with%20sofia2.pdf)
- [25] [HTTP://SOFIA2.COM/DOCS/\(EN\)%20SOFIA2-HOW%20TO%20DEVELOP%20ON%20THE%20SOFIA2%20PLATFORM.PDF](http://sofia2.com/docs/(en)%20sofia2-how%20to%20develop%20on%20the%20sofia2%20platform.pdf)
- [26] [HTTP://SOFIA2.COM/DOCS/SOFIA2-APIs%20SOFIA2.PDF](http://sofia2.com/docs/sofia2-apis%20sofia2.pdf)
- [27] [HTTP://SOFIA2.COM/SIB/](http://sofia2.com/sib/)
- [28] [HTTPS://WWW.OPENLDAP.ORG/](https://www.openldap.org/)
- [29] [HTTPS://HUB.DOCKER.COM/U/IOTIVITY/](https://hub.docker.com/u/iotivity/)
- [30] FIWARE WEB SITE: [HTTPS://WWW.FIWARE.ORG/](https://www.fiware.org/)
- [31] IOTIVITY WEB SITE: [HTTPS://WWW.IOTIVITY.ORG/](https://www.iotivity.org/)
- [32] SENIORSOME WEB SITE: [HTTP://WWW.SENIORSOME.COM](http://www.seniorsome.com)
- [33] ONEM2M WEBSITE: [HTTP://WWW.ONEM2M.ORG](http://www.onem2m.org)
- [34] ECLIPSE TECHNOLOGY: [HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY](https://projects.eclipse.org/projects/technology)
- [35] ECLIPSE PUBLIC LICENSE V1.0 : <https://www.eclipse.org/org/documents/epl-v10.php>
- [36] D7.1 INITIAL ECOSYSTEM MANAGEMENT PLAN: [ACTIVAGE > 3 ACTIVAGE PROJECT>014 PROJECT DELIVERABLES > 05 - DELIVERABLES MONTH 15](#) [D.7.1 INITIAL ECOSYSTEM MANAGEMENT PLAN](#)