Developers toolkit and deployment support

<table>
<thead>
<tr>
<th>Deliverable No.</th>
<th>Type</th>
<th>Version</th>
<th>Dissemination Level</th>
<th>Status</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4.4 (D4.1.2)</td>
<td>Report</td>
<td>1.0</td>
<td>Public</td>
<td>Release 1</td>
<td>30-Jun-2019</td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provides a set of methods and tools for (non-) technical developers to create applications and services within AIOTES.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work Package</td>
<td>WP4 – ACTIVAGE Application support tools and services layer.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Authors

<table>
<thead>
<tr>
<th>Name</th>
<th>Partner</th>
<th>e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stéphane Bergeon</td>
<td>CEA</td>
<td><a href="mailto:stephane.bergeon@cea.fr">stephane.bergeon@cea.fr</a></td>
</tr>
<tr>
<td>Levent Gürgeen</td>
<td>CEA</td>
<td><a href="mailto:levent.gurgen@cea.fr">levent.gurgen@cea.fr</a></td>
</tr>
<tr>
<td>Mathieu Gallissot</td>
<td>CEA</td>
<td><a href="mailto:mathieu.gallissot@cea.fr">mathieu.gallissot@cea.fr</a></td>
</tr>
<tr>
<td>Clara Valero</td>
<td>UPV</td>
<td><a href="mailto:clavalpe@dcom.upv.es">clavalpe@dcom.upv.es</a></td>
</tr>
<tr>
<td>Matilde Juan Segui</td>
<td>UPV</td>
<td><a href="mailto:majuse@upv.es">majuse@upv.es</a></td>
</tr>
<tr>
<td>Nikolaos Kaklanis</td>
<td>CERTH</td>
<td><a href="mailto:nkak@iti.gr">nkak@iti.gr</a></td>
</tr>
<tr>
<td>Konstantinos Votis</td>
<td>CERTH</td>
<td><a href="mailto:kvotis@iti.gr">kvotis@iti.gr</a></td>
</tr>
<tr>
<td>Dimitrios Tzovaras</td>
<td>CERTH</td>
<td><a href="mailto:Dimitrios.Tzovaras@iti.gr">Dimitrios.Tzovaras@iti.gr</a></td>
</tr>
<tr>
<td>Thanos Stavropoulos</td>
<td>CERTH</td>
<td><a href="mailto:athstavr@iti.gr">athstavr@iti.gr</a></td>
</tr>
<tr>
<td>Spiros Nikolopoulos</td>
<td>CERTH</td>
<td><a href="mailto:nikolopo@iti.gr">nikolopo@iti.gr</a></td>
</tr>
<tr>
<td>Ioannis Kompatsiaris</td>
<td>CERTH</td>
<td><a href="mailto:ikom@iti.gr">ikom@iti.gr</a></td>
</tr>
<tr>
<td>Stefanos Stavrotheodoros</td>
<td>CERTH</td>
<td><a href="mailto:stavrotheodoros@iti.gr">stavrotheodoros@iti.gr</a></td>
</tr>
<tr>
<td>Felipe Roca Blaya</td>
<td>HOPU</td>
<td><a href="mailto:felipe@hopu.eu">felipe@hopu.eu</a></td>
</tr>
<tr>
<td>Óscar García Martínez</td>
<td>HOPU</td>
<td><a href="mailto:oscar@hopu.eu">oscar@hopu.eu</a></td>
</tr>
<tr>
<td>Achille Zappa</td>
<td>NUIG</td>
<td><a href="mailto:achille.zappa@insight-centre.org">achille.zappa@insight-centre.org</a></td>
</tr>
<tr>
<td>Martin Serrano</td>
<td>NUIG</td>
<td><a href="mailto:martin.serrano@insight-centre.org">martin.serrano@insight-centre.org</a></td>
</tr>
<tr>
<td>Byron Ortiz Sanchez</td>
<td>TVES</td>
<td><a href="mailto:byrort@televes.com">byrort@televes.com</a></td>
</tr>
<tr>
<td>Pilar Sala</td>
<td>MYSPHERA</td>
<td><a href="mailto:psala@mysphera.com">psala@mysphera.com</a></td>
</tr>
<tr>
<td>Alejandro M. Medrano Gil</td>
<td>UPM</td>
<td><a href="mailto:amedrano@lst.tfo.upm.es">amedrano@lst.tfo.upm.es</a></td>
</tr>
<tr>
<td>Saied Tazari</td>
<td>Fh-IGD</td>
<td><a href="mailto:saied.tazari@igd.fraunhofer.de">saied.tazari@igd.fraunhofer.de</a></td>
</tr>
<tr>
<td>Helmi Ben Hmidia</td>
<td>Fh-IGD</td>
<td><a href="mailto:helmi.ben.hmida@igd.fraunhofer.de">helmi.ben.hmida@igd.fraunhofer.de</a></td>
</tr>
</tbody>
</table>
History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-April-2019</td>
<td>0.1</td>
<td>Initial ToC</td>
</tr>
<tr>
<td>24-April-2019</td>
<td>0.2</td>
<td>Integrated UPV contributions - 4.1.4</td>
</tr>
<tr>
<td>08 May 2019</td>
<td>0.3</td>
<td>Integrated NUIG contribution - 4.1.2</td>
</tr>
<tr>
<td>10 May 2019</td>
<td>0.4</td>
<td>Integrated HOPU contribution - 5.1.2</td>
</tr>
<tr>
<td>13 May 2019</td>
<td>0.5</td>
<td>Integrated UPM contribution - 4.1.4</td>
</tr>
<tr>
<td>21 May 2019</td>
<td>0.6</td>
<td>Integrated CERTH contribution - 4.1.1, 4.1.3, 5.1.1</td>
</tr>
<tr>
<td>23 May 2019</td>
<td>0.7</td>
<td>Integrated CEA contributions – 5.1.1</td>
</tr>
<tr>
<td>07 Juin 2019</td>
<td>0.8</td>
<td>Integrated FhG contributions – 4.1.3, 4.1.4</td>
</tr>
<tr>
<td>28 Juin 2019</td>
<td>0.9</td>
<td>First integrated version for the internal review</td>
</tr>
<tr>
<td>19 July 2019</td>
<td>0.95</td>
<td>Final version integrating review comments</td>
</tr>
<tr>
<td>05 Aug 2019</td>
<td>1.0</td>
<td>Released</td>
</tr>
</tbody>
</table>

Key data

<table>
<thead>
<tr>
<th>Keywords</th>
<th>ACTIVAGE, AHA, development toolkit, deployment tool, API, wiki, documentation, source code, swagger, tutorial, IoT platform, FIWARE, IoTivity, OpenIOT, SOFIA2, SENIORSOME, sensiNact, universAAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead Editor</td>
<td>Stéphane Bergeon 07 CEA</td>
</tr>
<tr>
<td></td>
<td>Levent Gürgen 07 CEA</td>
</tr>
<tr>
<td>Internal Reviewer(s)</td>
<td>Alexandre Duclos 16 MAD</td>
</tr>
<tr>
<td></td>
<td>Elena Tamburini 14 MEDEA</td>
</tr>
<tr>
<td></td>
<td>Philippe Dallemagne 19 CSEM</td>
</tr>
</tbody>
</table>
Abstract

This document is the second version of 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”. While the first version of the document made a status of existing development and deployment tools prior to the ACTIVAGE project and described the ones that are under progress to fulfill ACTIVAGE dedicated requirements, this final version presents the tools that have been developed and adapted for the ACTIVAGE project.

The document is structured as follows: the requirements for deployment and deployment tools are presented in Section 2, followed by some use case scenarios by different types of developers in Section 3. Section 4 and Section 5 present respectively the tools for development and deployment of applications on top of the AIOTES. Indeed, the main updated sections in this deliverable with respect to the D4.1 are the Section 4 and Section 5, presenting in detail the developed ACTIVAGE tools for development and deployment. Those tools are either developed from scratch or adapted from platform specific tools (which have been presented as Appendix in Section 7). For more details about the seven ACTIVAGE IoT platforms, please refer to the D3.1 Report on IoT European Platforms [1].

These existing development and deployment tools are available today and constitute the ACTIVAGE toolkit 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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE OF CONTENTS</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>7</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>9</td>
</tr>
<tr>
<td>1 ABOUT THIS DOCUMENT</td>
<td>13</td>
</tr>
<tr>
<td>1.1 DELIVERABLE CONTEXT</td>
<td>14</td>
</tr>
<tr>
<td>1.2 THE RATIONALE BEHIND THE STRUCTURE</td>
<td>14</td>
</tr>
<tr>
<td>2 REQUIREMENTS FOR DEVELOPMENT AND DEPLOYMENT</td>
<td>15</td>
</tr>
<tr>
<td>3 USE CASES</td>
<td>17</td>
</tr>
<tr>
<td>3.1 DEVELOPER PROFILES</td>
<td>17</td>
</tr>
<tr>
<td>3.2 INDICATIVE USE CASES FOR DEVELOPMENT</td>
<td>18</td>
</tr>
<tr>
<td>3.2.1 Indicative use cases for non technical developers</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2 Support</td>
<td>21</td>
</tr>
<tr>
<td>3.3 INDICATIVE USE CASES FOR DEPLOYMENT</td>
<td>23</td>
</tr>
<tr>
<td>4 DEVELOPMENT TOOLS</td>
<td>25</td>
</tr>
<tr>
<td>4.1 ARCHITECTURE</td>
<td>25</td>
</tr>
<tr>
<td>4.1.1 Semantic Interoperability Layer tools</td>
<td>28</td>
</tr>
<tr>
<td>4.1.2 Data Lake tools</td>
<td>38</td>
</tr>
<tr>
<td>4.1.3 Data / visual analytics tools</td>
<td>43</td>
</tr>
<tr>
<td>4.1.4 Integrated Development Environment (IDE)</td>
<td>54</td>
</tr>
<tr>
<td>4.1.5 Mapping between development tools requirements and modules</td>
<td>66</td>
</tr>
<tr>
<td>5 DEPLOYMENT TOOLS</td>
<td>68</td>
</tr>
<tr>
<td>5.1 ARCHITECTURE</td>
<td>68</td>
</tr>
<tr>
<td>5.1.1 IoT infrastructure management tools</td>
<td>71</td>
</tr>
<tr>
<td>5.1.2 Deployment management tools</td>
<td>82</td>
</tr>
<tr>
<td>5.1.3 Mapping between deployment tools requirements and modules</td>
<td>93</td>
</tr>
<tr>
<td>6 CONCLUSION / FUTURE WORK</td>
<td>95</td>
</tr>
<tr>
<td>7 REFERENCES</td>
<td>96</td>
</tr>
<tr>
<td>APPENDIX A AVAILABLE DEVELOPMENT TOOLS SUPPORTED BY THE ACTIVAGE IOT PLATFORMS</td>
<td>97</td>
</tr>
<tr>
<td>A.1 UNIVERSAAL</td>
<td>97</td>
</tr>
<tr>
<td>A.1.1 Support Tools</td>
<td>97</td>
</tr>
<tr>
<td>A.1.2 Useful tools in the context of ACTIVAGE</td>
<td>101</td>
</tr>
<tr>
<td>A.1.3 Mapping between universAAL and ACTIVAGE development tools</td>
<td>112</td>
</tr>
<tr>
<td>A.2 SOFIA2</td>
<td>114</td>
</tr>
<tr>
<td>A.2.1 Documentation and WIKI</td>
<td>114</td>
</tr>
<tr>
<td>A.2.2 APIs and libraries</td>
<td>114</td>
</tr>
</tbody>
</table>
A.2.3 Source code public access ................................................................. 115
A.2.4 SOFIA2 tutorial and sample source code ........................................... 115
A.2.5 Useful tools in ACTIVAGE context ...................................................... 116
A.2.6 Mapping between SOFIA2 and ACTIVAGE development tools ............ 118
A.3 OPENIoT ......................................................................................... 119
A.3.1 Service Delivery & Utility ................................................................. 119
A.3.2 Linked Stream Middleware Light ....................................................... 120
A.3.3 User Interfaces ............................................................................... 122
A.3.4 Utilities & Libraries ......................................................................... 125
A.3.5 Source Code & Binaries ................................................................... 126
A.3.6 Documentation ................................................................................ 127
A.3.7 Mapping between OpenIoT and ACTIVAGE Development Tools ........ 127
A.4 SENSI NACT .................................................................................. 128
A.4.1 sensiNact documentation and WIKI .................................................. 129
A.4.2 sensiNact API doc and swagger ......................................................... 129
A.4.3 sensiNact source code public access ................................................ 131
A.4.4 sensiNact tutorial and sample source code ....................................... 132
A.4.5 sensiNact ACTIVAGE development tools ......................................... 132
A.4.6 Summary of sensiNact development tools provided for ACTIVAGE .... 146
A.4.7 Mapping between sensiNact and ACTIVAGE Development Tools ...... 147
A.5 FIWARE ......................................................................................... 148
A.5.1 Supports Tools ............................................................................... 148
A.5.2 Tools supported by FIWARE .............................................................. 151
A.5.3 Mapping between FIWARE and ACTIVAGE development tools ......... 154
A.6 IOTIVITY ......................................................................................... 154
A.6.1 Support tools .................................................................................. 155
A.6.2 Tools useful in the ACTIVAGE context ............................................. 157
A.6.3 Mapping between IoTivity and ACTIVAGE development tools .......... 161
A.7 SENIOR SOME .............................................................................. 162
A.7.1 Support ......................................................................................... 162
A.7.2 Integrated Development Environment ............................................. 162
A.7.3 Data- and visual analytics tools ........................................................ 163
A.7.4 Data Lake tools .............................................................................. 163
A.7.5 Semantic Interoperability Layer ....................................................... 163
A.7.6 Mapping between SeniorSome and ACTIVAGE Development Tools .... 163
A.8 SUMMARY OF EXISTING TOOLS ..................................................... 164
A.8.1 Platform-specific development tools comparison ................................ 164
A.8.2 Mapping between platform-specific development tools and proposed ACTIVAGE development tools .................................................. 167
APPENDIX B AVAILABLE DEPLOYMENT TOOLS .............................................
SUPPORTED BY THE ACTIVAGE IOT PLATFORMS .................................. 169
B.1  PLATFORM INDEPENDENT AVAILABLE DEPLOYMENT TOOLS .............................................................. 169
  B.1.1  Docker .......................................................................................................................... 169
  B.1.2  Virtual Machine ............................................................................................................. 169
  B.1.3  OSGi based deployment ................................................................................................. 170
B.2  UNIVERSAAL ...................................................................................................................................... 171
  B.2.1  Distributions ...................................................................................................................... 171
  B.2.2  Runtime tools ................................................................................................................... 172
  B.2.3  Deploy Management & uCC ............................................................................................ 173
  B.2.4  Mapping between universAAL and ACTIVAGE deployment tools ..................................... 174
B.3  SOFIA2 ........................................................................................................................................... 175
  B.3.1  Mapping between deployment tools requirements and modules ........................................... 177
B.4  OPENIOT ...................................................................................................................................... 177
B.5  SENsinACT .................................................................................................................................... 178
  B.5.1  Deployment tools around sensiNACT ................................................................................. 178
  B.5.2  Mapping between deployment tools requirements and modules ........................................ 182
B.6  FIWARE ......................................................................................................................................... 182
  B.6.1  Mapping between deployment tools requirements and modules ........................................ 183
B.7  IOtIVITY ........................................................................................................................................ 183
  B.7.1  Easy Setup .......................................................................................................................... 183
  B.7.2  Device Management ............................................................................................................ 184
  B.7.3  Mapping between deployment tools requirements and modules .......................................... 191
B.8  SENiorsche .................................................................................................................................... 191
B.9  SUMMARY OF EXISTING TOOLS ............................................................................................ 192
  B.9.1  Platform-specific deployment tools comparison ............................................................... 192
  B.9.2  Mapping between platform-specific deployment tools and proposed ACTIVAGE deployment tools .................................................................................................................. 193

APPENDIX C  DETAILED CLICKDIGITAL ARCHITECTURE AND IMPLEMENTATION ................. 195
List of tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>The functionalities offered by the ACTIVAGE deployment tools.</td>
<td>69</td>
</tr>
<tr>
<td>Table 2</td>
<td>universAAL support tools</td>
<td>97</td>
</tr>
<tr>
<td>Table 3</td>
<td>universAAL API</td>
<td>99</td>
</tr>
<tr>
<td>Table 4</td>
<td>universAAL Tools Mappings with ACTIVAGE development tools</td>
<td>112</td>
</tr>
<tr>
<td>Table 5</td>
<td>SOFIA2 existing samples and source code</td>
<td>116</td>
</tr>
<tr>
<td>Table 6</td>
<td>MAPPING BETWEEN SOFIA2 AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>118</td>
</tr>
<tr>
<td>Table 7</td>
<td>LIST OF PRIMITIVES COMPRISING THE OPENiOT SD&amp;UM IMPLEMENTED API</td>
<td>119</td>
</tr>
<tr>
<td>Table 8</td>
<td>Service Delivery &amp; Utility Manager implemented API definition</td>
<td>119</td>
</tr>
<tr>
<td>Table 9</td>
<td>LIST OF PRIMITIVES COMPRISING THE OPENiOT LSM-LIGHT API</td>
<td>121</td>
</tr>
<tr>
<td>Table 10</td>
<td>LSM-LIGHT API SPECIFICATION</td>
<td>121</td>
</tr>
<tr>
<td>Table 11</td>
<td>MAPPING BETWEEN OPENiOT AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>127</td>
</tr>
<tr>
<td>Table 12</td>
<td>sensiNACT RESOURCES TYPES AND DESCRIPTION</td>
<td>130</td>
</tr>
<tr>
<td>Table 13</td>
<td>sensiNACT RESOURCE’S ACCESS METHOD</td>
<td>131</td>
</tr>
<tr>
<td>Table 14</td>
<td>THE 20 sensiNACT AHA FUNCTIONS</td>
<td>133</td>
</tr>
<tr>
<td>Table 15</td>
<td>sensiNACT AHA HISTORICAL STATISTICS AGENT (30 DAYS HORIZON)</td>
<td>141</td>
</tr>
<tr>
<td>Table 16</td>
<td>SUMMARY OF THE sensiNACT ACTIVAGE DEVELOPMENT TOOLS</td>
<td>146</td>
</tr>
<tr>
<td>Table 17</td>
<td>MAPPING BETWEEN sensiNACT AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>147</td>
</tr>
<tr>
<td>Table 18</td>
<td>FIWARE EXISTING TOUR GUIDES</td>
<td>151</td>
</tr>
<tr>
<td>Table 19</td>
<td>MAPPING BETWEEN IoTIVITY AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>154</td>
</tr>
<tr>
<td>Table 20</td>
<td>MAPPING BETWEEN IoTIVITY AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>161</td>
</tr>
<tr>
<td>Table 21</td>
<td>MAPPING BETWEEN SENiORsome AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>163</td>
</tr>
<tr>
<td>Table 22</td>
<td>HIGH LEVEL PLATFORM OVERVIEW</td>
<td>164</td>
</tr>
<tr>
<td>Table 23</td>
<td>DEVELOPMENT OVER PLATFORM</td>
<td>165</td>
</tr>
<tr>
<td>Table 24</td>
<td>DEVELOPMENT HELPING TOOLS</td>
<td>165</td>
</tr>
<tr>
<td>Table 25</td>
<td>SEMANTIC READY PLATFORM</td>
<td>166</td>
</tr>
<tr>
<td>Table 26</td>
<td>SUPPORT TOOLS</td>
<td>167</td>
</tr>
<tr>
<td>Table 27</td>
<td>IDE TOOLS</td>
<td>167</td>
</tr>
<tr>
<td>Table 28</td>
<td>DATA/VISUAL ANALYTICS TOOLS</td>
<td>167</td>
</tr>
<tr>
<td>Table 29</td>
<td>DATA LAKE TOOLS</td>
<td>167</td>
</tr>
<tr>
<td>Table 30</td>
<td>SEMANTIC INTEROPERABILITY LAYER TOOLS</td>
<td>168</td>
</tr>
<tr>
<td>Table 31</td>
<td>MAPPING BETWEEN universAAL AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>174</td>
</tr>
<tr>
<td>Table 32</td>
<td>MAPPING BETWEEN SOFIA2 AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>177</td>
</tr>
<tr>
<td>Table 33</td>
<td>MAPPING BETWEEN DS6/sensiNACT AND ACTIVAGE DEPLOYMENT TOOLS</td>
<td>182</td>
</tr>
<tr>
<td>Table 34</td>
<td>MAPPING BETWEEN FIWARE AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>183</td>
</tr>
<tr>
<td>Table 35</td>
<td>DEVICE PARAMETERS ACCORDING TO THEIR COMMUNICATION TYPE</td>
<td>185</td>
</tr>
<tr>
<td>Table 36</td>
<td>MAPPING BETWEEN IoTIVITY AND ACTIVAGE DEVELOPMENT TOOLS</td>
<td>191</td>
</tr>
<tr>
<td>Table 37</td>
<td>MAPPING BETWEEN SENiORsome AND ACTIVAge DEVELOPMENT TOOLS</td>
<td>192</td>
</tr>
<tr>
<td>Table 38</td>
<td>DEPLOYMENT TECHNOLOGIES PER PLATFORM</td>
<td>192</td>
</tr>
<tr>
<td>Table 39</td>
<td>DEPLOYMENT TOOLS FOR DEVICE CONFIGURATION</td>
<td>193</td>
</tr>
</tbody>
</table>
Deliverable 4.4 — Developers toolkit and deployment support

TABLE 40: IOT INFRASTRUCTURE MANAGEMENT TOOLS ................................................................. 193
TABLE 41: DEPLOYMENT MANAGEMENT TOOLS ........................................................................... 193
List of figures

FIGURE 1: DEVELOPMENT AND DEPLOYMENT TOOLS FUNCTIONALITIES .................................................. 16
FIGURE 2. PROFILES OF DEVELOPERS THAT ARE FORESEEN 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 5: CLICKDIGITAL LOGO AND ELEVATOR PITCH .................................................................................. 20
FIGURE 6: ACTIVAGE WIKI ........................................................................................................................................ 22
FIGURE 7: THE SUPPORT DEVELOPMENT TOOLS .................................................................................................. 23
FIGURE 8: USE CASE DIAGRAM OF THE DEPLOYMENT TOOLS FUNCTIONALITIES ................................................................. 24
FIGURE 9: POSITIONING OF THE ACTIVAGE DEVELOPMENT TOOLS WITHIN THE OVERALL ACTIVAGE ARCHITECTURE .................................................................................. 25
FIGURE 10: ARCHITECTURE OF THE ACTIVAGE DEVELOPMENT TOOLS COMPONENT AND ITS CONNECTION TO THE OTHER ACTIVAGE COMPONENTS ................................................................................................. 26
FIGURE 11: ACTIVAGE DEVELOPMENT TOOLS .................................................................................................. 27
FIGURE 12: THE SEMANTIC INTEROPERABILITY LAYER (SIL) TOOLS ..................................................................... 28
FIGURE 13: FUNCTIONALITIES AND COMMUNICATION OF THE ACTIVAGE ONTOLOGY EXPLORER .................................................. 29
FIGURE 14: MAIN MENU FOR SELECTING ONE OF THE SEMANTIC INTEROPERABILITY LAYER TOOLS .................................................. 29
FIGURE 15: THE INTERACTIVE GRAPH STRUCTURE OF THE ACTIVAGE ONTOLOGY EXPLORER ................................. 30
FIGURE 16: TEXTUAL DESCRIPTION OF A CLASS IN TURTLE FORMAT .................................................................................. 31
FIGURE 17: REPRESENTATION OF THE OBJECT PROPERTIES .............................................................................. 31
FIGURE 18: FUNCTIONALITIES AND COMMUNICATION OF THE QUERY TRANSLATOR .................................................. 32
FIGURE 19: REPRESENTATIVE SCREENSHOT OF THE QUERY TRANSLATOR TOOL .................................................. 33
FIGURE 20: CONNECTION OF THE DEVICE SEMANTICS EDITOR TO THE SIL .................................................. 34
FIGURE 21: SCREENSHOT OF THE DEVICE SEMANTICS EDITOR THAT REPRESENTS THE EDITABLE INFORMATION RELATED TO A SELECTED DEVICE .................................................................................. 34
FIGURE 22: MENU FOR ADDING NEW DEVICE SEMANTICS .................................................................................. 35
FIGURE 23: CONNECTION OF THE DEVICE SEMANTICS EDITOR TO THE SIL .................................................. 36
FIGURE 24: SCREENSHOT OF THE SERVICE SEMANTICS EDITOR THAT REPRESENTS THE EDITABLE INFORMATION RELATED TO A SELECTED SERVICE .................................................................................. 37
FIGURE 25: THE DATA LAKE MODULES .................................................................................................. 38
FIGURE 26: FUNCTIONALITIES AND COMMUNICATION OF THE ACTIVAGE DATA MODEL WORKBENCH .................................................................................. 39
FIGURE 27: WORKBENCH TOOL, MODEL INTERFACE .................................................................................. 39
FIGURE 28: FUNCTIONALITIES AND COMMUNICATION OF THE METADATA STORAGE EXPLORER .................................................................................. 41
FIGURE 29: WORKBENCH TOOL, META EXPLORER INTERFACE .................................................................................. 41
FIGURE 30: THE DATA / VISUAL ANALYTICS DEVELOPMENT TOOLS .................................................................................. 44
FIGURE 31: FUNCTIONALITIES AND COMMUNICATION OF THE DATA MANIPULATOR DEVELOPMENT TOOL .................................................................................. 45
FIGURE 32: SCREENSHOT OF THE DATA MANIPULATOR TOOL, SHOWING THE DATA LOADING AND FILTERING FUNCTIONALITIES .................................................................................. 45
FIGURE 33: SCREENSHOT OF THE DATA MANIPULATOR TOOL, SHOWING THE ADDITION OF A CALCULATED ATTRIBUTE .................................................................................. 46
FIGURE 34: FUNCTIONALITIES AND COMMUNICATION OF THE DATA ANALYSER DEVELOPMENT TOOL .................................................................................. 47
FIGURE 35: REPRESENTATIVE SCREENSHOT OF THE DATA ANALYSER TOOL .................................................................................. 49
Figure 36: Functionalities and communication of the feature / result viewer development tool ........ 50
Figure 37: Functionalities and communication of the visualization explorer development tools ........ 51
Figure 38: Visualization related case diagram of the ClickDigital IDE visual tool ..................................... 52
Figure 39: The integrated development environment (IDE) components ................................................. 54
Figure 40: Functionalities and communication of the code generator development tool ...................... 55
Figure 41: Functionalities and communication of the code templates development tool ...................... 56
Figure 42: Service Composition Tool instance .......................................................................................... 58
Figure 43: Service composer components ................................................................................................ 58
Figure 44: ClickDigital IDE for programming IoT solutions on top of several IoT platforms .............. 60
Figure 45: Functionalities and communication of the ClickDigital IDE ................................................... 60
Figure 46: Added value of ClickDigital IDE ............................................................................................. 61
Figure 47: Plug capabilities of the ClickDigital IDE .................................................................................. 61
Figure 48: Create capabilities of the ClickDigital IDE .............................................................................. 61
Figure 49: The deliver capabilities of the ClickDigital IDE ...................................................................... 62
Figure 50: ClickDigital system architecture ............................................................................................. 62
Figure 51: ClickDigital – detailed components interactions ..................................................................... 63
Figure 52: ClickDigital – WebSocket components interactions ................................................................ 64
Figure 53: ClickDigital – front-end structure ............................................................................................ 64
Figure 54: ClickDigital – back-end related structure ............................................................................... 65
Figure 55 Mapping between requirements (orange) to the ACTIVAGE development tools (green) ...... 67
Figure 56: Positioning of the ACTIVAGE deployment tools component within the overall ACTIVAGE architecture ........................................................................................................................................... 68
Figure 57: Architecture of the ACTIVAGE deployment tools component and its connection to the other ACTIVAGE components ...................................................................................................................... 71
Figure 58: The ACTIVAGE deployment tools ............................................................................................. 71
Figure 59: The IoT infrastructure management deployment tools .......................................................... 72
Figure 60: Functionalities and communication of the device manager deployment tool ...................... 72
Figure 61: Listing of all registered devices ................................................................................................ 73
Figure 62: Creation of a new device ............................................................................................................ 73
Figure 63: Functionalities and communication of the service manager deployment tool ..................... 74
Figure 64: Listing of all available services ................................................................................................ 75
Figure 65: Editing an already registered service ....................................................................................... 75
Figure 66: Functionalities and communication of the semantic auto-discovery platform deployment tool ........................................................................................................................................... 76
Figure 67: Inserting characteristics of a device for discovery ................................................................... 76
Figure 68: The “Discover similar” option for services .............................................................................. 77
Figure 69: Functionalities and communication of the benchmarking deployment tool ....................... 78
Figure 70 Tool for live monitoring of connected gateways and devices ............................................... 79
Figure 71 Screenshot from the monitoring tool showing the connected devices and registered beneficiaries ........................................................................................................................................... 79
Figure 72 Tool provides a portal with various monitoring and management widgets .............................. 80
Figure 73 Screenshot from the KPI monitoring interface of the tool ........................................................ 80
FIGURE 74: SCREENSHOTS FROM THE sensINact SHOWING INDOOR AND OUTDOOR DEVICE AND SERVICE NAVIGATOR ................................................................. 81
FIGURE 75: THE DEPLOYMENT MANAGEMENT TOOLS ................................................................. 82
FIGURE 76: FUNCTIONALITIES AND COMMUNICATION OF THE DEPLOYMENT MANAGER TOOL ................................................................. 82
FIGURE 77: DEPLOYMENT MANAGER. CREATION OF A NEW DEPLOYMENT INSTALLATION ................................................................. 83
FIGURE 78: DEPLOYMENT MANAGER. ADD DEVICES/SERVICES IN THE INSTALLATION ................................................................. 84
FIGURE 79: DEPLOYMENT MANAGER. VIEW AND EDIT INSTALLATION-SPECIFIC INFORMATION ................................................................. 84
FIGURE 80: FUNCTIONALITIES AND COMMUNICATION OF THE COMPONENT CONFIGURATION DEPLOYMENT TOOL ................................................................. 85
FIGURE 81: COMPONENT CONFIGURATION. CREATION OF A NEW PLATFORM ................................................................. 86
FIGURE 82: COMPONENT CONFIGURATION. VIEW PLATFORM-SPECIFIC INFORMATION ................................................................. 86
FIGURE 83: COMPONENT CONFIGURATION. EDIT PLATFORM-SPECIFIC INFORMATION ................................................................. 87
FIGURE 84: COMPONENT CONFIGURATION. CREATION OF A NEW DEVICE ................................................................. 87
FIGURE 85: COMPONENT CONFIGURATION. VIEW DEVICE-SPECIFIC INFORMATION ................................................................. 87
FIGURE 86: COMPONENT CONFIGURATION. EDIT DEVICE-SPECIFIC INFORMATION ................................................................. 88
FIGURE 87: FUNCTIONALITIES AND COMMUNICATION OF THE MAINTENANCE PANEL DEPLOYMENT TOOL ................................................................. 89
FIGURE 88: MAINTENANCE PANEL. REPORT A NEW ISSUE ................................................................. 90
FIGURE 89: MAINTENANCE PANEL. VISUALIZE, UPDATE OR DELETE CURRENT ISSUES ................................................................. 90
FIGURE 90: UPDATE MANAGER. SERVICES VERSION AND DEPLOYMENT GUIDELINES ................................................................. 91
FIGURE 91: UPDATE MANAGER. SERVICE DEPLOYMENT GUIDE UPDATE ................................................................. 92
FIGURE 92: MAPPING BETWEEN REQUIREMENTS (ORANGE) TO THE ACTIVAGE DEPLOYMENT TOOLS (GREEN) ................................................................. 94
FIGURE 93: UNIVERSAAI IOT TOOLSET (BLUE), MODULES (VIOLET) AND WORKFLOW ................................................................. 102
FIGURE 94 INSTALLATION PROCESS OF AAL STUDIO SUITE IN ECLIPSE ................................................................. 103
FIGURE 95 SOURCE DOWNLOAD INTERFACE ................................................................. 104
FIGURE 96 BASIC INFORMATION PROVIDED FOR THE PROJECT WIZARD ................................................................. 105
FIGURE 97 CUSTOMIZATION OF THE PROJECT DEPENDENCIES AND COMPONENTS IN THE PROJECT WIZARD ................................................................. 105
FIGURE 98 BASIC INFORMATION FOR THE ONTOLOGY PROJECT WIZARD ................................................................. 106
FIGURE 99 VIEW OF RECENTLY CREATED PROJECT ................................................................. 107
FIGURE 100 TOOLTIPS TO CREATE ONTOLOGIES GRAPHICALLY ................................................................. 107
FIGURE 101 A CODE-GENERATION JAVA GENERATION EXAMPLE ................................................................. 111
FIGURE 102 REQUEST DEFINITION USER INTERFACE (UI) ................................................................. 123
FIGURE 103 REQUEST PRESENTATION UI ................................................................. 124
FIGURE 104: sensINact GENERIC DATA MODEL. THIS GENERIC DATA MODEL ALLOWS A SIMILAR ACCESS TO THE SENSORS AND ACTUATORS USING HETEROGENEOUS PROTOCOLS ................................................................. 130
FIGURE 105: sensINact AHA FUNCTIONS (IMPLEMENTED AND IN PROGRESS) ................................................................. 135
FIGURE 106: sensINact AHA SERVICE API SPECIALIZATION ................................................................. 136
FIGURE 107: LIST OF AVAILABLE sensINact AHA SERVICES ................................................................. 137
FIGURE 108: SWAGGER SCREENSHOT FOR THE sensINact AHA SERVICE REST API ................................................................. 138
FIGURE 109: IoTIVITY JAVA API DOCUMENTATION ................................................................. 155
FIGURE 110: A VIEW OF THE IoTIVITY ARCHITECTURE WITH CLOUD FUNCTIONALITY ................................................................. 158
FIGURE 111: ONEIOTA DATA MODELS DEVELOPMENT PROCESS ................................................................. 159
FIGURE 112: SERVICE PROVIDER PERSPECTIVE OF IoTIVITY SIMULATOR ................................................................. 160
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 APIs wrapped into a Software Development Kit (SDK) integrated in a developer’s toolkit.

The document strives for achieving the following more concrete objectives:

– To improve the re-usability, interoperability and sharing of ACTIVAGE cross pilot IoT services/applications (see Section 4.1.1).
– 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 (see Section 4.1.4).
– To provide all the necessary tools for the creation of services/applications from users with minimum technical training (see Section 4.1.4.4).
– To provide, a cloud-based semantic auto-discovery platform component to support the overall deployment process (see Section 5.1.1).
– To provide the ACTIVAGE technology integrators with tools to help the hardware installation and the software deployment (see Section 5.1.2).

This is a technical document. The expected readers of this report are mainly technical teams involved in the AIoTES ecosystem such as users of AIoTES tools in deployment sites, developers of AHA applications, integrators, deployment and installation teams in deployment sites, open callers newly joining to the project, etc.

The 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 and concrete context.

This document aims at answering a set of “how to” questions related to the AHA application development in Section 4. For example, in the context of the application development use cases, we examined the question from the perspective of application developers: “How to use the ACTIVAGE AHA APIs?”; 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 in Section 5, 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 Active and Healthy Ageing
1.1 Deliverable context

<table>
<thead>
<tr>
<th>Project item</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td>D4.4 contributes directly to the following ACTIVAGE objective: O1. To deliver the ACTIVAGE IoT Ecosystem Suite (AIOTES)</td>
</tr>
<tr>
<td>Exploitable results</td>
<td>Each section of the document describes an individual exploitable asset developed/adapted/matured in the project. Furthermore, as a whole, D4.4 becomes one of the necessary outputs for creating the ACTIVAGE IoT Ecosystem Suite as one of the major exploitable results of the project.</td>
</tr>
<tr>
<td>Work plan</td>
<td>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”.</td>
</tr>
<tr>
<td></td>
<td>– 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.</td>
</tr>
<tr>
<td></td>
<td>– The T4.3 task refers to the actual deployment and continuous operation of the ACTIVAGE 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 semantic auto-discovery platform component to support the overall deployment process whenever needed (e.g. new services offered, updated).</td>
</tr>
<tr>
<td>Milestones</td>
<td>D4.4 is a complementary deliverable for assessing the achievement of MS2 - DEMONSTRATE.</td>
</tr>
<tr>
<td>Deliverables</td>
<td>D4.4 is an update of the D4.1 which takes as input the deliverables D3.1 and D3.2. It is an important input for the WP5 deliverables. The most important outcome is the delivery of the tools themselves for supporting the objective. Tools may be used for the implementation of test scenarios, used in the validation and the assessment of the AIoTES.</td>
</tr>
<tr>
<td>Risks</td>
<td>D4.4 contributes to gaining control of the following risk: Rk5: Failure to attract proposals for open call (the existence of tools makes the open calls possible and more attractive) Rk15: Risk of time consuming due to multiple technology (tools are appropriate means for hiding the cumbersome details and atavisms of the various technologies, thus mitigating such risk)</td>
</tr>
</tbody>
</table>

1.2 The rationale behind the structure

After a short overview of the requirements in Section 2 and use case examples in Section 3, the deliverable presents two main parts: Section 4 describes the AIOTES development tools to assist the developers to build applications on top of the AIOTES; Section 5 presents the deployment tools to monitor and manage the applications and data generated by them.
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 [2], technical experts from deployment sites…), 23 requirements related to development tools have been identified and organised into four categories: support, implementation, data processing and IoT infrastructure management (See Figure 1):

- **Support**
  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 3.2.2

- **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.

- **Data processing**
  Data processing support tools allow the operator to (a) perform AIOTES data analytics methods and visualise the results on existing (open) databases 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.

- **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.

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.

- **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.

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

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).
Deliverable 4.4 – Developers toolkit and deployment support

Figure 1: Development and Deployment tools functionalities
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 4.
3.2.1 Indicative use cases for non technical developers

In the analysis of the development tool kit requirements, a particular attention has been paid to non technical developers. ClickDigital tool aims at address in that requirement.

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 pilot sites, including beyond the project duration as a part of the further exploitation and sustainability strategy, an important barrier is the programming complexity & heterogeneity that should be taken into consideration. Therefore, the main vision to enable the widespread and to increase the acceptance of AIOTES is the creation of an “Enabler API”, that eases the integration of platforms, quick development of smart interoperable IoT Services, and also usage of further APIs and services from other platforms. In this context, ClickDigital will bring a added value to the project. “ClickDigital” is 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 developers 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 are:

- 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.

Several use cases have been developed to ease the exploitation of the AIOTES Platform, mainly:

**AIOTES ClickDigital common use cases**

- 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 different available Widgets (Widget for devices mgmt., visualization, 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
AIOTES ClickDigital devices related use cases
- View existing devices
- Find a device
- Add a device
- Monitor/Visualize a device status (Functional visualization)
- Control device
- Control a group of devices

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… )
- Rule based notification
- Visualize rules

Data Visualization related use cases:
- Real time Value
- Bar chart
- Bubble chart
- Doughnut chart
- Line chart
- Polar Area chart

A detailed overview about the system architecture and the implemented usecases is reported in Section 4.

3.2.2 Support

The support modules provide resources and documentation, in order to facilitate the developer in developing applications within the AIOTES infrastructure and resolving issues. The support modules include the following (see also Figure 7):
- **Documentation**: Detailed documentation of all available ACTIVAGE Web APIs, DS experiences, 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.
ACTIVAGE at a glance

Table of contents

1. ACTIVAGE at a glance
   - What is AIOTES
   - Architecture
   - IoT platforms
   - Use Cases and Examples
   - How to integrate your solution with AIOTES?
     - Data Model
     - Semantic Interoperability Layer
     - Tools
     - Security and Privacy
     - Information package and software components available
   - Deployment sites at a glance
   2. ACTIVAGE full documentation

What is AIOTES?

Architecture

ACTIVAGE has defined a reference architecture for IoT Platforms Interoperability. This architecture aims to build general approaches to face the interoperability problem from a vertical way, with the objective of joining as common framework to build interoperable smart living solutions that can be deployed, extended and replicated at Deployment Sites across Europe.

The ACTIVAGE wiki is online at the following address:
https://git.activageproject.eu/Development/Support-Wiki_content/wiki#activage-full-documentation

- **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.
Support tools are available to the developer through a Web-based help center. Hyperlinks to the primary categories of available material, 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.

3.3 Indicative use cases for deployment

Figure 8 illustrates the different steps that a developer would/could have to go through when implementing and deploying 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.

Finally benchmarking helps to monitor various KPIs thus measure the performance (from the user and system perspectives) of the deployment.
Figure 8: 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 9.

![Figure 9: Positioning of the ACTIVAGE development tools within the overall ACTIVAGE architecture.](image-url)

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 are illustrated in Figure 10. 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.

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 about tools that are related to the AIOTES components, rather than 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 11.

ACTIVAGE 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 11 correspond to the AIOTES architectural layers of the AIOTES, namely the Semantic Interoperability Layer, the Data Lake, the Data Analytics and the Visual Analytics components.

In the following sub-sections, the ACTIVAGE development tools of each category are presented in detail, starting from the bottom layer, that corresponds 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 12, are the following:

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

Figure 12: 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 13.

**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/relationship 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 13: Functionalities and communication of the ACTIVAGE ontology explorer.

Take into consideration that because of the close relation of this tool with the Device Semantics Editor and the Service Semantics Editor, which are described in following sections, all three tools are hosted in the same application suite. The user can select which tool to use from a main menu, as it is depicted in Figure 14.

Figure 14: Main menu for selecting one of the Semantic Interoperability Layer tools

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/relationship are 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).

The functionalities provided by the Activage Ontology Explorer are divided into the following 5 tabs:

1. Classes
   - Browsing of all classes through a tree-like menu
   - Representation of all classes through an interactive graph structure
   - Pan and zoom for the navigation through the graph
   - Search class by name
- Display on selection all available information of the selected class and all its instances
- Textual class description in TURTLE format
- Export of textual class description in TURTLE format

2. Annotation properties
- Browsing of all annotation properties through a tree-like menu
- Display on selection all available information of the selected property
- Textual property description in TURTLE format
- Export of textual property description in TURTLE format

3. Object properties
- Browsing of all object properties through a tree-like menu
- Display on selection all available information of the selected property
- Textual property description in TURTLE format
- Export of textual property description in TURTLE format

4. Datatype properties
- Browsing of all datatype properties through a tree-like menu
- Display on selection all available information of the selected property
- Textual property description in TURTLE format
- Export of textual property description in TURTLE format

5. Individuals by class
- Browsing of all ontology instances by class
- Display on selection all available information of the selected instance

Figure 15 shows the “Classes” tab and specifically the interactive graph structure for the representation of all classes. On the left, a menu allows the browsing of all classes, while on
the right a dedicated panel displays details related to the selected class. Figure 16 shows how the user can view the textual description of a class in TURTLE format.

The other tabs utilise a similar approach for the representation of the instances and of the annotation, datatype and object properties. For example, Figure 17 depicts how a user can navigate through the object properties by using the application.

The ACTIVAGE Ontology explorer is a web-based application and is based on the open source Java framework Apache Jena that provides an API for the manipulation of Ontologies. For the implementation of the user interface, the UI framework for JSF called Primefaces is used, while for the creation of the interactive graph the D3 JavaScript library. As the same
application suite hosts also the Device Semantics Editor and the Service Semantics Editor, the same technologies were also used for their implementation.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor">https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor/wiki">https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor/wiki</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/ont-explorer_dev-semantic_serv-semantics">https://docker-registry-activage.satrd.es/repo/tags/ont-explorer_dev-semantic_serv-semantics</a></td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-dSeed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007342&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007343&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-dSeed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007342&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007343&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=</a></td>
</tr>
<tr>
<td>License</td>
<td>Depends on: Primefaces (Apache License Version 2.0), Jena (Apache License Version 2.0), Liferay Portal (GNU Lesser General Public License v2.1), d3.js (BSD 3-clause)</td>
</tr>
</tbody>
</table>

### 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 specific API calls that are made by the SIL towards the 7 IoT platforms available in ACTIVAGE. The platform-specific API calls may be useful to the developer in developing platform-specific applications, and the developer can actually call these APIs through the query translator tool. 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 18.

![Figure 18: Functionalities and communication of the query translator.](image)

Note that in the previous version of this deliverable (D4.1), the query translator was described as translating between the ACTIVAGE data model and the IoT platform data models. In the final version of the query translator, it was considered that it is more important for the developer to be able to view the API calls made by the SIL towards the IoT platforms, since this is the actual manner of interaction between the SIL and the platforms.
Usage

As seen in Figure 19, the query translator offers a text editing window, through which the developer can write an input query. By clicking on a translation button, the input query is translated into the platform-specific API calls for data retrieval, which are presented as a list of URLs. Each API call can be expanded to view further details, such as the HTTP method and headers used, and the body of the sent message. The developer can use this information to add it in their own application. The tool also offers the functionality to call each API directly and view its result, i.e. the data retrieved from a single platform. The user can download the response in a local file, to examine it further.

![Figure 19: Representative screenshot of the query translator tool.](image)

The query translator tool is a web-based application, using the SIL translation web services in the background. For the implementation of the HTML user interface, the W3.CSS\(^2\) styling library is used, while JavaScript is used to implement the functionality. The query editor part uses the Ace\(^3\) library, which is a JavaScript-based text editor.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Development/SIL-Query_translator">https://git.activageproject.eu/Development/SIL-Query_translator</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/query_translator">https://docker-registry-activage.satrd.es/repo/tags/query_translator</a></td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5edd63f8307/ShowPage?returnView=0&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007408&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007409&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5edd63f8307/ShowPage?returnView=0&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007408&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007409&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=</a></td>
</tr>
<tr>
<td>License</td>
<td>The tool depends on: Angular JS MIT license and npm modules licenses</td>
</tr>
</tbody>
</table>

\(^2\) [https://www.w3schools.com/w3css/](https://www.w3schools.com/w3css/)

\(^3\) [https://ace.c9.io/](https://ace.c9.io/)
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. The device semantics editor is connected to the SIL, in order to retrieve the Activage Ontology. However, in order to ensure that the semantics of existing devices currently used by applications are not altered, and thus avoid application malfunctioning, the application does not support the modification of the Ontology that is deployed on the project server but allows the export of the updated Ontology in OWL format for local use. The connection of the device semantics editor to the SIL is depicted in Figure 20.

Figure 20: 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. 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 (Figure 21). 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 (Figure 22).

Figure 21: Screenshot of the Device Semantics Editor that represents the editable information related to a selected device.
Finally, the user can delete at will any of the semantic information that he/she created by using the equivalent option of the bottom menu. In the same menu, there is also an option for exporting the updated ontology in OWL format. To summarize the supported functionalities of this tool are the following:

- Browsing of all device semantics
- Creation of a new device class / edit / delete
- Creation of an annotation property for a device class / edit / delete
- Change parent class of a device class
- Creation of a new restriction for a device class / edit / delete
- Creation of a datatype property that has as domain a device class / edit / delete
- Creation of an object property that has as domain a device class / edit / delete
- Export of the update ontology in owl format

The device semantics editor is a web-based application and its implementation was based on the same technologies that were used for the Ontology Explorer.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor">https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor/wiki">https://git.activageproject.eu/Development/OntologyExplorer_DeviceSemanticsEditor_ServiceSemanticsEditor/wiki</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/ont-explorer_dev-semantic_serv-semantics">https://docker-registry-activage.satrd.es/repo/tags/ont-explorer_dev-semantic_serv-semantics</a></td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=</a></td>
</tr>
</tbody>
</table>
4.1.1.4 Service 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. The service semantics editor is implemented as a form through which the developer can specify the semantics of the service’s functionality, its inputs and outputs. The service semantics editor is connected to the SIL, in order to retrieve the Activage Ontology. The updated ontology can be exported in OWL format. The connection of the service semantics editor to the SIL is depicted in Figure 23.

![Figure 23: Connection of the device semantics editor to the SIL](image)

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. 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 (Figure 24). 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. Similarly, the user can use this tool in order to manipulate the semantics related to service data formats, service input and service output.
To summarize, the supported functionalities of this tool are the following:

- Browsing of all service semantics
- Creation of a new class / edit / delete
- Creation of an annotation property related to services / edit / delete
- Change parent class of a class
- Creation of a new restriction for a class / edit / delete
- Creation of a datatype property related to services / edit / delete
- Creation of an object property related to services / edit / delete
- Export of the update ontology in owl format

The service semantics editor is a web-based application and its implementation was based on the same technologies that were used for the Ontology Explorer.
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 25:

- ACTIVAGE data model workbench
- Metadata storage explorer

Each tool is described in the following sub-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 26.
Usage

The ACTIVAGE data model workbench offers a Graphical User Interface (GUI) for viewing the data available and executing queries.

The main functionalities are related to the model management.

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.
In the Database management screen, the tree view shows the list of the models, and options. While the Right screen shows the list of models, so that a user can easily edit, create or view details. The user can: Create model, Edit model, Update existing models and “view detail model” to see model details e.g. model params.

The data model tool is also accessible via API, e.g.:

- For model management

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/_search/models</td>
<td>searchModels</td>
</tr>
<tr>
<td>GET</td>
<td>/api/models</td>
<td>getAllModels</td>
</tr>
<tr>
<td>POST</td>
<td>/api/models</td>
<td>createModel</td>
</tr>
<tr>
<td>PUT</td>
<td>/api/models</td>
<td>updateModel</td>
</tr>
<tr>
<td>DELETE</td>
<td>/api/models/[id]</td>
<td>deleteModel</td>
</tr>
<tr>
<td>GET</td>
<td>/api/models/[id]</td>
<td>getModel</td>
</tr>
</tbody>
</table>

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.

### Source code

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Data_Analytics/DL-datamodel_workbench">https://git.activageproject.eu/Data_Analytics/DL-datamodel_workbench</a></th>
</tr>
</thead>
</table>

### Wiki

|------|------------------------------------------------------------------------------------------------------------------|

### Docker registry

<table>
<thead>
<tr>
<th>Docker registry</th>
<th><a href="https://docker-registry-activage.satrd.es/repo/tags/dl-datamodel-workbench">https://docker-registry-activage.satrd.es/repo/tags/dl-datamodel-workbench</a></th>
</tr>
</thead>
</table>

### Course

- LESSONS > 2. AIoTES Framework > 2.4. Development tools > 2.4.5 ACTIVAGE datamodel workbench

### License

- Depends on: Spring Boot (Apache License Version 2.0), Elasticsearch (Apache License Version 2.0), MongoDB (Apache License Version 2.0), AngularJS (Apache License Version 2.0)

#### 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 28.
Usage

The Metadata storage explorer offers a GUI for viewing the metadata available and executing queries.

![Metadata storage explorer diagram](image)

The main functionalities are:

- Database management
- Table management
- Schema management
- Query Editor
- Data Export

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.

In the Database management screen, the tree view shows the list of the databases, and their tables. While the Right screen shows the list of databases, so that a user can easily edit, delete or view details. The user can: Create database, Edit database, Export data on the query response, Create new tables, Update existing tables.

In the Schema management, similar idea to database management, on left menu, you can see “Schemas” on the menu tree view plus various options. The user can: Create schemas, Edit existing schemas, query some data using the Query Editor on Schema.

The Metadata storage explorer is also accessible via API, e.g.:
- For database management

<table>
<thead>
<tr>
<th>database-resource</th>
<th>Database Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /api_search/databases</td>
<td>searchDatabases</td>
</tr>
<tr>
<td>GET /api/databases</td>
<td>getAllDatabases</td>
</tr>
<tr>
<td>POST /api/databases</td>
<td>createDatabase</td>
</tr>
<tr>
<td>PUT /api/databases</td>
<td>updateDatabase</td>
</tr>
<tr>
<td>DELETE /api/databases/{id}</td>
<td>deleteDatabase</td>
</tr>
<tr>
<td>GET /api/databases/{id}</td>
<td>getDatabase</td>
</tr>
<tr>
<td>GET /api/databases/{id}/tables</td>
<td>getDatabaseTables</td>
</tr>
</tbody>
</table>

- For schema management

<table>
<thead>
<tr>
<th>schema-resource</th>
<th>Schema Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /api_search/schemata</td>
<td>searchSchemata</td>
</tr>
<tr>
<td>GET /api/schemata</td>
<td>getAllSchemata</td>
</tr>
<tr>
<td>POST /api/schemata</td>
<td>createSchema</td>
</tr>
<tr>
<td>PUT /api/schemata</td>
<td>updateSchema</td>
</tr>
<tr>
<td>DELETE /api/schemata/{id}</td>
<td>deleteSchema</td>
</tr>
<tr>
<td>GET /api/schemata/{id}</td>
<td>getSchema</td>
</tr>
</tbody>
</table>
4.4 Developers toolkit and deployment support

For table management

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.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Data_Analytics/DL-Metadata_storage_explorer">https://git.activageproject.eu/Data_Analytics/DL-Metadata_storage_explorer</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Data_Analytics/DL-Metadata_storage_explorer/wiki/home">https://git.activageproject.eu/Data_Analytics/DL-Metadata_storage_explorer/wiki/home</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/dl-metadata-storage-explorer">https://docker-registry-activage.satrd.es/repo/tags/dl-metadata-storage-explorer</a></td>
</tr>
<tr>
<td>Course</td>
<td>LESSONS &gt; 2. AoTES Framework &gt; 2.4. Development tools &gt; 2.4.6 Metadata storage explorer</td>
</tr>
<tr>
<td>License</td>
<td>Depends on: Spring Boot (Apache License Version 2.0), Elasticsearch (Apache License Version 2.0), MongoDB (Apache License Version 2.0), AngularJS (Apache License Version 2.0)</td>
</tr>
</tbody>
</table>

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 30:

- Data manipulator
- Data analyser
- Feature / result viewer
- Visualization explorer
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 31.

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 spreadsheet 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:
  - Sort records by column.
  - Filter records, by inserting filters for each attribute (column). The filters can include selecting a specific or multiple attribute values, or by a condition.
  - Select/deselect subsets of attributes (columns).
  - Create new attributes, by transforming existing ones (e.g. multiplying by a value or adding two attributes).
The final data, after these preprocessing operations, can be exported to a CSV or JSON file, in order to be further used by the user, as needed. For instance, the exported file could be used as the input to the Data Analyser tool (Section 0), in order to be further analysed.

Note that in the previous version of this deliverable (D4.1), an "Analyse" was described as a functionality of the data manipulator, which was meant to be a button that automatically opens the data analyser, with the manipulator's output as the analyser's input. This feature was ultimately not included in the final version of the data manipulator, in order to keep the functionalities of each component as independent from other components as possible. The functionality is indirectly supported by the "Export to files" functionality, since the exported file can be readily be used in the data analyser or any other tool used by the user.
spreadsheet view at the right part of the screen. The user can filter the data by selecting values in a column, or by specifying a condition.

As seen in Figure 32, the user can add more attributes (columns) to the displayed data, by generating them from the existing columns. The user can provide an expression which is evaluated for each row by taking into account the corresponding rows from the other columns and the generated values are added as an extra column. This can be used, e.g. to compute sums or averages, or to format a column.

![Data manipulator tool screenshot](image)

**Figure 33**: Screenshot of the Data manipulator tool, showing the addition of a calculated attribute.

The data manipulator tool is a web-based application, using Data Lake API in the background, for loading data from AIOTES. The Data Lake API in turn uses the SIL API to actually retrieve the data. For the implementation of the HTML user interface, the W3.CSS\(^4\) styling library is used, while JavaScript is used to implement the functionality. The spreadsheet view is implemented using the Handsontable\(^5\) library.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Development/DVA-Data_manipulator">https://git.activageproject.eu/Development/DVA-Data_manipulator</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Development/DVA-Data_manipulator/wiki">https://git.activageproject.eu/Development/DVA-Data_manipulator/wiki</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/data-manipulator">https://docker-registry-activage.satrd.es/repo/tags/data-manipulator</a></td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007412&amp;newTopLevel=false&amp;postedComment=false">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007412&amp;newTopLevel=false&amp;postedComment=false</a></td>
</tr>
</tbody>
</table>

\(^4\) https://www.w3schools.com/w3css/

\(^5\) https://handsontable.com/
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 exported to a file in order to be further processed or visualized by the user. 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 (text boxes, select boxes, 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 depicted in Figure 34.

![Figure 34: Functionalities and communication of the data analyser development tool.](image)

Note that in the previous version of this deliverable, the data analyser was considered directly connected to the data manipulator tool, for receiving its input, and to the feature/result viewer tool (Section 4.1.3.3), to show its output in a visual manner. In the current version of the deliverable, and in the final implementation of the data analyser tool, the tool has been detached from the other two tools, in order to be modular and reusable. The results of the data analyser tool, i.e. the analysis results, can be exported in CSV or JSON format, in order to be further used in any way the user wishes. Its input can also be a local file, which could be the output of the data manipulator tool.

**Usage**

As can be seen in Figure 35, the Data analyser tool offers a Web-based GUI, with the following main components:
Deliverable 4.4 — Developers toolkit and deployment support

- A data selection panel (left part), 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.
- A table view (middle part, top) showing the loaded data in a tabular format.
- An analysis type selection drop-down menu (right part, top), 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 (right part, middle) 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 (right part, bottom) 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 select boxes.
- A table view (middle part, bottom) of the analysis results, e.g. the list of record IDs, with an added “anomaly score” column, indicating the “outlierness” of each record, or with a cluster label denoting the cluster each record belongs to. The analysis results can be downloaded by the developer in CSV or JSON format.
- 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.

The data analyser tool is a web-based application, communicating with the Data Analytics API to analyse the data (see deliverable D4.6 for more details). The tool has been implemented in Angular 7, using the Nebular library for the user interface design. For the table views, the Smart Table library was used.

---

6 https://angular.io/
7 https://akveo.github.io/nebular/
8 https://github.com/akveo/ng2-smart-table
Figure 35: Representative screenshot of the data analyser tool.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Development/Data_Analyser">https://git.activageproject.eu/Development/Data_Analyser</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Development/Data_Analyser/wiki/Home">https://git.activageproject.eu/Development/Data_Analyser/wiki/Home</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/data-analyser">https://docker-registry-activage.satrd.es/repo/tags/data-analyser</a></td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007368&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007369&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007368&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007369&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=</a></td>
</tr>
<tr>
<td>License</td>
<td>Depends on: jQuery (MIT license), jQuery DataTables (MIT license), D3 (BSD 3-Clause license)</td>
</tr>
</tbody>
</table>

**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 36.

![Figure 36: Functionalities and communication of the feature / result viewer development tool.](image)

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.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Development/DVA-Feature_result_viewer">https://git.activageproject.eu/Development/DVA-Feature_result_viewer</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Development/DVA-Feature_result_viewer/wiki">https://git.activageproject.eu/Development/DVA-Feature_result_viewer/wiki</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/feature_result_viewer">https://docker-registry-activage.satrd.es/repo/tags/feature_result_viewer</a></td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?familyId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007422&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=false&amp;itemId=6007423&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?familyId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007422&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=false&amp;itemId=6007423&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=</a></td>
</tr>
<tr>
<td>License</td>
<td>-</td>
</tr>
</tbody>
</table>
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 37.

**Figure 37: Functionalities and communication of the visualization explorer development tools**

**Usage**

The ClickDigital visualization explorer is an independent Web based manager to automatically personalize, interact and visually analyse data based on different types of diagrams depending on the visualization context and purpose. The below figure illustrates the use cases a developer and the end users can be perform from a visualization perspective based on the AIOTES framework, mainly:
The following Vizual analytics tools are integrated within the ClickDigital IDE with a very flexible personalisation of the mainly:

- Real time Value
- Line Chart

- Bar chart
- Bubble chart

- Doughnut chart

- Polar Area chart
A very detailed overview about the concrete implemented Vizual Analytics diagram and tools might be found in Deliverable D4.6.

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

![Diagram of IDE components](image)

**Figure 39:** The Integrated Development Environment (IDE) components.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Development/IDE-AIOTES_IDE">https://git.activageproject.eu/Development/IDE-AIOTES_IDE</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/aiotes-ide">https://docker-registry-activage.satrd.es/repo/tags/aiotes-ide</a></td>
</tr>
<tr>
<td>Course</td>
<td>LESSONS &gt; 2. AIoTES Framework &gt; 2.4. Development tools &gt; 2.4.8 AIOTES IDE</td>
</tr>
<tr>
<td>License</td>
<td>Depends on: express (MIT license), Request (Apache License Version 2.0)</td>
</tr>
</tbody>
</table>
4.1.4.1 Code generator

The code generator development tool provides generated code for manipulating devices and sub-services, among other patterns, within a larger application. The code generator allows the developer to select among available components and services using a graphical user interface or API. The developer can select among the functions available for any component, in the ontology, as they are registered in the ACTIVAGE ontology. The developer can generate source code for implementing these models and 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 40.

Usage

The codegenerator tool integrates with the Protégé tool, and Maven projects, as well as offering a stand alone REST web service. The code generator allows the developer to select one, or more, ontologies (and possibly their imported references recursively); together with the selection of a template system and a set of variable values, code generator performs translations from the ontology to a given code. According to the provided templates, the code generator is capable of representing the ontology using any of the IoT platforms (particularly interesting for non-semantic platforms). If the translated ontology consists of the extension of a specific ontology the template can create specific code for the model. For instance, the developer can define a temperature sensor extending it from the AIOTES data model, which has a “get status” and a “collect data” functionality, or a lamp, which as a “get status”, a “switch on” and a “switch off” functionality. The developer can select the desired functionality from the ontology and generate the source code needed to perform this functionality for the specific device/service.
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, or components, for the individual IoT platforms registered in the ACTIVAGE federation, as well as essential customizable components of AIOTES, such as SIL bridges. 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 41.

**Figure 41**: 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 clone or fork the code, where the code will be used. Each template is complete with a README file explaining the building steps, test and possible configuration for the end component; this readme also contains the steps to follow to instantiate the template, links to the appropriate courses (See D5.6) and the process to incorporate the results to the code base.
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. In this tool, a Flow Based Programming approach has been followed. In this approach, the code blocks, called nodes, perform the access to the IoT services and applications. The selected implementation of this paradigm offers a visual programming approach that allows developers to connect a set of predefined nodes together to perform a task. The connected nodes, when wired together, make up ‘flows’ of interoperability between services and applications, offering the service composition capabilities.

The existent graphical tool selected to perform this paradigm is Node-RED. This tool is an open-source project hosted on GitHub. There is already an active community regularly producing new nodes and is extensively used in the domain of IoT. A large number of open source nodes are developed in order to include the services of the main IoT platforms.

The current implementation of the service composition tool takes as a core element the aforementioned Flow Based Programming tool. For that reason, Node-RED has been extended and customized in order to add some new capabilities and to be adapted in order to satisfy the requirements of Activage. Some extra components are developed during the Activage Project to perform the access, use, import, export, catalog, discovery and combination of heterogeneous services from AloTES and the IoT Platforms of the Activage ecosystem.

An instance of the service composition tool offers a complete interoperability architecture. In addition, the dockerization of the instances of the service composition tool allows access to different instances of this solution on the same host and provides an easy way to manage them. Each instance of the server includes its own flow of interoperability, properties and purpose.

As can be seen in the previous paragraphs, nodes and flows are the two main elements in this tool. In order to provide a continuous extensibility of the solution, users and developers can contribute in two ways. The first one is the creation of nodes, compatible with Service Composition tool, to implement the access and the functionalities of their services. The second one is the creation of flows of interoperability between IoT services and applications, to offer a composition of services.

Regarding the development of new nodes. IoT services provide an API to access their functionalities. They usually offer a REST API or other alternatives like SOAP Web Services, programming libraries or wrapping their information to access to it. In order to create a mechanism to make the access to the services compatible with Service Composition Tool, the users need to develop pieces of code to wrap the functionalities of these services. This task is done taking advantage of their available API, libraries, etc. Those pieces of code are the nodes and provide the mechanism to access and interact with the IoT service.

A node needs input parameters and offers output information. It executes a series of internal processes in the service or application that is calling. Paying attention to technical issues, a node consists in a JavaScript file that runs in the Service Composition tool instance, and a HTML file consisting in a visual description of the node and interface to provide its input elements.

Regarding the flows, the use of the graphical tool in the browser facilitates the creation of flows, which are a collection of nodes wired together to exchange messages. In a technical way, a flow consists of a list of JavaScript objects that describe the nodes and their configurations, as well as the list of downstream nodes they are connected to, the wires.

The wires define the connections between node input and output endpoints in a flow. They connect the output endpoints of nodes to inputs of downstream nodes indicating that
messages generated by one node should be processed by the next connected node. Messages passed between nodes are, by convention, JavaScript Objects, consisting of a set of named properties. A message contains a payload and may attach other properties, which can be used to carry other information into the next node in the flow. The flow created is stored in a JSON file; it can be exported and can be reused in other instances of the service composition 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:

- 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 JSON description of the services and their connections is edited.
  - 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 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 CIekDigital - 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:

ClickDigital is a visual, pluggable, user friendly Integrated Development Environment (IDE) for IoT platforms. It allows you to quickly prepare smart digital solutions and offer the resulting Application to your clients. ClickDigital aims at:

- Decreasing the learning curve/complexity of creating applications for heterogenous IoT platforms
- Offering a new application creation experience for the developers of IoT solutions
- Optimizing the path and time-to-market
- Enabling IT departments to develop quicker, optimize their costs and enhance the usage of IoT solutions

The main goals of 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 strategic vision of "IoT for all"

ClickDigital addresses mainly developers of IoT solutions & applications. These developers present our target users group, mainly because we solve following of their pain-points:

- 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
- Use the time to extend and empower the created system quality
- Reduce the time-to-market

The second target user group are the IT departments of different service provider companies. 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
- Shorten the path-to-market
- Enable a quick time reaction toward smart solutions adjustment, modification and extension
- Empower a user-centric approach through the direct contact between the IT departments and the respective service providers.

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

- It is a pluggable, visual IDE for IoT platforms, with which one can create their own IoT dashboard and easily present it in different views to your clients.
- It covers the spectrum of needed use-cases for a successful IoT application, mainly: management, visualization, control, creation of logic.

This tool will enable visual programming of IoT solutions based on different IoT platforms, including AIOTES. It will cover mainly following domains: Smart Living, building management, eHealth, energy economy.

![ClickDigital IDE for programming IoT solutions on top of several IoT platforms](image)

The ClickDigital IDE communicates with the Semantic Interoperability Layer (SIL), in order to have access to the available components and services that can be used for visual dashboard creation. The functional blocks and the communication to the SIL of the ClickDigital IDE are presented in Figure 45.

![Functionalities and communication of the ClickDigital IDE.](image)
Usage
As shown in Figure 46, the added value of ClickDigital is to: plug, create, and deliver.

Create

The plug capability aims to offer possibilities to the developers to plug it to different IoT platforms based on partners/clients requirements (Figure 47).

Once plugged, ClickDigital, through its widgets, allow the users to select, drag and drop different widgets to the main dashboard screen. This will enable them to visually plug new devices, control them, visualize data and create rules.

Once created, ClickDigital offers the possibility to deliver the created application and dashboard in a use-only mode to the target users.
Figure 49: The deliver capabilities of the ClickDigital IDE.

4.1.4.4.1 ClickDigital system architecture

This section describes the final system architecture. More details on the ClickDigital system implementation can be found in the Appendix Section Appendix C.

The final system architecture of ClickDigital is mainly composed of two components. A front-end and a back-end. The front-end offers the ClickDigital user interface and connects to the back-end. The back-end is connected to a database on one side and to IoT from the other side.

Figure 50: ClickDigital system architecture
Figure 51: ClickDigital — detailed components interactions
Deliverable 4.4 – Developers toolkit and deployment support

Figure 52: ClickDigital – Websocket components interactions

Figure 53: ClickDigital – Front-end structure
Deliverable 4.4 — Developers toolkit and deployment support

Figure 54: ClickDigital — back-end related structure

- Backend
  - Visualmanager
    - Models
    - Platforms
  - Rulemanager
    - Models
    - Platforms
  - Platformmanager
    - Models
    - Platforms
  - Usermanager
    - Models
    - Platforms
  - Anomalymanager
    - Models
    - Nupic
    - Logfilter
  - Acpmanager
  - Dataprotectionmanager
  - Userssessionmanager
  - Authentication
  - Exceptions
  - Frontend
    - Aiotes
    - Openhab
  - Platforms
  - Services

Version 1.0 | 2019-Aug-05
ACTIVAGE ©
ClickDigital Technology:
From technological perspective, the following technologies have been chosen for the ClickDigital development:
- Frontend: Angular 2, Typescript, HTML, CSS Deployment: Apache Web Server
- Backend: Java, MongoDB Deployment: Glassfish Server

<table>
<thead>
<tr>
<th>Source code</th>
<th>Source code not available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Development/IDE-ClickDigital/wiki/_pages">https://git.activageproject.eu/Development/IDE-ClickDigital/wiki/_pages</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/clickdigital">https://docker-registry-activage.satrd.es/repo/tags/clickdigital</a></td>
</tr>
<tr>
<td>Course</td>
<td>Lessons -&gt; 2. AloTES Framework -&gt; 2.4. Development tools -&gt; 2.4.11 ClickDigital</td>
</tr>
<tr>
<td>License</td>
<td>Depends on: Google Gson, MongoDb driver, log4j, jjwt, Jena, rxjs (Apache License Version 2.0), Jersey Glassfisch (Eclipse Public License Version 2.0, GNU General Public Licence Version 2.0), Angular, core-js, hammerjs, http-status-codes, jquery, mode-saas, priming, zone.js, karma, angular-graidster (MIT License), org.json (JSON License), font-awsome (SIL OFL 1.1 License).</td>
</tr>
</tbody>
</table>

4.1.5 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 55 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 55 Mapping between requirements (Orange) to the ACTIVAGE development tools (green)
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 AoTES. 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 highest level of the AoTES architecture, along with development tools and data analytics. The positioning of the deployment tools within the overall ACTIVAGE architecture is depicted in Figure 56.

Figure 56: 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 running 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.

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 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 57.

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 58. 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.

---

9 Passive Infrared sensor
– **Deployment management tools**: Tools for component configuration and management of installations in deployment units.

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 59.

– Device manager
– Service manager
– Semantic auto-discovery platform
– Benchmarking
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 etc. 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. The device manager tool is connected to the Semantic Interoperability Layer of the AIOTES, as depicted in Figure 60, in order to have access to the device ontologies and semantic attributes.

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 information that can be inserted about a device is related to the information supported by the Activage Ontology:

– Name
– Description
– Device abilities (Sensors or actuators) hosted by the device

The following figure demonstrates an example of how all registered devices are listed. Next to each device there is the option for edit or delete.
In case the user decides to register a new device, then a form enables the input of all needed information.

The device manager is connected with metadata storage mechanism, which is part of the Dast Lake. It utilises its API in order to store, update, list or delete a device.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/IM-Device_manager">https://git.activageproject.eu/IM-Device_manager</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/device_manager">https://docker-registry-activage.satrd.es/repo/tags/device_manager</a></td>
</tr>
<tr>
<td>Course</td>
<td>('=false&amp;addBefore=&amp;itemId=6007448&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id='</td>
</tr>
</tbody>
</table>
**SSIL extension**

An extension of the application functionality will be presented in case the interoperability layer extension allows the retrieval of information related to the devices connected to an IoT platform. In this case, the Device Manager will communicate with IoT platforms through the SSIL and it will be able to provide a mechanism for managing all connected devices.

**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. The service manager provides forms through which the developer can provide all this information, as well as update it when necessary. The service manager tool is connected to the AIOTES SIL, as depicted in Figure 63, in order to have access to the service-related ontologies and semantic mappings.

![Figure 63: Functionalities and communication of the service manager deployment tool.](image)

**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.
Figure 64: Listing of all available services

Figure 65: Editing an already registered service

The current version of this tool allows the manipulation of all services that are used by the query mechanism, described in deliverable D4.6 “Data Layer Support Tools”. After the implementation of SSIL, it will be further extended in order to provide access to the Service Registry of the SSIL. More information about this component can be found in deliverable D3.11 “Interoperability report”

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Deployment/IM-Service_manager">https://git.activageproject.eu/Deployment/IM-Service_manager</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://git.activageproject.eu/Deployment/IM-Service_manager/wiki">https://git.activageproject.eu/Deployment/IM-Service_manager/wiki</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/service_manager">https://docker-registry-activage.satrd.es/repo/tags/service_manager</a></td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007461&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007462&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5eed63f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007461&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;itemId=6007462&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=</a></td>
</tr>
<tr>
<td>License</td>
<td>Depends on: Primefaces (Apache License Version 2.0), Jena (Apache License Version 2.0), Liferay Portal (GNU Lesser General Public License v2.1)</td>
</tr>
</tbody>
</table>

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 66.

![Figure 66: Functionalities and communication of the semantic auto-discovery platform deployment tool.](image)

**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. An example of this form is represented in Figure 67.

- Through the selection of another device/service and selecting a “discover similar devices/services” option (Figure 68).

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.

![Figure 67: Inserting characteristics of a device for discovery](image)
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 69.

![Figure 69: Functionalities and communication of the benchmarking deployment tool.](image)

### 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.

CEA has developed the tool and tested it in the ISERE deployment. For the Panels 1 and 2, CEA provides the production server, building a private secured network with all gateways in elderly homes (80 homes planned). Gateways are preinstalled in the CEA facilities using prebuilt images. Each gateway is preconfigured with specific sensor devices to be installed around it: the gateway + sensors are packed together in a "kit" provided to the installer. The installer is in charge of the pairing sensors-gateway, and installs the gateway and the sensors in the elderly home. The support team is then in charge of the maintenance.

For each steps of the installation process, some tools are developed:

- **At preinstallation step**
  
  The sd card image to be installed in the gateway is generated by a dedicated software.

- **At pre-configuration step**
  
  The preconfiguration is automated through a dedicated script that generates all required AHA functions configuration files specialized for the ACTIVAGE functions.

- **At pairing step**
  
  The pairing can be done through the openHAB web interface embedded in the gateway.

- **At installation step**
  
  A web app embedded in the gateway can be accessed by the installer at any time during the installation, to check the status of data connectivity between the installed sensors and the gateway.

  In the production server, several dashboards are online to provide status of gateways in the ACTIVAGE DS6 panel1&2 network.
Figure 70 Tool for live monitoring of connected gateways and devices

Figure 71 Screenshot from the monitoring tool showing the connected devices and registered beneficiaries
Figure 72 Tool provides a portal with various monitoring and management widgets.

Figure 73 Screenshot from the KPI monitoring interface of the tool.
The tool can be interfaced with other tools such as:

- **At support step**
  
  A ticketing system is running, in addition with a call center to gather issues from beneficiaries.
  
  The client entry (to enter a new bug and check resolution status):
  
  http://193.48.18.249/otrs/customer.pl
  
  The agent entry (to diagnose, propose workarounds, declare fixes):
  
  http://193.48.18.249/otrs/index.pl

- **At maintenance step**
  
  The same dashboards used at installation step are used during maintenance.
  
  The KPI dashboard provides high level key performance indicators.
  
  A unique DS6 Activage portal is inline to give simple access to installation and maintenance tools (including also extra links to communication and testing tools)
  
  http://193.48.18.249/

In addition, the sensiNact Studio tool embedded in the sensiNact platform has been designed to check the status of deployed devices and services. This Eclipse based plug-in make it possible to localize devices through the ‘Navigator’ panel (cf. Figure 74). This Studio navigator gave the status of both installed devices and deployed services.

![Figure 74: Screenshots from the sensiNact showing indoor and outdoor device and service navigator](image)

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://projects.eclipse.org/projects/technology.sensinact/developer">https://projects.eclipse.org/projects/technology.sensinact/developer</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td><a href="https://wiki.eclipse.org/SensiNact">https://wiki.eclipse.org/SensiNact</a></td>
</tr>
<tr>
<td>Docker registry</td>
<td>NA</td>
</tr>
<tr>
<td>Course</td>
<td><a href="https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5e6d3f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007464&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;(itemId=6007465&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=">https://poliformat.upv.es/portal/site/ESP_0_2626/tool/4136ab45-e867-4287-ac8e-d5e6d3f8307/ShowPage?returnView=&amp;studentItemId=0&amp;backPath=&amp;errorMessage=&amp;clearAttr=&amp;source=&amp;title=&amp;sendingPage=6007464&amp;newTopLevel=false&amp;postedComment=false&amp;addBefore=&amp;(itemId=6007465&amp;path=push&amp;addTool=-1&amp;recheck=&amp;id=</a></td>
</tr>
<tr>
<td>License</td>
<td>Eclipse Public License (EPL)</td>
</tr>
</tbody>
</table>
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 75.

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

![Deployment management tools](image)

**Figure 75: 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 deployment manager communicates with the Metadata Storage Server, as depicted in Figure 76, in order to retrieve all deployment-related metadata.

![Deployment manager interactions](image)

**Figure 76: 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 list 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 in the installation. The devices can be added to the tree-like structure by selecting among the devices registered in AIOTES
- View/insert/edit installation-specific information for the devices of an installation, through edit forms.
- Configure selected devices, through the component configuration tool
- View maintenance information for selected devices, through the maintenance panel tool.

Development and deployment

The tool has been developed to consume the devices defined in the Metadata StorageServer by the Device Manager and produce Deployment units. In this sense, a Deployment unit is a logical unit that groups devices and services, providing isolation and a better understanding of the physical deployment of devices.

The final development of the tool provides a REST API that can be consumed directly by a developer for the creation and modification of Deployment units or, in a more user-friendly way, through the graphical interface.

The tool has been developed using the Java programming language, relying on technologies such as JAX-RS, Apache Tomcat and Maven. Finally the application has been packaged using Docker and is available in the project repository.

Graphical interface functionalities

The main objective of this section is to present as an image the main functionalities of the tool.
SSIL extension

In the previous version of this document, this application had been described to provide similar functionality for both devices and services. The current application limits its functionality to devices, since the current version of the interoperability layer did not allow the description of services.

In this way, an extension of the application functionality is considered only if the implementation of the interoperability layer allows the semantic description of the services associated with a platform. Adding the following functionality:

- Add services in the installation. The services can be added to the tree-like structure by selecting among the services registered in AIOTES.
- View/insert/edit installation-specific information for the services of an installation, through edit forms.
– Configure selected services, through the component configuration tool
– View maintenance information for selected services, through the maintenance panel tool.

<table>
<thead>
<tr>
<th>Source code</th>
<th><a href="https://git.activageproject.eu/Deployment/DM-Deployment_manager">https://git.activageproject.eu/Deployment/DM-Deployment_manager</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker registry</td>
<td><a href="https://docker-registry-activage.satrd.es/repo/tags/deployment-manager">https://docker-registry-activage.satrd.es/repo/tags/deployment-manager</a></td>
</tr>
<tr>
<td>Course</td>
<td>Lessons &gt; 2. AloTES Framework &gt; 2.5. Deployment tools &gt; 2.5.5 Deployment manager</td>
</tr>
<tr>
<td>License</td>
<td>The tool is distributed under Apache v2.0 license. The tool depends on: Angular JS MIT license, npm modules licenses and maven artifact licenses. In any case one of this licenses are more restrictive than the one under which the tool is distributed.</td>
</tr>
</tbody>
</table>

5.1.2.2 Component configuration (HOPU)

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. The component configuration provides also a The component configuration is connected to the Semantic Interoperability Layer, in order to store the device/service-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 80.

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.
Development and deployment

The tool provides a graphical interface for effective configuration of the SIL API, providing methods for creating, modifying and deleting devices, platforms, clients and alignments, making use of a graphical interface.

The tool provides both, a REST API and a graphical interface. The REST API can be consumed directly by a developer for the creation and modification of devices, services, platforms, clients and alignments or, can be consumed, in a more user-friendly way, through the graphical interface.

The tool has been developed using the Java programming language, relying on technologies such as JAX-RS, Apache Tomcat and Maven. Finally the application has been packaged using Docker and is available in the project repository.

– Graphical interface functionalities

The main objective of this section is to present as an image the main functionalities of the tool.

![Component Configuration. Creation of a new platform.](image1)

![Component Configuration. View platform-specific information.](image2)

Figure 81: Component Configuration. Creation of a new platform.

Figure 82: Component Configuration. View platform-specific information.
Figure 83: Component Configuration. Edit platform-specific information.

Figure 84: Component Configuration. Creation of a new device.

Figure 85: Component Configuration. View device-specific information.
SSIL extension

Similar to the Deployment Manager tool, the document provides equivalent functionality for devices and services. The limitation for the development of the functionality is again the interoperability layer, which provides interoperability at the data level but not at the service level.

In this way, an extension of the application functionality is considered only if the implementation of the interoperability layer allows the creation, modification and elimination of services in the interoperability layer. Thus, the functionality of the application will be extended by providing methods for the correct configuration of services in the system.

| Source code | https://git.activageproject.eu/Deployment/DM-Component_configuration |
| Docker registry | https://docker-registry-activage.satrd.es/repo/tags/component-configuration |
| Course      | Lessons > 2. AIoTES Framework > 2.5. Deployment tools > 2.5.3 Component configuration |
| License     | The tool is distributed under Apache v2.0 license. The tool depends on: Angular JS MIT license, npm modules licenses and maven artifact licenses. In any case one of this licenses are more restrictive than the one under which the tool is distributed. |
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. The maintenance panel communicates with the Deployment Manager, as depicted in Figure 87, in order to retrieve all deployment-specific information about the devices and services installed, from which the deployer can select the installation components for which to open the maintenance panel.

![Diagram of maintenance panel deployment tool](image)

Figure 87: 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.

Development and deployment

The tool provides an easy way to store issues related to a particular deployment installation or device. These requests are stored by the maintenance panel and can be modified or completed using the graphical interface. As described above the application makes use of the deployment manager tool to retrieve information from the deployment installation and devices in the system.

The tool has been developed using the Java programming language, relying on technologies such as JAX-RS, Apache Tomcat and Maven. Finally the application has been packaged using Docker and is available in the project repository.

Graphical interface functionalities

The main objective of this section is to present as an image the main functionalities of the tool.
SSIL extension

The current version of the tool allows the creation and storage of alerts or notifications about the status of Deployment Installations and Devices. The application extension would first allow the creation and modification of these alerts for services.

In addition, if the interoperability layer finally has a metrics reporting mechanism that includes real-time status information on the different entities in a deployment, the functionality will be extended to provide real-time access to these notifications.

Source code  https://git.activageproject.eu/Deployment/DM-Maintenance_panel


Docker registry  https://docker-registry-activage.satrd.es/repo/tags/maintenance-panel

Course  Lessons > 2. AloTES Framework > 2.5. Deployment tools > 2.5.6 Maintenance panel
5.1.2.4 Update manager

The update manager deployment tool provides a graphical user interface which facilitates the deployer in discovering 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.

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 an interface that provides a clear description of the process of deployment and configuration of the various services in the system.

Development and deployment

The tool provides an easy way to deployment information and guides for each service. These requests are stored in a relational database and can be modified or completed using the graphical interface. The application makes use of the activage docker registry tool to retrieve information from the deployment installation and services in the system.

The tool has been developed using the Java programming language, relying on technologies such as JAX-RS, Apache Tomcat and Maven. Finally the application has been packaged using Docker and is available in the project repository.

Graphical interface functionalities

The main objective of this section is to present as an image the main functionalities of the tool.
SSIL extension

An extension of the application functionality will be presented in case the interoperability layer extension allows the retrieval of information related to the version of the services and devices deployed in a Deployment installation.

In this case, the update manager will communicate with Semantic Interoperability Layer, as depicted in Figure 90, 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.
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. Figure 93 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 93: Mapping between requirements (orange) to the ACTIVAGE deployment tools (green).
6 Conclusion / Future Work

This deliverable D4.4 is the updated version of the deliverable D4.1, which reports on the progress and work done in Task 4.1 and Task 4.3, aiming at providing a set of AIOTES development and deployment tools that can facilitate the adoption of the ACTIVAGE approach for AHA IoT solutions.

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. 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 ACTIVAGE 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 [2], technical experts from deployment sites…), 23 requirements related to ACTIVAGE 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 ACTIVAGE deployment tools have been organised in two main categories, such as IoT infrastructure management, both devices and services, and distribution and deployment.

The analysis of these requirements took 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.

One of the cornerstone of ACTIVAGE project is the reusability, from all possible aspects. We have done a deep analysis for the tools that are already provided by the IoT platforms integrated in ACTIVAGE and could form the basis for the set of development and deployment tools. The result of this analysis is given as appendix in Section 7, which provides an accurate landscape of available tools.

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 AIOTES development tools and 8 deployment tools have been specified, describing their goal, main functionalities and intended usage. The proposed tools address all identified requirements.

The next step regarding the AIOTES development and deployment tools is to make them easily accessible and used by the deployment sites and open callers. The focus in the coming months will be to adopt those tools, evaluate and improve them if necessary until the end of the project.
7 References


[4] WWW.FIWARE.ORG


[8] HTTPS://GITHUB.COM/OPENIOTORG/OPENIOT


[10] HTTP://SOFIA2.COM


[16] UNIVERSAAL IoT, SEMANTIC INTEROPERABILITY FOR RAPID INTEGRATION & DEPLOYMENT: HTTPS://GITHUB.COM/UNIVERSAAL


[22] SENSI NACT WEB SITE: HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY.SENSINACT

[23] HTTP://EDU.FIWARE.ORG/


[26] HTTP://SOFIA2.COM/DQCS/(EN)%20SOFIA2-HOW%20TO%20DEVELOP%20ON%20THE%20SOFIA%20PLATFORM.PDF

[27] HTTP://SOFIA2.COM/DQCS/SOFA2-APIS%20SOFIA2.PDF

[28] HTTP://SOFIA2.COM/SIB/

[29] HTTPS://WWW.OPENLDAP.ORG/


[31] FIWARE WEB SITE: HTTPS://WWW.FIWARE.ORG/


[33] SENIORSOME WEB SITE: HTTP://WWW.SENIORSOME.COM

[34] ONE2M2M WEB SITE: HTTP://WWW.ONE2M2M.ORG

[35] ECLIPSE TECHNOLOGY: HTTPS://PROJECTS.ECLIPSE.ORG/PROJECTS/TECHNOLOGY


[37] D7.1 INITIAL ECOSYSTEM MANAGEMENT PLAN: ACTIVAGE > 3 ACTIVAGE PROJECT > 014 PROJECT DELIVERABLES > DELIVERABLES MONTH 15, D.7.1 INITIAL ECOSYSTEM MANAGEMENT PLAN
Appendix A  Available development tools supported by the ACTIVAGE IoT platforms

A.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).

A.1.1 Support Tools

Table 2: universAAL support tools

<table>
<thead>
<tr>
<th>Documentation</th>
<th><a href="https://github.com/universAAL/platform/wiki">https://github.com/universAAL/platform/wiki</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>API Doc</td>
<td><a href="http://universaal.github.io/platform/apidocs/index.html">http://universaal.github.io/platform/apidocs/index.html</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://github.com/universAAL/remote/wiki/REST-API">https://github.com/universAAL/remote/wiki/REST-API</a></td>
</tr>
<tr>
<td>Source Code</td>
<td><a href="https://github.com/universAAL">https://github.com/universAAL</a></td>
</tr>
<tr>
<td>Tutorial and Sample code</td>
<td><a href="https://github.com/universAAL/platform/wiki#core-tutorials">https://github.com/universAAL/platform/wiki#core-tutorials</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://github.com/universAAL/platform/wiki/Hello-World">https://github.com/universAAL/platform/wiki/Hello-World</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://github.com/universAAL/samples">https://github.com/universAAL/samples</a></td>
</tr>
</tbody>
</table>

A.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 universAAL wiki (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) that contains video tutorials related to universAAL core topics and the creation of universAAL-based applications.

A.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>
<thead>
<tr>
<th>URL</th>
<th>METHOD</th>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>{url}/uaal</td>
<td>GET</td>
<td></td>
<td>{url}/uaal/spaces</td>
</tr>
<tr>
<td>{url}/uaal/spaces</td>
<td>GET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>Space (json/xml)</td>
<td>Space (json/xml list)</td>
<td></td>
</tr>
<tr>
<td>{url}/uaal/spaces/{myspace}</td>
<td>GET</td>
<td>{url}/uaal/spaces/{myspace}/context</td>
<td>{url}/uaal/spaces/{myspace}/service</td>
</tr>
<tr>
<td>PUT</td>
<td>Space (json/xml)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{url}/uaal/spaces/{myspace}/context</td>
<td>GET</td>
<td>{url}/uaal/spaces/{myspace}/context/publishers</td>
<td>{url}/uaal/spaces/{myspace}/context/subscribers</td>
</tr>
<tr>
<td>{url}/uaal/spaces/{myspace}/context/publishers</td>
<td>GET</td>
<td>Publisher (json/xml list)</td>
<td></td>
</tr>
</tbody>
</table>
Examples of body requests and usage of REST-API are also provided in the wiki page of universAAL.
A.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).

A.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) 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.

A.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.
Figure 94 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 are useful to create source code, which is then compiled using Maven), this allows developers to use IDEs other than the one proposed. With in the build process, the developer can choose to create tests using testing frameworks, which have been extended to make it easier to develop said tests. With in the runtime environment there are 2 kinds of tools, proper tools which are used to test, monitor and produce data of existing applications; and modules of the platform itself which can be extended through specific files to create or extend applications.

A.1.2.1 AAL Studio

The AAL Studio is a suite that 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.


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.

A.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
A.1.2.1.1 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.
Figure 97 Basic information provided for the Project wizard

Figure 98 Customization of the project dependencies and components in the project wizard.
A.1.2.1.2 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.

![Ontology project properties](image)

**Figure 99 Basic information for the Ontology project wizard.**

A.1.2.1.3 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)

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

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.

A1.2.1.1.4 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 will be capable of not only generating universAAL code, but by switching the template system, generate code for other platforms, frameworks, or programming languages.

A1.2.1.2 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.

A1.2.1.2.1 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.
A.1.2.2 Build

These tools are used to setup each building environment for proper testing by particular testing framework; 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.

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 AITOES. Yet the concept of a tool that automatically generates the required runtime configuration given a module might be interesting in AIOTES.

A.1.2.3 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.

A.1.2 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 AIOTES tool set.

A.1.2.3 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.

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 AITOES. 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.
A.1.2.3.1 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.

A.1.2.3.2 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.

A.1.2.3.3 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.

A.1.2.4 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\(^1\), and the Service Orchestrator\(^2\). 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).

\(^1\) [https://github.com/universAAL/context/wiki](https://github.com/universAAL/context/wiki)
\(^2\) [https://github.com/universAAL/service/wiki/Service-Orchestration](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.

**A.1.2.5 Planned Tools**

**A.1.2.5.1 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>13</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 102 a code-generation Java generation example](image)

<sup>13</sup> [https://owlcs.github.io/owlapi/](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.

A.1.2.5.2 universAAL Control Center 2.0

In an effort to make running tools more coherent a similar approach to the AAL Studio 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.

A.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>
<thead>
<tr>
<th>universAAL development tool</th>
<th>Corresponding ACTIVAGE development tool(s)</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support tools (documentation, wiki, source code samples)</td>
<td>Support</td>
<td>Yes</td>
</tr>
<tr>
<td>How?</td>
<td>Why?</td>
<td></td>
</tr>
<tr>
<td>The universAAL support tools (documentation, wiki, code samples, etc.) will be used as part of the overall AIOTES support material.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project and Item wizard</td>
<td>IDE / Source code templates</td>
<td>Yes</td>
</tr>
<tr>
<td>How?</td>
<td>Why?</td>
<td></td>
</tr>
<tr>
<td>These tools are too universAAL specific (they relate to universAAL projects and modules). A Similar wizard could be useful for AIOTES development.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ontology project wizard, modelling tool, and transformation.</td>
<td>IDE / Code Generator</td>
<td>Yes</td>
</tr>
<tr>
<td>How?</td>
<td>Why?</td>
<td></td>
</tr>
</tbody>
</table>
| These tools are too universAAL specific (they relate to universAAL ontology development). This tool would only make sense if AIOTES requires a
<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable 4.4 – Developers toolkit and deployment support</td>
<td>This deliverable focuses on developing a toolkit and deployment support for ACTIVAGE, allowing developers to design and export code using an ontology. A generic tool is proposed to facilitate the design of tools and code export.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Monitor</td>
<td>Not mapped with AIOTES</td>
</tr>
<tr>
<td>Makro Recorder</td>
<td>Not mapped with AIOTES</td>
</tr>
<tr>
<td>Sparql Tester</td>
<td>Data Lake / ACTIVAGE data model Workbench</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Compliance and Code Quality</td>
<td>Not mapped with AIOTES</td>
</tr>
<tr>
<td>Maven build extensions</td>
<td>Not Mapped with AIOTES</td>
</tr>
<tr>
<td>Maven Achetypes</td>
<td>IDE / Source code templates</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td>These tools can be used to generate universAAL projects, but also Karaf features, specially indicated for OSGi based deployments.</td>
</tr>
<tr>
<td>Testing frameworks</td>
<td>Not Mapped with AIOTES</td>
</tr>
<tr>
<td>Reasoner and orchestrator</td>
<td>IDE / Service Composer</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td>The universAAL Modules can be used through the SIL to execute their composition operations.</td>
</tr>
<tr>
<td>Template Based Ontology Code Generator</td>
<td>Semantic Interoperability tools; IDE / Code Generator</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td>The tool will have the capability of defining its own templates, specific templates for platform specific adaptations (given an arbitrary ontology) can be 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 adapting applications to AIOTES Ontologies.</td>
</tr>
<tr>
<td>universAAL Control Center</td>
<td>Semantic Interoperability Layer tools / *;</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td>uCC is too universAAL Specific. The concept of universAAL Control Center is too specific for AIOTES.</td>
</tr>
</tbody>
</table>

Version 1.0 I 2019-Aug-05 ACTIVAGE ©
### A.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.

#### A.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/](https://about.sofia2.com/)

#### A.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 event-based 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.

### A.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](https://github.com/Sofia2)

### A.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/UC6VGVIIN9gB2mJcPdYHBOA](https://www.youtube.com/channel/UC6VGVIIN9gB2mJcPdYHBOA).

In addition, SOFIA2 has a tutorial of first steps in SOFIA2: [https://www.iorad.com/player/19843/Primeros-Pasos-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 5: SOFIA2 existing samples and source code

<table>
<thead>
<tr>
<th>Example</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic viewer</td>
<td>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></td>
</tr>
<tr>
<td>Dashboard Smart Health</td>
<td>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></td>
</tr>
<tr>
<td>Geographical search Twitter</td>
<td>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></td>
</tr>
<tr>
<td>Demo Twitter Streaming</td>
<td>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></td>
</tr>
</tbody>
</table>

A.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 Scrips 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 on the 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
It allows 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.

### A.2.6 Mapping between SOFIA2 and ACTIVAGE development tools

Part of the SOFIA2 development tools can be used within the AIOTES infrastructure as ACTIVAGE development tools. This mapping is presented in Table 6 where it is specified if a tool can be generalized to be used within AIOTES and how.

<table>
<thead>
<tr>
<th>SOFIA2 development tool</th>
<th>Corresponding ACTIVAGE development tool(s).</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ontologies management tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>KPs/APPs management tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>Token management tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>Token management tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>Rules management tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>Visualization tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>Predefined Queries Management</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>Send SSAP Messages tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td>API Manager tool</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
</tr>
<tr>
<td><strong>Support tools</strong></td>
<td><strong>Support</strong></td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>(documentation, wiki, source code samples)</td>
<td></td>
<td>Why?</td>
</tr>
</tbody>
</table>

The SOFIA2 support tools (documentation, wiki, code samples, etc.)
A.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[14] space under the Documentation[15] section.

A.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.

A.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 7.

Table 7: List of primitives comprising the OpenIoT SD&UM implemented API

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Input</th>
<th>Output</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>pollForReport (applicationID: String): SdumServiceResultSet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getApplication (applicationID: String): OAMO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getService (serviceID: String): OSMO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getAvailableAppIDs (userID: String): DescriptiveIDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getAvailableServiceIDs (applicationID: String): DescriptiveIDs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getUser (userID: String): OpenIotUser</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The services description as long as their inputs and outputs are listed in Table 8.

Table 8: Service Delivery & Utility Manager implemented API definition

14 https://github.com/OpenIotOrg/openiot/wiki
15 https://github.com/OpenIotOrg/openiot/wiki/Documentation
A.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

A.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.

A.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 9 below illustrates the main API primitives that provide the LSM-Light functionalities, while Table 10 provides more details about all services that comprise the API.

Table 9: List of primitives comprising the OpenIoT LSM-Light API

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getSensorById</td>
<td>String sensorID</td>
<td>Sensor</td>
<td>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.</td>
</tr>
<tr>
<td>getSensorBySource</td>
<td>String sensorSource</td>
<td>Sensor</td>
<td>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.</td>
</tr>
<tr>
<td>sensorAdd</td>
<td>Sensor sensor</td>
<td>void</td>
<td>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.</td>
</tr>
<tr>
<td>sensorDataUpdate</td>
<td>Observation observation</td>
<td>void</td>
<td>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.</td>
</tr>
<tr>
<td>deleteTriples</td>
<td>String graphURL</td>
<td>void</td>
<td>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 deleted or not.</td>
</tr>
</tbody>
</table>

Table 10: LSM-light API Specification

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Input</th>
<th>Output Info</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getSensorById</td>
<td>String sensorID</td>
<td>Sensor</td>
<td>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.</td>
</tr>
<tr>
<td>getSensorBySource</td>
<td>String sensorSource</td>
<td>Sensor</td>
<td>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.</td>
</tr>
<tr>
<td>sensorAdd</td>
<td>Sensor sensor</td>
<td>void</td>
<td>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.</td>
</tr>
<tr>
<td>sensorDataUpdate</td>
<td>Observation observation</td>
<td>void</td>
<td>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.</td>
</tr>
<tr>
<td>deleteTriples</td>
<td>String graphURL</td>
<td>void</td>
<td>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 deleted or not.</td>
</tr>
</tbody>
</table>
A.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.

A.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 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 103 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.
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
A.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 104).

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.

A.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/).

A.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/utils.commons/ 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.commons 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

---

16 https://github.com/OpenIoTorg/openiot/tree/master/utils/utils.commons
A.3.5 Source Code & Binaries

A.3.5.1 Source Code Availability

The OpenIoT repository is hosted at the GitHub\(^\text{17}\) and can be found at the following link: [https://github.com/OpenIoTOrg/openiot](https://github.com/OpenIoTOrg/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

A.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
  - Ide.core

\(^\text{17}\) [http://github.com/](http://github.com/)
A.3.5.3 Binaries Availability

OpenIoT binaries are available through the OpenIoT Wiki site\(^{18}\) under the Users>Downloads\(^{19}\) 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.

A.3.6 Documentation

The OpenIoT Wiki is publicly available on Github\(^{20}\) which provides access to all support materials necessary for developers and users to start working on and/or using the platform.

A.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 11. Some of the mapped tools may be used within the AIOTES infrastructure, possibly with some modifications or generalizations. Table 11 also summarizes which tools can be generalized to be used within AIOTES, or are too specific to be included.

Table 11: Mapping between OpenIoT and ACTIVAGE development tools.

<table>
<thead>
<tr>
<th>OpenIoT development tool</th>
<th>Corresponding ACTIVAGE development tool(s).</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Delivery &amp; Utility</td>
<td>Not mapped with AIOTES</td>
<td></td>
</tr>
<tr>
<td>Linked Stream Middleware Light</td>
<td>Not mapped with AIOTES</td>
<td></td>
</tr>
<tr>
<td>User Interfaces / Request Definition</td>
<td>Data/Visual Analytics Tools / Data Analyser</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How?</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Definition module provides WYSIWYG UI to create service (similar to NodeRED) which can be an analytics service in case of ACTIVAGE.</td>
<td></td>
</tr>
</tbody>
</table>

---

\(^{18}\) [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](https://github.com/OpenIoTOrg/openiot/wiki)
## Deliverable 4.4 — Developers toolkit and deployment support

### User Interfaces / Request Representation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Representation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>How?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Why?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Representation is similar to Feature/Result viewer.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### User Interfaces / Schema Editor

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic Interoperability Layer tools / Device Semantics Editor</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>How?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Why?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schema Editor only supports annotating sensors using the OpenIoT ontology. For ACTIVAGE ontology, it needs major changes to work.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Utilities & Libraries

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not mapped with AIOTES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Documentation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>How?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Why?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The OpenIoT Documentation can be used as part of the overall AIOTES support material.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Source Code & Binaries

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE / Code generator IDE / Source code templates</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>How?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Why?</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## A.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.
A.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.

A.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 105. 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-butter-smart-gateway/](http://open-platforms.eu/library/sensinact-aka-butter-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-butter-smart-gateway/](http://open-platforms.eu/library/sensinact-aka-butter-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 105 the Service and Resource model.

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 Table 12, while the access methods are described in Table 13.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSORDATA</td>
<td>Sensory data provided by a service. This is real-time information provided, for example, by the SmartObject that measures physical quantities.</td>
</tr>
<tr>
<td>ACTION</td>
<td>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.)</td>
</tr>
</tbody>
</table>
**STATEVARIABLE**

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.

**PROPERTY**

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

**Table 13: sensiNact resource’s access method.**

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.

**A.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](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](https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.gateway)


**sensinact/org.eclipse.sensinact.studio**

Clone: [https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.studio](https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.studio)


**sensinact/org.eclipse.sensinact.studioweb**

Clone: [https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.studioweb](https://git.eclipse.org/r/sensinact/org.eclipse.sensinact.studioweb)

A.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

A.4.5 sensiNact ACTIVAGE development tools

A.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 now implemented, based on the IoT system 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 and updated in the D3.11 deliverable.

The functional analysis provides also the service layer specifications that describes the mandatory AHA functions provided in the service layer in order to fulfil the expected features. This service layer specification specifies the 20 AHA functions for DS6.

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

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

- AHA-building function provider (container for studios and homes)
AHA-studio / AHA-home function provider (description of living place in institution / at home)

**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-alert
- AHA-bedroom-inactivity-alert
- AHA-bathroom-inactivity-alert
- AHA-shower-alert
- AHA-pain-alert (in institution, rises daily notification to ask the patient to autoevaluate pain and rise alert to medical staff if necessary)
- AHA-washing-autonomy (when necessary, rises notification message to the patient in institution not to forget to wash in the morning)
- AHA-domestic-alert (triggers alert in case of abnormal use of device – too long fridge door open, too long oven on…)
- AHA-air-quality (triggers alert in case of abnormal measure of air quality factor such as CO and CO2 rate, relative humidity, smoke, gas…)
- AHA-opening-alert (triggers alert when opening remains open at night)

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

- AHA-night-rising-monitor (bed inoccupancy during night)
- AHA-day-laying-monitor (bed occupancy during daytime)
- AHA-podometer (walked distances)
- AHA-weight-monitoring (weight – not yet available)
- AHA-water-consumption
- AHA-electric-consumption (current power and historized energy)
- AHA-kpi (technical key performance indicators)

**Automation function API** provides service to automatically switch on the light in a bedroom at night.

- AHA-automatic-lighting-control is the dedicated function to configure the automatic lighting control

These 20 AHA functions are implemented on the top of the generic sensiNact API, using the provider/service/resource pattern previously described. AHA functions are specialization of providers, with dedicated services and resources.

The table below describes the 20 specified AHA function for DS6:

**Table 14: The 20 sensiNact AHA functions.**

<table>
<thead>
<tr>
<th>Category</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>

Version 1.0 | 2019-Aug-05 ACTIVAGE ©
<table>
<thead>
<tr>
<th><strong>Building configuration</strong></th>
<th><strong>AHA-building</strong></th>
<th>Provides service to configure the overall building gathering several studios</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHA-studio</td>
<td>Provides service to configure the elementary living place in institution considered for AHA functions. This elementary living place is called studio and is constituted of a bedroom and a bathroom</td>
<td></td>
</tr>
<tr>
<td>AHA-home</td>
<td>Provides service to configure the elementary living place at home, with several rooms (not only a bedroom and a bathroom)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Notifications and alerts</strong></th>
<th><strong>AHA-temperature-alert</strong></th>
<th>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 acquittance by the patient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHA-bedroom-inactivity-alert</td>
<td>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 acquittance by the patient)</td>
<td></td>
</tr>
<tr>
<td>AHA-bathroom-inactivity-alert</td>
<td>Provides service to trigger alert to carers when long inactivity is detected in a given bathroom</td>
<td></td>
</tr>
<tr>
<td>AHA-shower-alert</td>
<td>Provides service to trigger alert to carers when long shower duration is measured in a given bathroom</td>
<td></td>
</tr>
<tr>
<td>AHA-pain-alert</td>
<td>In institution, provides service to rise daily notification to ask the patient to autoevaluate pain and rise alert to medical staff if necessary</td>
<td></td>
</tr>
<tr>
<td>AHA-washing-autonomy</td>
<td>In institution, provides service to rise notification message when necessary to the patient not to forget and ask for help to wash in the morning</td>
<td></td>
</tr>
<tr>
<td>AHA-domestic-alert</td>
<td>At home, provides service to trigger alert in case of abnormal use of device – too long fridge door open, too long oven on…</td>
<td></td>
</tr>
<tr>
<td>AHA-opening-alert</td>
<td>At home, provides service to trigger alert when opening remains open at night</td>
<td></td>
</tr>
<tr>
<td>AHA-air-quality</td>
<td>At home, provides service to trigger alert in case of abnormal measure of air quality factor such as CO and CO2 rate, relative humidity, smoke, gas…</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Activity monitoring</strong></th>
<th><strong>AHA-night-rising-monitor</strong></th>
<th>Provides service to get “in live” and past occurrences of risings during night in a given bedroom.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHA-day-laying-monitor</td>
<td>Provides service to get “in live” and past occurrences of bed presences during daytime in a given bedroom.</td>
<td></td>
</tr>
<tr>
<td>AHA-podometer</td>
<td>Provides service to get “in live” and past occurrences of walked distances (weight – not yet available)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(current power and historized energy)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(technical key performance indicators)</td>
<td></td>
</tr>
</tbody>
</table>
## AHA-weight-monitoring
Provides service to get "in live" and past occurrences of weights (not yet available – developments in progress)

## AHA-water-consumption
Provides service to get "in live" and past occurrences of water consumption.

## AHA-electric-consumption
Provides service to get "in live" power and past energies.

## AHA-kpi
Provides service to get "in live", target, and past occurrences of technical kpi for the whole DS, including number of devices, number of beneficiaries. Implementation is in progress to provide the complete list of technical KPIs.

## AHA-automatic-lighting-control
Provides service to configure the automatic lighting control in a given room

The link to download and view the complete documentation for these 20 functions is:

ACTIVAGE > 3 ACTIVAGE PROJECT > 010 Work Packages > 004 WP4 - CERTH - App Su... > T4.1 > doc-API-sensiNact-ACTIVAGE.zip

### A.4.5.2 sensiNact ACTIVAGE API doc and swagger

#### A.4.5.2.1 Documentation of the sensiNact AHA service API

The sensiNact AHA service API gives access to ten specified functions (see Figure 106).

![Figure 106: the sensiNact AHA functions (implemented and in progress)](image)

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.
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 in Figure 108 describes the available AHA services specified for AHA function providers. 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 contains 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/carer). 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.
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 > doc-API-sensiNact-ACTIVAGE.zip

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.

### A.4.5.2.2 Swagger of the sensiNact AHA service API

A swagger documentation of the sensiNact AHA service API is available on line:
The swagger web application provides the list of available requests as illustrated in Figure 109.

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 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.
A.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)

<table>
<thead>
<tr>
<th>Statistics aggregation</th>
<th>Sampling period</th>
</tr>
</thead>
<tbody>
<tr>
<td>dayLayingCounter</td>
<td>day</td>
</tr>
<tr>
<td>dayLayingDuration</td>
<td>day</td>
</tr>
<tr>
<td>nightRisingCounter</td>
<td>day</td>
</tr>
<tr>
<td>nightRisingDuration</td>
<td>day</td>
</tr>
<tr>
<td>tapOnDuration</td>
<td>day</td>
</tr>
<tr>
<td>showerOnDuration</td>
<td>day</td>
</tr>
<tr>
<td>showerAlertCounter</td>
<td>day</td>
</tr>
<tr>
<td>washingNotificationCounter</td>
<td>day</td>
</tr>
<tr>
<td>temperatureAlertCounter</td>
<td>day</td>
</tr>
<tr>
<td>stepCounter</td>
<td>day and hour</td>
</tr>
<tr>
<td>painAlertCounter</td>
<td>day</td>
</tr>
<tr>
<td>averagePainLevel</td>
<td>day and hour</td>
</tr>
<tr>
<td>falseAlertCounter</td>
<td>day</td>
</tr>
<tr>
<td>alertCounter</td>
<td>day</td>
</tr>
<tr>
<td>notificationAcceptmentCounter</td>
<td>day</td>
</tr>
</tbody>
</table>

### A.4.5.4 sensiNact public access to ACTIVAGE dedicated source code

The source code is not available yet in a public git repository. After verifying the license issues some part of the code may be available in the official Eclipse repository of sensiNact.

### A.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:

```plaintext
RoomDictionary is Map<String, String> with key: roomId, value: buildingId студия_0[bathroom|bedroom]

1. init
2. roomMap: Map<String, String> 3. studiaProviders: List<String>
4. for each studid in studiaProviders
6. bathroom = bathroomresponse.value
8. bathroomId = bathroomId
12. building = buildingresponse.value
14. roomMap.put(bathroomId, studidId + */studid + */bathroomId)
15. roomMap.put(bathroomId, studidId + */bedroomId)
```

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:

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

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

private synchronized void buildRoomMap() throws IOException {
    Map<String, SNaRoom> roomMap = new HashMap<>();
    List<String> studioProviderIds = requestGet("aha/sensinact/getStudioProviderIds", SNaResponse.ProviderIdsResponse.class);
    for(String studioProviderId : studioProviderIds) {
        SNaResponse.StudioProviderIdsResponse studioProviderIdsResponse = requestGet("sensinact/providers/"+studioProviderId + "/services/studio/resources/id/GET", SNaResponse.StudioProviderIdsResponse.class);
        String studioName = studioProviderIdsResponse.studioName;
        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(buildingId, 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(bathroomId, 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 code illustrates how to build the alert dictionary and use it to handle alert triggering and acquitment.

A.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

<table>
<thead>
<tr>
<th>documentation</th>
<th>API doc and swagger</th>
<th>Public source code</th>
<th>Tutorial</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensiNact</td>
<td>sensiNact wiki</td>
<td>Swagger (real data) for live data sensinact.ddns.net/swagger-api/ and historic data</td>
<td>Eclipse sensiNact Git repository</td>
<td>sensiNact tutorial section in wiki</td>
</tr>
</tbody>
</table>
AHA sensiNact service API

- sensiNact AHA service API shared documentation
- 193.48.18.245:8081/swagger-api/ (mock instance)
- To be published

| services API                      | 193.48.18.251:8080/swagger-ui.html | A public sensiNact instance running with mocked AHA functions provides dynamic random values for test purposes |

A.4.7 Mapping between sensiNact and ACTIVAGE Development Tools

Part of the sensiNact development tools described above correspond to specific ACTIVAGE development tools. Some of the mapped tools may be used within the AIOTES infrastructure, possibly with some modifications or generalizations. Table 17 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

<table>
<thead>
<tr>
<th>sensiNact development tool</th>
<th>Corresponding ACTIVAGE development tool(s.)</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensiNact AHA historical statistics agent</td>
<td>Data/Visual Analytics Tools / Data Analyser</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**How?**
- Embedded in the sensiNact ACTIVAGE distribution, computation results are accessible through the sensiNact AHA API and through the AIOTES API

**Why?**
- 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 |

**How?**
- 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.

**Why?**
- 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 |

**How?**
- The sensiNact support tools (documentation, wiki, swaggers, code samples, etc.) will be used as part of the overall AIOTES support material.

**Why?**
- The sensiNact support tools (documentation, wiki, swaggers, code samples, etc.) will be used as part of the overall AIOTES support material.
A.5 FIWARE

FIWARE\(^{21}\) is an open middleware platform, driven by the European Union under the Future Internet Public Private Partnership Programme\(^{22}\), for the development and global 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.

A.5.1 Supports Tools

A.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\(^{23}\) 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\(^{24}\) focused on the most common instructions to make 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\(^{25}\) 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**

\(^{21}\) https://www.fiware.org/

\(^{22}\) https://www.fi-ppp.eu/.

\(^{23}\) https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Welcome_to_the_FIWARE_Wiki

\(^{24}\) https://www.fiware.org/developers-entrepreneurs/

\(^{25}\) https://catalogue.fiware.org/
This category brings information related to FIWARE documentation including:

- 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.
- 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\(^{26}\) 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 complementarywiki 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.

**A.5.12 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

\(^{26}\) 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/](https://forge.fiware.org/) under

The common aspects in FI-WARE Open Restful API Specifications can be found under

Additionally, each GE can be accessed through its own API (normally REST), for example

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

**A.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](https://github.com/Fiware)

**A.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 guides27 dedicated to the gathering all the getting starting guides of the FIWARE
components. Table 18 lists all the guides along with a short description.

---

27 [https://www.fiware.org/developers-entrepreneurs/](https://www.fiware.org/developers-entrepreneurs/)
Table 18: FIWARE existing tour guides

<table>
<thead>
<tr>
<th>TOUR GUIDE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development of Context-Aware Applications using FIWARE</td>
<td>Orion Context Broker allows you to model, manage and gather context information at large scale enabling context-aware applications.</td>
</tr>
<tr>
<td>Real time processing of Context Events</td>
<td>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.</td>
</tr>
<tr>
<td>Publication of Context Information as Open Data</td>
<td>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.</td>
</tr>
<tr>
<td>Creating Application Dashboards</td>
<td>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.</td>
</tr>
<tr>
<td>Providing an Advanced User Experience (UX)</td>
<td>These components allow you to incorporate advanced features in your web-based user interface, such as Augmented Reality or 3D visualization.</td>
</tr>
<tr>
<td>Connection to the Internet of Things</td>
<td>IDAS IoT Agents allow your application to easily gather context information from sensors or actuate upon physical objects.</td>
</tr>
<tr>
<td>Handling Authorization &amp; Access Control to APIs</td>
<td>FIWARE brings a powerful framework that will allow you to setup Authorization and Access Control policies based on widely adopted Security standards (OAuth, XACML).</td>
</tr>
<tr>
<td>Big Data Analysis of Historic Context Information</td>
<td>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.</td>
</tr>
<tr>
<td>Real time processing of Media Streams</td>
<td>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,…).</td>
</tr>
<tr>
<td>Hosting your Application on a FIWARE Cloud</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

A.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 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.

28 http://edu.fiware.org/
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 catalogue²⁹):

- 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. 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 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.

A.5.3 Mapping between FIWARE and ACTIVAGE development tools

Part of the FIWARE development tools, can be used within the AIOTES infrastructure as ACTIVAGE development tools. 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.

<table>
<thead>
<tr>
<th>FIWARE development tool</th>
<th>Corresponding ACTIVAGE development tool(s)</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Time Historic</td>
<td>Not mapped with AIOTES</td>
<td></td>
</tr>
<tr>
<td>Connector Framework (Cygnus)</td>
<td>Not mapped with AIOTES</td>
<td></td>
</tr>
<tr>
<td>Complex Event Processing</td>
<td>Not mapped with AIOTES</td>
<td></td>
</tr>
<tr>
<td>Support tools (documentation, wiki, source code samples)</td>
<td>Support</td>
<td>Yes</td>
</tr>
<tr>
<td>How?</td>
<td>Why?</td>
<td></td>
</tr>
<tr>
<td>The FIWARE support tools (documentation, wiki, code samples, etc.) will be used as part of the overall AIOTES support material.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source code samples</td>
<td>IDE / Code generator IDE / Source code templates</td>
<td>Yes</td>
</tr>
<tr>
<td>How?</td>
<td>Why?</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A.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.
A.6.1 Support tools

A.6.1.1 Documentation

IoTivity offers a [wiki](https://api-docs.iotivity.org/latest) that provides detailed documentation and tutorials 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.

A.6.1.2 APIs

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>
A.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.

- **roomservicer**: 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.

- **simpleservers**: 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.

- **threading** sample**: 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

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/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.

A.6.1.4 Tutorials

IoTivity Tutorials are included in the IoTivity wiki <https://wiki.iotivity.org/iotivity_simulator>, covering all aspects mentioned above. IoTivity tutorials also contain sample source code, as presented previously.

A.6.2 Tools useful in the ACTIVAGE context

A.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.

A.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.
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. 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/>
- MongoDB <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/>

### A.6.2.3 OneIoTa 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.

---

30 [https://hub.docker.com/u/iotivity/](https://hub.docker.com/u/iotivity/)
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.

Creating data models (resource models) for OCF IoT Devices is done by oneIoTa\textsuperscript{31} 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 oneIoTa repository.

![oneIoTa data models development process.](image)

**A.6.2.4 Provisioning Manager**

Provisioning Manager (PM)\textsuperscript{32} 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).

\textsuperscript{31} https://openconnectivity.org/developer/oneiota-data-model-tool

\textsuperscript{32} https://wiki.iotivity.org/provisioning
A.6.2.5 IoTivity simulator

IoTivity provides a tool <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 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 113: Service Provider Perspective of IoTivity Simulator](image-url)
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.

### A.6.3 Mapping between IoTivity and ACTIVAGE development tools

Part of the IoTivity development tools described above correspond to specific ACTIVAGE development tools. 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.

<table>
<thead>
<tr>
<th>IoTivity development tool</th>
<th>Corresponding ACTIVAGE development tool(s)</th>
<th>Can the tool be used in AIOTES?</th>
<th>How?</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoAP - HTTP Proxy</td>
<td>Not mapped with AIOTES</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloud connection tools</td>
<td>Data Lake tools / ACTIVAGE data model workbench</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oneloTa OCF design tool</td>
<td>Semantic Interoperability Layer tools / ACTIVAGE</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The functionalities of the IoTivity cloud connection tools will form a conceptual basis for the implementation of the cloud-based Data Lake functionalities.
A.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:

A.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.

A.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.

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

A.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.

A.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.

A.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>
<thead>
<tr>
<th>SeniorSome development tool</th>
<th>Corresponding ACTIVAGE development tool(s).</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeniorSome API:s</td>
<td>Support, Bridge, Data/Visual Analytics tools /Data Analyser, Semantic Interoperability layer</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>By using the API definition and adding this to the Activage tools and development process.</td>
</tr>
<tr>
<td>SeniorSome Backend</td>
<td>Data/Visual Analytics Tools / Feature/Result viewer / Device control, Semantic Interoperability layer</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>through the API definitions and/or Bridged connectivity. Partly can be used as a ui/interface for the AIOTES where needed.</td>
</tr>
<tr>
<td>Documentation</td>
<td>Support,</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>How?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

Table 21: Mapping between SeniorSome and ACTIVAGE development tools.
A.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.

A.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>
<thead>
<tr>
<th>Platform</th>
<th>Programming language</th>
<th>Open source</th>
<th>API documentation</th>
<th>Wiki documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Java (plain, OSGi, Android), JavaScript</td>
<td>Yes</td>
<td>Apache 2.0</td>
<td><a href="https://github.com/universAAL">https://github.com/universAAL</a></td>
</tr>
<tr>
<td>SOFIA 2</td>
<td>Java, C, JavaScript (Multilanguage)</td>
<td>Yes</td>
<td>Apache 2.0</td>
<td><a href="http://sofia2.com/desarrollador_en.html#documentacion">http://sofia2.com/desarrollador_en.html#documentacion</a></td>
</tr>
<tr>
<td>OpenIoT</td>
<td>Java</td>
<td>Yes</td>
<td>LPL V3.0</td>
<td><a href="https://github.com/OpenIoTOrg/openiot/wiki/Documentation">https://github.com/OpenIoTOrg/openiot/wiki/Documentation</a></td>
</tr>
<tr>
<td>Sensinact</td>
<td>Java</td>
<td>Yes</td>
<td>Eclipse License v1.0</td>
<td><a href="http://sensinact.ddns.net/swagger-api/index.html#/">http://sensinact.ddns.net/swagger-api/index.html#/</a></td>
</tr>
<tr>
<td>FIWARE</td>
<td>Multilanguage</td>
<td>Yes</td>
<td>Apache 2.0</td>
<td><a href="http://www.fiware.org/developers">http://www.fiware.org/developers</a></td>
</tr>
<tr>
<td>IoTvity</td>
<td>C, C++, Java</td>
<td>Yes</td>
<td>Apache 2.0</td>
<td><a href="https://www.iotivity.org/documentation">https://www.iotivity.org/documentation</a></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Provides IDE</th>
<th>IDE</th>
<th>How the IDE is provided?</th>
<th>What functionalities are offered by the IDE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Yes</td>
<td>AAL Studio</td>
<td>via plugins</td>
<td>– wizards for creating projects,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>over Eclipse</td>
<td></td>
<td>– build tools for simplifying building and launching of applications,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– modeling and transformation tools for making the development more efficient.</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>Yes</td>
<td>Eclipse</td>
<td>via plugins</td>
<td>– ontologies management,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– KPs/APPs management,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– token management,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– rules management,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– predefined queries management,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– send SAP messages,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– API manager</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>Yes</td>
<td>Eclipse</td>
<td>via plugins</td>
<td>– API for logging in and out,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– Token validity check</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– Access control utility methods</td>
</tr>
<tr>
<td>SensiNact</td>
<td>Yes</td>
<td>SensiNact</td>
<td>Uses Eclipse tools to perform continuous integration</td>
<td>Eclipse Jenkins runs a compilation of the sensiNact Gateway every day, the resulting compilation are kept in the download area of Eclipse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Studio over</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Eclipse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIWARE</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IoTivity</td>
<td>Yes</td>
<td>Eclipse</td>
<td>via plugins</td>
<td>Enables the simulation of services, OCF resources and OCF clients</td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
<td>Not available</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Samples of code</th>
<th>Dependency manager</th>
<th>Code generator</th>
<th>Build manager</th>
<th>Service Composition Builder</th>
<th>GUI Tool for management and configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Yes</td>
<td>Yes – Maven</td>
<td>Yes - AAL Studio plugins</td>
<td>Pax</td>
<td>Yes, SPARQL SPARQL Tester</td>
<td>Yes, AAL Space Monitor</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>Yes</td>
<td>Yes - Maven</td>
<td>Yes</td>
<td>Not specified</td>
<td>Yes</td>
<td>Provides visualization tools but not specified whether the requirements are matched</td>
</tr>
</tbody>
</table>
Table 25: Semantic ready platform

<table>
<thead>
<tr>
<th>Platform</th>
<th>Semantic enhanced</th>
<th>Semantic tools</th>
<th>Offered Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Yes</td>
<td>Yes</td>
<td>- 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)</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>Yes</td>
<td>Yes - Ontologies Management tool, API Management tool</td>
<td>- Creating, modifying adn 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</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>Yes</td>
<td>APIs, Request Definition GUI, Common utils and libraries</td>
<td>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.</td>
</tr>
<tr>
<td>SensiNact</td>
<td>Provider/service/resource generic datamodels</td>
<td>Provides APIs</td>
<td>Provides query methods over devices based on sensiNact data model Provides dedicated AHA service providers specialized for DS6 specified AHA functions</td>
</tr>
<tr>
<td>FIWARE</td>
<td>No</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>IoTivity</td>
<td>Yes</td>
<td>OneIoTa OCF design tool</td>
<td>Data models creation in order to reach interoperability between devices and platforms</td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
</tr>
</tbody>
</table>

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 if, they exist.
A.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, a mapping is made with the tools provided by each platform.

### Table 26: Support tools

<table>
<thead>
<tr>
<th>Platform</th>
<th>Documentation</th>
<th>Wiki</th>
<th>Tutorials</th>
<th>Code samples</th>
<th>Discussion forum</th>
<th>Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SensiNact</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>FIWARE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>IoTivity</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
<td>Not available</td>
<td>Not available</td>
<td>Not available</td>
</tr>
</tbody>
</table>

### Table 27: IDE tools

<table>
<thead>
<tr>
<th>Platform</th>
<th>Code generator</th>
<th>Code templates</th>
<th>Service composer</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>Yes - source code templates</td>
<td>Yes - source code templates</td>
<td>Yes - source code templates</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>SensiNact</td>
<td>Not specified</td>
<td>Yes – source code templates</td>
<td>Yes – sensiNact Studio</td>
</tr>
<tr>
<td>FIWARE</td>
<td>Yes - source code templates</td>
<td>Yes - source code templates</td>
<td>Yes - source code templates</td>
</tr>
<tr>
<td>IoTivity</td>
<td>Yes - IoTivity source code samples</td>
<td>Yes - IoTivity source code samples</td>
<td>Yes - IoTivity source code samples</td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
</tr>
</tbody>
</table>

### Table 28: Data/visual analytics tools

<table>
<thead>
<tr>
<th>Platform</th>
<th>Data manipulator</th>
<th>Data analyzer</th>
<th>Feature/result viewer</th>
<th>Visualization explorer</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>OpenIoT</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>SensiNact</td>
<td>Historical statistics agent</td>
<td>Historical statistics agent</td>
<td>Technosens E-lio manager web application</td>
<td>Not specified</td>
</tr>
<tr>
<td>FIWARE</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>IoTivity</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
<td>Not available</td>
</tr>
</tbody>
</table>
Table 29: Data Lake tools

<table>
<thead>
<tr>
<th>Platform</th>
<th>ACTIVAGE data model workbench</th>
<th>Metadata storage explorer</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>SensiNact</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>FIWARE</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>IoTvity</td>
<td>Yes - IoTivity cloud connection tools</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>Platform</td>
<td>ACTIVAGE data model workbench</td>
<td>Metadata storage explorer</td>
</tr>
</tbody>
</table>

Table 30: Semantic Interoperability Layer tools

<table>
<thead>
<tr>
<th>Platform</th>
<th>ACTIVAGE ontology explorer</th>
<th>Query translator</th>
<th>Device semantic editor</th>
<th>Service semantic editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>OpenIoT</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>SensiNact</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>FIWARE</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>IoTvity</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>Yes - oneIoTa OCF desing tool</td>
<td></td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
<td>Not available</td>
</tr>
</tbody>
</table>
Appendix B  Available deployment tools supported by the ACTIVAGE IoT platforms

B.1 Platform independent Available Deployment tools

B.1.1 Docker

Docker\(^{33}\) is a 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.

B.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.

\(^{33}\) https://www.docker.com/
B.1.3 OSGi based deployment

According to the OSGi Alliance, 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 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 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 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 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.

---

34 https://www.osgi.org/
35 https://www.knopflerfish.org/
36 http://felix.apache.org/
37 http://www.eclipse.org/equinox/
- **Pax Runner**
Pax Runner is a tool to provision OSGi bundles in all major open source OSGi framework implementations (Felix, Equinox, Knopflerfish, Concierge).

- **Apache Karaf**
Karaf 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.

- **Apache ACE**
Apache ACE 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.

**B.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.

**B.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 AIoTES support tool set is limited to this factor.

**B.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 the configuration files and other customizations of the karaf container for a more reliable, robust and pleasant deployment experience.

Figure 115 Run configuration of Pax Runner for Eclipse

B.2.1.2 Android universAAL Middleware App

To provide easy installation of universAAL middleware there is an universAAL middleware app which makes it easy. Then adding modules to this container is as easy as installing universAAL-android ready apps on the system.

B.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.

---

41 [https://github.com/universAAL/distro.karaf/wiki](https://github.com/universAAL/distro.karaf/wiki)
42 [https://github.com/universAAL/nativeandroid/wiki](https://github.com/universAAL/nativeandroid/wiki)
Deliverable 4.4 – Developers toolkit and deployment support

– **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.

– **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.

### B.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

– AALSspace 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.

### B.2.4 Mapping between universAAL and ACTIVAGE deployment tools

Table 31: Mapping between universAAL and ACTIVAGE deployment tools.

<table>
<thead>
<tr>
<th>universAAL development tool</th>
<th>Corresponding ACTIVAGE development tool(s).</th>
<th>Can the tool be used in AIOTES?</th>
<th>How?</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Monitor</td>
<td>Maintenance Panel / operating status notifications</td>
<td>Yes</td>
<td>No</td>
<td>This tool is too universAAL specific. Although it could be used to observe events and calls coming from the AIOTES universAAL Bridge.</td>
</tr>
<tr>
<td>Makro Recorder.</td>
<td>Service Manager / semantics specification</td>
<td>Yes</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>Sparql Tester</td>
<td>Maintenance Panel</td>
<td>Yes</td>
<td>No</td>
<td>This tool is used primarilly 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.</td>
</tr>
<tr>
<td>Security Profile Management</td>
<td>Not mapped with AIOTES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAL Space Monitor</td>
<td>Maintenance Panel</td>
<td>Yes</td>
<td>No</td>
<td>This tool is too universAAL Specific, it relates to the uSpace, describing all reachable IoT gateways with universAAL middleware.</td>
</tr>
</tbody>
</table>
B.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 software configuration, the version will increase in one and will be saved as disabled by default.
  - By default, when a configuration or software application is modified or 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 cannot be removed from the system. And can only be modified if they haven’t been assigned to any KP or 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.

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 software 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.

**Figure 116: A Sofja2 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 🗑️.
B.3.1 Mapping between deployment tools requirements and modules

Table 32: Mapping between SOFIA2 and ACTIVAGE development tools.

<table>
<thead>
<tr>
<th>FIWARE development tool</th>
<th>Corresponding ACTIVAGE development tool(s)</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Software Configuration tool</td>
<td>Not mapped with AIOTES</td>
<td></td>
</tr>
<tr>
<td>Assignment of Software in KPs</td>
<td>Not mapped with AIOTES</td>
<td></td>
</tr>
</tbody>
</table>

B.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.
B.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.

B.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.

B.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 was initially designed as a tablet application to be provided to installer and professional care-giver.

The final implementation of the tool is made with three different shared tables describing the specified information at recruitment level (identity, profile, location), at installation level (identity, location, devices, functions) and support/backend level (devices, functions). These three different shared tables contains all necessary information for the different stakeholders, with three level of access rights in order to grant privacy.

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 117. Installer tool mock 'up: discovering the ground plane of the installation

Figure 118. 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.

**B.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. The sensINact rpi distribution is deployed using a online binary repository (archiva). 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. Some dedicated shell scripts have been developed to automate the configuration at each new installation.

**B.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 119). This Studio Web navigator gives the status of both installed devices and deployed services.

*Figure 119: screenshot of the sensINact Studio Web device and service navigator*
B.5.1.4 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.

The sensiNact studio standalone software delivered with the sensiNact platform, as a tool on top of the sensiNact gateway API, 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 121).

Through the DSL\textsuperscript{43} editor, it is possible to program Event-Condition-Action scripts (called “applications”) in a dedicated language.

\textsuperscript{43} Domain Specific Language
Auto-completion, type checking, validation features are provided by the DSL editor (see Figure 123).

B.5.2 Mapping between deployment tools requirements and modules

Table 33: Mapping between DS6/sensiNact and ACTIVAGE deployment tools.

<table>
<thead>
<tr>
<th>sensiNact deployment tool</th>
<th>Corresponding ACTIVAGE deployment tool</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensiNact Studio and studio web</td>
<td>IoT infrastructure management tools / Device and Service manager</td>
<td>Yes</td>
</tr>
<tr>
<td>Installer tool</td>
<td>Distribution/deployment Component configuration and update</td>
<td>Yes</td>
</tr>
</tbody>
</table>

How?
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.

Why?
The Installer tool can be shared to other DS with minor adaptations

B.6 FIWARE

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.

### B.6.1 Mapping between deployment tools requirements and modules

<table>
<thead>
<tr>
<th>FIWARE deployment tool</th>
<th>Corresponding ACTIVAGE deployment tool</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker</td>
<td>Platform independent Available Deployment tools / Docker</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td><strong>How?</strong></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Why?</strong></td>
<td></td>
</tr>
<tr>
<td>Virtual Machine</td>
<td>Platform independent Available Deployment tools / Virtual Machine</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td><strong>How?</strong></td>
<td>For better performance, durability and host ability ACTIVAGE tools can be build as Virtual Machines.</td>
</tr>
<tr>
<td></td>
<td><strong>Why?</strong></td>
<td></td>
</tr>
</tbody>
</table>

### B.7 IoTivity

#### B.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:
B.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

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.
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>
<thead>
<tr>
<th>Device Properties</th>
<th>Bluetooth</th>
<th>Wifi</th>
<th>Zigbee</th>
<th>ZWave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Id</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
</tr>
<tr>
<td>Name</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
</tr>
<tr>
<td>Type</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
</tr>
<tr>
<td>Address</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
<td>mandatory</td>
</tr>
<tr>
<td>Password</td>
<td>optional</td>
<td>optional</td>
<td>optional</td>
<td>optional</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>optional</td>
<td>optional</td>
<td>mandatory</td>
<td>mandatory</td>
</tr>
<tr>
<td>Model</td>
<td>optional</td>
<td>optional</td>
<td>mandatory</td>
<td>optional</td>
</tr>
</tbody>
</table>

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.

```java
@Path("/service")
public class WebService {

@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;
}
```
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.

![Sequence Diagram](image)

**Figure 125: Device Management Mechanism for IoTivity Platform**
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.

```
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.

For this functionality, a PUT operation is used with the following parameters:
Resource URL: "/service/update"

Input

```json
{
   "name": "Bloodpressure monitor",
   "password": "789"
}
```

Output

```json
{
   "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.
For this functionality, a DELETE operation is used with the following parameters:

**Resource URL:** `/service/remove`

**Input**

```json
{
   "name": "Bloodpressure monitor"
}
```

**Output**

```json
{
   "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 129: 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

```json
{
    "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"
    }
    ]
}
```
B.7.3 Mapping between deployment tools requirements and modules

Table 36: Mapping between IoTivity and ACTIVAGE deployment tools.

<table>
<thead>
<tr>
<th>IoTivity deployment tool</th>
<th>Corresponding ACTIVAGE deployment tool</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy Setup</td>
<td>Deployment management tools / Component configuration</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>How?</td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td>The tool offers a C++ SDK, which cannot easily be used to implement ACTIVAGE web-based tools.</td>
</tr>
<tr>
<td>Device Management Tool</td>
<td>IoT infrastructure management tools / Device manager</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>How?</td>
<td>Why?</td>
</tr>
<tr>
<td></td>
<td>The tool is implemented as RESTful web services, which can be used directly in ACTIVAGE, with few modifications.</td>
<td></td>
</tr>
</tbody>
</table>

B.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 37: Mapping between SeniorSome and ACTIVAGE deployment tools.

<table>
<thead>
<tr>
<th>SeniorSome deployment tool</th>
<th>Corresponding ACTIVAGE deployment tool</th>
<th>Can the tool be used in AIOTES?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeniorSome API-based backend tool</td>
<td>Deployment management tools / Component configuration</td>
<td>Yes</td>
</tr>
<tr>
<td>How?</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Why?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seniorsome Device Management Tool</td>
<td>IoT infrastructure management tools / Device manager</td>
<td>Yes</td>
</tr>
<tr>
<td>How?</td>
<td>This tool can be used through API-configuration and possibly through an application layer integration.</td>
<td></td>
</tr>
<tr>
<td>Why?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Support</td>
<td>As a part of the AIOTES support documentation.</td>
</tr>
</tbody>
</table>

**B.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.

**B.9.1 Platform-specific deployment tools comparison**

Table 38: Deployment technologies per platform

<table>
<thead>
<tr>
<th>Platform</th>
<th>Orchestration deployment</th>
<th>Package technology</th>
<th>Deployment and configuration technology</th>
<th>Accessible service repository</th>
<th>Marketplace</th>
<th>GUI Tool for deployment and configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Yes</td>
<td>Pax Runner</td>
<td>karaf</td>
<td>Yes, but not specified</td>
<td>UniversAAL Control Center (uCC)</td>
<td>universAAL Control Center (uCC)</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>Yes</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Management Software Configuration</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>Yes</td>
<td>docker and virtual machine</td>
<td>new: virtual machines, future: docker</td>
<td>Not specified</td>
<td>Request definition tool</td>
<td>Yes</td>
</tr>
<tr>
<td>SensiNact</td>
<td>Not specified</td>
<td>Virtual machine</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>sensiNact studio</td>
</tr>
<tr>
<td>FIWARE</td>
<td>Yes</td>
<td>docker and virtual machine</td>
<td>docker and docker-compose</td>
<td>docker-hub, fiware-lab</td>
<td>fiware marketplace</td>
<td>fiware-lab</td>
</tr>
</tbody>
</table>
### Table 39: Deployment tools for device configuration

<table>
<thead>
<tr>
<th>Platform</th>
<th>Provides device deployment and configuration?</th>
<th>Provides status monitoring and software updates on physically deployed devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>Yes - Management software configuration</td>
<td>Yes - Easy Setup tool</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>Yes - Request definition tool</td>
<td>Yes - Request definition tool</td>
</tr>
<tr>
<td>SensiNact</td>
<td>No - manual pairing</td>
<td>No - manual pairing</td>
</tr>
<tr>
<td>FIWARE</td>
<td>Yes - IDAS and Orion</td>
<td>Yes - IDAS and Orion</td>
</tr>
<tr>
<td>IoTvity</td>
<td>Yes - Easy Setup tool</td>
<td>Yes - Easy Setup tool</td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
</tr>
</tbody>
</table>

### B.9.2 Mapping between platform-specific deployment tools and proposed ACTIVAGE deployment tools

#### Table 40: IoT infrastructure management tools

<table>
<thead>
<tr>
<th>Platform</th>
<th>Device manager</th>
<th>Service manager</th>
<th>Semantic auto-discovery platform</th>
<th>Benchmarking</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>SensiNact</td>
<td>sensiNact Studio and studio-Web</td>
<td>sensiNact Studio and studio-Web</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>FIWARE</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>IoTvity</td>
<td>Yes - Device Management tool</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
<td>Not available</td>
</tr>
</tbody>
</table>

#### Table 41: Deployment management tools

<table>
<thead>
<tr>
<th>Platform</th>
<th>Deployment manager</th>
<th>Component configuration</th>
<th>Maintenance panel</th>
<th>Update manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>universAAL</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
<td>Not specified</td>
</tr>
<tr>
<td>SOFIA2</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>OpenIoT</td>
<td>Yes - Virtual machine and docker manager</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
</tr>
<tr>
<td>Tool</td>
<td>Feature</td>
<td>Compliance</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>------------</td>
<td>--------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>SensiNact</td>
<td>Virtual machine</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>FIWARE</td>
<td>Yes - Virtual machine and docker manager</td>
<td>No tool matching AIOTES requirements</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>IoTvity</td>
<td>No tool matching AIOTES requirements</td>
<td>Yes - Easy setup</td>
<td>No tool matching AIOTES requirements</td>
<td></td>
</tr>
<tr>
<td>SeniorSome</td>
<td>Not available</td>
<td>Not Available</td>
<td>Not available</td>
<td></td>
</tr>
</tbody>
</table>

SensiNact: Virtual machine: No tool matching AIOTES requirements
FIWARE: Yes - Virtual machine and docker manager: No tool matching AIOTES requirements
IoTvity: No tool matching AIOTES requirements, Yes - Easy setup: No tool matching AIOTES requirements
SeniorSome: Not available: Not Available
Appendix C  Detailed ClickDigital architecture and implementation

The ClickDigital IDE is mainly characterized by its modular structure and currently composed by the following managers:

- Device manager
- Rules manager
- Visual manager
- Security manager

Devices manager:

While using ClickDigital, the developer as the main user should be able to address the following common use cases:
Figure 132 AONES ClickDigital API use cases
In the following, the related frontend implementation of each use case will be highlighted.

– Create user:

– Log in to ClickDigital
- View and edit profile

- Create new project

- Connect to AIOTES or a compatible IoT platform
Create dashboard:

- Use the drag and drop paradigm for UI – card widgets to manage the different available widgets (widget for device management, visualization, monitoring, rules creation...)

Add widget
- Delete widget

- Adjust themes settings
- **Add projects, dashboards, sheets**

- **Share projects**

- **Configure widgets**

Once exported the dashboard to a “use only mode”, the end-user will be able to login to the dashboard and exploit the already prepared functionalities.
While managing devices, ClickDigital widgets enable the developers to visually and smoothly create different rules, based on AIOTES connected to one or more IoT platforms:

Figure 133: ClickDigital – Rules manager structure

Figure 134: Rules related case diagram of the ClickDigital IDE visual tool.
The rules related widgets will be highlighted in details as following:

- Create a rule

- Visualize or edit an existing rule

- Find a rule
- Manage Rules (activate, deactivate… )

- Rules notification
Share rules

The user of the created dashboard will be also allowed to activate or deactivate rules.
ClickDigital integrates a visualisation manager with the architecture as illustrated in the following Figure.

![ClickDigital – visualisation manager structure](image)

Figure 135: ClickDigital – visualisation manager structure

The developer, while using ClickDigital plugged to AIOTES and the targeted IoT platform, should be able, through the usage of widgets, to perform following tasks:

![Devices related use case diagram of the ClickDigital IDE visual tool.](image)

Figure 136 Devices related use case diagram of the ClickDigital IDE visual tool.

In the following, a detailed overview about the created devices related widgets are highlighted:

- ClickDigital devices panel overview
- View existing devices – find a device

- Add a device

- Monitor/Visualize a device status
Deliverable 4.4 — Developers toolkit and deployment support

- Control device

- Wallplug Switch

- Bedroom Light

- Dimmer