<table>
<thead>
<tr>
<th>Deliverable No.</th>
<th>D5.3</th>
<th>Due Date</th>
<th>31-12-2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Report</td>
<td>Dissemination Level</td>
<td>Public</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
<td>Status</td>
<td>Release 1</td>
</tr>
<tr>
<td>Description</td>
<td>This deliverable provides the compilation of the validation results and their analysis until the start of the Grow phase of the innovation path.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work Package</td>
<td>WP5 – ACTIVAGE IoT Ecosystem Suite Integration.</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>Philippe Dallemagne</td>
<td>19 CSEM</td>
<td><a href="mailto:pda@csem.ch">pda@csem.ch</a></td>
</tr>
<tr>
<td>Byron Ortiz Sanchez</td>
<td>03 TVES</td>
<td><a href="mailto:byrort@televes.com">byrort@televes.com</a></td>
</tr>
<tr>
<td>Alejandro M. Medrano Gil</td>
<td>05 UPM</td>
<td><a href="mailto:amedrano@lst.tfo.upm.es">amedrano@lst.tfo.upm.es</a></td>
</tr>
<tr>
<td>Helmi Ben Hmida</td>
<td>06 Fh-IGD</td>
<td><a href="mailto:.ben.hmida@igd.fraunhofer.de">.ben.hmida@igd.fraunhofer.de</a> &gt;</td>
</tr>
<tr>
<td>Stéphane Bergeon</td>
<td>07 CEA</td>
<td><a href="mailto:Stephane.BERGEON@cea.fr">Stephane.BERGEON@cea.fr</a></td>
</tr>
<tr>
<td>Mathieu Gallissot</td>
<td>07 CEA</td>
<td><a href="mailto:mathieu.gallissot@cea.fr">mathieu.gallissot@cea.fr</a></td>
</tr>
<tr>
<td>Thanos Stavropoulos</td>
<td>08 CERTH</td>
<td><a href="mailto:athstavr@iti.gr">athstavr@iti.gr</a></td>
</tr>
<tr>
<td>Nikolaos Kaklanis</td>
<td>08 CERTH</td>
<td><a href="mailto:nkak@iti.gr">nkak@iti.gr</a></td>
</tr>
<tr>
<td>Stefanos Stavrotheodoros</td>
<td>08 CERTH</td>
<td><a href="mailto:kvotis@iti.gr">kvotis@iti.gr</a></td>
</tr>
<tr>
<td>Konstantinos Votis</td>
<td>08 CERTH</td>
<td><a href="mailto:kvotis@iti.gr">kvotis@iti.gr</a></td>
</tr>
<tr>
<td>Dimitrios Tzovaras</td>
<td>08 CERTH</td>
<td><a href="mailto:Dimitrios.Tzovaras@iti.gr">Dimitrios.Tzovaras@iti.gr</a></td>
</tr>
<tr>
<td>Regel G. Usach</td>
<td>11 UPV</td>
<td><a href="mailto:regonus@doctor.upv.es">regonus@doctor.upv.es</a></td>
</tr>
<tr>
<td>Matilde Julián</td>
<td>11 UPV</td>
<td><a href="mailto:majuse@dcom.upv.es">majuse@dcom.upv.es</a></td>
</tr>
<tr>
<td>Clara Valero</td>
<td>11 UPV</td>
<td><a href="mailto:clavalpe@upv.es">clavalpe@upv.es</a></td>
</tr>
<tr>
<td>Andreu Belsa</td>
<td>11 UPV</td>
<td><a href="mailto:anbelpel@upv.es">anbelpel@upv.es</a></td>
</tr>
<tr>
<td>Huy Le Van</td>
<td>13 NUI Galway</td>
<td><a href="mailto:huy.levan@insight-centre.org">huy.levan@insight-centre.org</a></td>
</tr>
<tr>
<td>Aqueel H. Kazmi</td>
<td>13 NUI Galway</td>
<td><a href="mailto:aqeel.kazmi@insight-centre.org">aqeel.kazmi@insight-centre.org</a></td>
</tr>
<tr>
<td>Pierre Barralon</td>
<td>15 TECNALIA</td>
<td><a href="mailto:pierre.barralon@tecnalia.com">pierre.barralon@tecnalia.com</a></td>
</tr>
<tr>
<td>Cristina Rodríguez De Pablo</td>
<td>15 TECNALIA</td>
<td><a href="mailto:cristina.rodriguez@tecnalia.com">cristina.rodriguez@tecnalia.com</a></td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Sept. 2018</td>
<td>0.1</td>
<td>First structure (results, analysis, etc.)</td>
</tr>
<tr>
<td>10 Oct. 2018</td>
<td>0.1b</td>
<td>Detailed structure</td>
</tr>
<tr>
<td>28 Nov 2018</td>
<td>0.2</td>
<td>SIL testing, AloTES Management, Tools, Datalake</td>
</tr>
<tr>
<td>10 Dec. 2018</td>
<td>0.9</td>
<td>Version 1, for internal review</td>
</tr>
<tr>
<td>15 Jan. 2019</td>
<td>0.9b</td>
<td>Revision from reviewers integrated and distributed</td>
</tr>
<tr>
<td>8 Feb. 2019</td>
<td>0.9c</td>
<td>Comments from reviewers applied in the document</td>
</tr>
<tr>
<td>14 Feb. 2019</td>
<td>1.0</td>
<td>Final</td>
</tr>
<tr>
<td>8 March 2019</td>
<td>2.0</td>
<td>Moved Data Lake and analytics details to Appendix</td>
</tr>
</tbody>
</table>

Key data

<table>
<thead>
<tr>
<th>Keywords</th>
<th>AloTES, Validation, results, integration, test, analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead Editor</td>
<td>Philippe Dallemagne, 19 CSEM</td>
</tr>
<tr>
<td>Internal Reviewers</td>
<td>Teresa Gallelli, Lepida</td>
</tr>
<tr>
<td></td>
<td>Rami Mäkelä, SeniorSome</td>
</tr>
</tbody>
</table>
Abstract

This document is the deliverable D5.3 “Intermediate Validation Results” and reports about the tests performed at the technical and functional level in the various deployment sites and labs, as well as the assumptions, setup, tools, preparation and execution of the various steps for performing the verification tests and running validation scenarios, in which the AIoTES (ACTIVAGE IoT Ecosystem Suite) is involved. At the time of delivery, the results are partial and mostly concentrated on the verification of the AIoTES, with respect to the technical specification. A first validation of the Semantic Interoperability Layer is reported.

D5.3 refers to the results of WP3, WP4 and WP5 and builds on top of D5.1. The document is the result of the joint efforts of the activities T5.1 “Integration and ACTIVAGE IoT Ecosystem Suite”, T5.2 “Test and validation of ACTIVAGE IoT Ecosystem Suite” and T5.3 “Test and Validation Framework for deployment sites” which are part of Work Package WP5 “ACTIVAGE IoT Ecosystem Suite Integration”.

The document presents the results of the tests using the format and content provided by the various contributors, who used the reporting tools and elements available at the time of the initial integration and execution of tests. This is a first version of the report based on an initial plan aiming at checking independently many components by their developers, in the way they find more appropriate (so they choose the tests and testing procedures based on their expertise and knowledge in the specific development, what seemed appropriate, and as far as these are intermediate results or first testing). At this stage, it was not possible to guaranty any homogeneity across the results from the testing of very different and heterogeneous components. This also reflects the various maturity levels and advancement between the components and DSs at the time of D5.3 preparation. As a consequence, some functional testing is used in some components instead of a deeper type of testing, to check in a timely way a correct performance.

This document is organised as follows: it begins with general information, acronyms and definitions, followed by an introduction (Chapter 2) that describes the aim, context and content of the deliverable; Chapter 2 also introduces the concept of verification and validation. The third Chapter is dedicated to the results obtained during the verification of the AIoTES. Chapter 4 is dedicated to the initial results obtained in the validation of the AIoTES. The last Chapter initiates the discussion about the innovation path.

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><strong>TABLE OF CONTENTS</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>LIST OF TABLES</strong></td>
<td>8</td>
</tr>
<tr>
<td><strong>LIST OF FIGURES</strong></td>
<td>9</td>
</tr>
<tr>
<td><strong>ACRONYMS</strong></td>
<td>13</td>
</tr>
<tr>
<td><strong>1 ABOUT THIS DOCUMENT</strong></td>
<td>15</td>
</tr>
<tr>
<td>1.1 Deliverable Context</td>
<td>15</td>
</tr>
<tr>
<td><strong>2 INTRODUCTION</strong></td>
<td>17</td>
</tr>
<tr>
<td>2.1 Verification and validation description</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Verification and validation scope and strategy</td>
<td>19</td>
</tr>
<tr>
<td><strong>3 VERIFICATION</strong></td>
<td>20</td>
</tr>
<tr>
<td>3.1 Semantic interoperability layer</td>
<td>20</td>
</tr>
<tr>
<td>3.1.1 SIL Testing facilities</td>
<td>20</td>
</tr>
<tr>
<td>3.1.2 Bridge unit tests</td>
<td>20</td>
</tr>
<tr>
<td>3.1.3 Semantic unit test – (Messages, semantic translation &amp; alignments)</td>
<td>22</td>
</tr>
<tr>
<td>3.1.4 Integration tests</td>
<td>24</td>
</tr>
<tr>
<td>3.1.5 Testing environment</td>
<td>26</td>
</tr>
<tr>
<td>3.1.6 IoTivity bridge testing</td>
<td>27</td>
</tr>
<tr>
<td>3.1.6.1 Bridge unit tests</td>
<td>27</td>
</tr>
<tr>
<td>3.1.6.2 Integration tests</td>
<td>30</td>
</tr>
<tr>
<td>3.1.6.3 Testing environment</td>
<td>33</td>
</tr>
<tr>
<td>3.1.7 SIL API</td>
<td>34</td>
</tr>
<tr>
<td><strong>3.2 AIOTES MANAGEMENT</strong></td>
<td>34</td>
</tr>
<tr>
<td><strong>3.3 DEVELOPMENTS TOOLS</strong></td>
<td>42</td>
</tr>
<tr>
<td>3.3.1 ACTIVAGE Ontology Explorer</td>
<td>42</td>
</tr>
<tr>
<td>3.3.2 Device Semantics Editor</td>
<td>42</td>
</tr>
<tr>
<td>3.3.3 Service Semantics Editor</td>
<td>42</td>
</tr>
<tr>
<td>3.3.4 Data Analyser</td>
<td>42</td>
</tr>
<tr>
<td>3.3.5 DataModel Workbench</td>
<td>43</td>
</tr>
<tr>
<td>3.3.6 Metadata Storage Explorer</td>
<td>43</td>
</tr>
<tr>
<td>3.3.7 AIOTES IDE tool</td>
<td>43</td>
</tr>
<tr>
<td>3.3.8 AIOTES SIL tool</td>
<td>44</td>
</tr>
<tr>
<td>3.3.9 Service Composer</td>
<td>44</td>
</tr>
<tr>
<td>3.3.10 Code generation tool</td>
<td>44</td>
</tr>
<tr>
<td>3.3.11 Wiki</td>
<td>45</td>
</tr>
<tr>
<td><strong>3.4 DATA LAKE</strong></td>
<td>45</td>
</tr>
<tr>
<td><strong>3.5 ANALYTICS</strong></td>
<td>45</td>
</tr>
<tr>
<td><strong>3.6 SECURITY AND PRIVACY MANAGEMENT</strong></td>
<td>46</td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

3.7 AIOTES API ............................................................................................................. 46
3.8 MARKETPLACE ....................................................................................................... 49

4 VALIDATION ............................................................................................................. 51
4.1 SEMANTIC INTEROPERABILITY LAYER ............................................................... 51
  4.1.1 First scenario of validation ............................................................................. 51
    4.1.1.1 Preparation of first scenario ................................................................. 52
    4.1.1.2 Execution of first scenario .................................................................... 54
  4.1.2 Second scenario of validation ....................................................................... 56
    4.1.2.1 Preparation of the second scenario ..................................................... 57
    4.1.2.2 Execution of second scenario ............................................................... 59

5 CONCLUSION AND INNOVATION PATH ............................................................... 61
5.1 INNOVATION PATH ............................................................................................. 61
5.2 OPEN CALL .......................................................................................................... 62

REFERENCES .............................................................................................................. 63

APPENDIX A : SIL API FUNCTIONAL TESTS .......................................................... 65

APPENDIX B : TESTS DONE ON DS6 ISÈRE ......................................................... 101
  1-Sensors > 1-Info sNa .......................................................................................... 101
  2-Actuators > 1-Action sNa .................................................................................. 108
  3-Network .............................................................................................................. 110
  4-Interfaces > 1-Parameters .............................................................................. 110
  4-Interfaces > 3-Notification ............................................................................. 116
  5-Functions > Area of walking ............................................................................ 119
  5-Functions > Automatic light ........................................................................... 124
  5-Functions > Bed statistics .............................................................................. 130
  5-Functions > Shower alert ................................................................................ 135
  5-Functions > Temperature alert ....................................................................... 140

APPENDIX C DS 3 MADRID - UNIVERSAAL ............................................................. 148
  C.1 EQUIMETRIX DEVICE ..................................................................................... 148
  C.2 BALANCE ASSESSMENT .............................................................................. 149
  C.3 BALANCE TRAINING .................................................................................... 150
  C.4 EQUIMETRIX AND AIOTES ....................................................................... 151
  C.5 EQUIMETRIX DATA ....................................................................................... 152
  C.6 TECHNICAL VERIFICATION ....................................................................... 154
    C.6.1 Methodology ............................................................................................. 154
    C.6.2 Continuous testing .................................................................................. 155

APPENDIX D DEVELOPMENTS TOOLS ................................................................. 157
  D.1 ACTIVAGE ONTOLOGY EXPLORER ............................................................. 157
  D.2 DEVICE SEMANTICS EDITOR ..................................................................... 161
  D.3 DATA ANALYSER ......................................................................................... 164
  D.4 DATA MODEL WORKBENCH ......................................................................... 166
    D.4.1 Unit tests .................................................................................................. 166
List of tables

Table 1: Input and expected outcome examples for testing the method ‘toJenaModel’................. 29
Table 2: Description of available tests and their steps .......................................................... 30
Table 4: KPIs used on DS 6 with sensINact ........................................................................ 36
Table 5: AioTES API summary .............................................................................................. 46
Table 6: Marketplace functionality per user role .................................................................... 49
Table 7: Equimetrix data stored ............................................................................................. 152
Table 8: Ontology explorer tool tests ...................................................................................... 157
Table 9: Semantics editor tool tests ........................................................................................ 161
Table 10: Data analyzer tool tests .......................................................................................... 164
Table 11: API functional test inputs and expected outputs ...................................................... 179
Table 12: Independent data storage API general information ................................................ 188
Table 13: CreateDB operation ............................................................................................... 188
Table 14: DeleteDB operation ............................................................................................... 190
Table 15: Select operation ..................................................................................................... 191
Table 16: Insert operation ...................................................................................................... 194
Table 17: Update operation .................................................................................................. 196
Table 18: Delete operation ..................................................................................................... 199
Table 23. API /createIndex ..................................................................................................... 207
Table 24. API /updateIndex .................................................................................................... 209
Table 25. API /deleteIndex/{id} ........................................................................................... 212
Table 26. API /getAllIndex .................................................................................................... 214
Table 27. API /getIndex/{id} ............................................................................................... 217
Table 34: Marketplace functionality per user role ................................................................. 235
Table 35. Marketplace GitHub issue statistics ........................................................................ 237
List of figures

FIGURE 1: AIOTES ARCHITECTURE AND ITS MAIN COMPONENTS ..................................................... 17
FIGURE 2: VERIFICATION AND VALIDATION PROCESSES .......................................................... 18
FIGURE 3: UNIT TEST MESSAGE EXAMPLE ................................................................................. 21
FIGURE 4: SYNTAXIC TRANSLATION TEST .................................................................................. 22
FIGURE 5: SEMANTIC TRANSLATION CONFIGURATION ............................................................... 23
FIGURE 6: SEMANTIC TRANSLATION TEST .................................................................................. 23
FIGURE 7: SEMANTIC TRANSLATION MONITORING ................................................................. 24
FIGURE 8: INTEGRATION TEST EXAMPLE ................................................................................... 25
FIGURE 9: INTEGRATION TEST EXAMPLE: DEVICES REGISTRATION ..................................... 25
FIGURE 10: TESTING ENVIRONMENT SCHEME ......................................................................... 26
FIGURE 11: SERVICES DEPLOYED IN THE SERVER ................................................................. 26
FIGURE 12: IoTIVITY BRIDGE ARCHITECTURE ....................................................................... 27
FIGURE 13: JUnit tests for the CoAP client module .................................................................... 28
FIGURE 14: GENERAL ARCHITECTURE OF THE FIRST SCENARIO OF VALIDATION .............. 52
FIGURE 15: Step1 - Connection of the application to the SIL ...................................................... 53
FIGURE 16: Step2 – Addition of SIL bridges/alignments and inclusion of platforms in the SIL .... 53
FIGURE 17: Step3 – Subscription of the application to IoT data .................................................. 54
FIGURE 18: Architecture Overview of execution with SOFIA2 of Validation of Multiplatform Application .................................................................................................................. 55

FIGURE 19: Architecture Overview of execution with UniversAAL of Validation of Multiplatform Application .................................................................................................................. 56
FIGURE 20: Architecture Overview of the Second Scenario of Validation ................................... 57
FIGURE 21: Step1 – Use of SIL bridges/alignments and inclusion of platforms in the SIL .......... 58
FIGURE 22: Step2 – Creation of virtual devices in SOFIA2 associated to real devices from an uAAL platform ......................................................................................................................... 58

FIGURE 23: Overview of the Execution of the Second Scenario of Validation ......................... 60
FIGURE 24: IPSM GET ALIGNMENTS ....................................................................................... 66
FIGURE 25: IPSM POST ALIGNMENTS ..................................................................................... 67
FIGURE 26: IPSM POST ALIGNMENTS RESPONSE ............................................................... 68
FIGURE 27: IPSM DELETE ALIGNMENTS .................................................................................. 69
FIGURE 28: IPSM DELETE ALIGNMENTS RESPONSE ............................................................. 69
FIGURE 29: IPSM GET ALIGNMENT BY NAME AND VERSION ................................................ 70
FIGURE 30: IPSM GET ALIGNMENT BY NAME AND VERSION RESPONSE ......................... 71
FIGURE 31: IPSM CONVERT ALIGNMENT REQUEST .............................................................. 72
FIGURE 32: IPSM CONVERT ALIGNMENT RESPONSE ............................................................ 72
FIGURE 33: IPSM CONVERT ALIGNMENT TO TURTLE FORMAT REQUEST ....................... 73
FIGURE 34: IPSM CONVERT ALIGNMENT TO TURTLE FORMAT RESPONSE ..................... 74
FIGURE 35: REGISTER CLIENT (REQUEST) ............................................................................. 75
D5.3 Intermediate Validation Results

Figure 36: Register Client (Response) ................................................................. 76
Figure 37: List Clients (Request) ........................................................................ 77
Figure 38: List Clients (Response) ................................................................. 77
Figure 39: Get Client (Request) ........................................................................ 78
Figure 40: Get Client (Response) ................................................................. 78
Figure 41: Get Supported Platform Types (Request) ........................................ 79
Figure 42: Get Supported Platform Types (Response) ....................................... 79
Figure 43: Register Platform (Request) ............................................................. 81
Figure 44: Register Platform (Response) .......................................................... 81
Figure 45: List Platforms (Request) ................................................................. 83
Figure 46: List Platforms (Response) ............................................................... 83
Figure 47: Get Platform (Request) ................................................................. 84
Figure 48: Get Platform (Response) ................................................................. 84
Figure 49: Update Platform (Request) ............................................................. 86
Figure 50: Update Platform (Response) ........................................................... 86
Figure 51: Unregister Platform (Request) ......................................................... 87
Figure 52: Unregister Platform (Response) ....................................................... 87
Figure 53: Create Virtual Device (Request) ...................................................... 89
Figure 54: Create Virtual Device (Response) ................................................... 89
Figure 55: Get Devices (Request) ................................................................. 91
Figure 56: Get Devices (Response) ................................................................. 91
Figure 57: Update Device (Request) ................................................................. 92
Figure 58: Update Device (Response) ............................................................. 93
Figure 59: Delete Device (Request) ................................................................. 94
Figure 60: Delete Device (Response) .............................................................. 94
Figure 61: Subscribe (Request) ....................................................................... 95
Figure 62: Subscribe (Response) ..................................................................... 95
Figure 63: List Subscriptions (Request) ......................................................... 96
Figure 64: List Subscriptions (Response) ....................................................... 97
Figure 65: Unsubscribe (Request) ................................................................. 97
Figure 66: Unsubscribe (Response) ............................................................... 98
Figure 67: Get Response Messages (Request) ............................................... 99
Figure 68: Get Response Messages (Response) ............................................. 100
Figure 69: Equimetrix Device. Wearable camera attached onto the trunk. Flexipad mattress below user’s feet. Equimetrix software for balance assessment and training .................................................. 149
Figure 70: Equimetrix Assessment Screenshot .................................................. 150
Figure 71: Equimetrix Balance Training Game ............................................. 151
Figure 72. DS Madrid Architecture ............................................................... 152
Figure 73: Example of Equimetrix Data Log File ......................................... 154
Figure 74. Example of universAAL repository with Equimetrix data .......... 155
Figure 75. Users of a specific machine (ID: d8aa38ee) .................................... 156

Version 1.0  |  2019-02-14  |  ACTIVAGE ©
D5.3 Intermediate Validation Results

FIGURE 76: DATABASE TESTS ................................................................. 168
FIGURE 77: TABLE TESTS ........................................................................ 170
FIGURE 78: SCHEMA TEST ...................................................................... 172
FIGURE 79: QUERY TEST ......................................................................... 173
FIGURE 80: DATABASE CREATION FUNCTIONAL SAMPLE VALIDATION RESULT .................................................. 174
FIGURE 81: MODEL TEST ........................................................................ 176
FIGURE 82: MODEL CREATION FUNCTIONAL SAMPLE VALIDATION RESULT ............................................................. 177
FIGURE 83: CREATEDB FUNCTIONAL SAMPLE TEST ................................. 189
FIGURE 84: DELETEDB FUNCTIONAL SAMPLE TEST .................................. 191
FIGURE 85: SELECT FUNCTIONAL SAMPLE TEST ..................................... 193
FIGURE 86: INSERT FUNCTIONAL SAMPLE TEST ...................................... 196
FIGURE 87: UPDATE FUNCTIONAL SAMPLE TEST ..................................... 198
FIGURE 88: DELETE FUNCTIONAL SAMPLE TEST ...................................... 200
FIGURE 91: THE OUTPUT OF THE Mocha INTEGRATION TESTING FOR THE "CREATEMODEL" AND "GETMODEL" ANALYTICS METADATA STORAGE SERVICES ......................................................... 205
FIGURE 92: THE OPERATIONS IN THE ACTIVAGE DATA INTEGRATION API INDEXING TESTED ........................................ 206
FIGURE 93: /CREATEINDEX FUNCTIONAL SAMPLE TEST ......................... 208
FIGURE 94: /CREATEINDEX FUNCTIONAL RESPONSE VALIDATION TEST ............................................................................. 209
FIGURE 95: /UPDATEINDEX FUNCTIONAL SAMPLE TEST ....................... 211
FIGURE 96: /UPDATEINDEX FUNCTIONAL RESPONSE VALIDATION TEST ............................................................................. 212
FIGURE 97: /DELETEINDEX/{ID} FUNCTIONAL SAMPLE TEST .................. 213
FIGURE 98: /DELETEINDEX/{ID} FUNCTIONAL RESPONSE VALIDATION TEST ................................................................. 214
FIGURE 99: /GETALLINDEX FUNCTIONAL SAMPLE TEST .......................... 216
FIGURE 100: /GETALLINDEX FUNCTIONAL RESPONSE VALIDATION TEST .............................................................................. 217
FIGURE 101: /GETINDEX/{ID} FUNCTIONAL SAMPLE TEST ........................ 219
FIGURE 102: /GETINDEX/{ID} FUNCTIONAL RESPONSE VALIDATION TEST .............................................................................. 220
FIGURE 103: THE BASIC UNIT OF THE DATA ANALYTICS COMPONENT. ................................................................................. 228
FIGURE 104: COMPARISON OF METHOD OUTPUT TO EXPECTED OUTPUT ............................................................................. 229
FIGURE 105: EXAMPLE MOCHA UNIT TESTING BLOCK FOR THE K-MEANS ANALYTICS METHOD ........................................... 230
FIGURE 106: THE OUTPUT OF THE MOCHA UNIT TESTING FOR THE K-MEANS ANALYTICS METHOD ........................................ 231
FIGURE 107: WRAPPING OF A DATA ANALYTICS METHOD AS A WEB SERVICE, TO BE INTEGRATED WITH THE AIoT ES ENVIRONMENT ............................................................... 231
FIGURE 108: INTEGRATION TESTING OF A DATA ANALYTICS WEB SERVICE BY COMPARING THE SERVICE OUTPUT TO EXPECTED OUTPUT ................................................................. 232
FIGURE 109: EXAMPLE MOCHA INTEGRATION TESTING BLOCK FOR THE "KMEANS" ANALYTICS METHOD ......................... 232
FIGURE 110: THE OUTPUT OF THE MOCHA INTEGRATION TESTING FOR THE K-MEANS ANALYTICS METHOD .................................................. 233
FIGURE 111. MARKETPLACE HOMEPAGE ................................................. 235
FIGURE 112. ISSUE CATEGORY DISTRIBUTION ...................................... 237
FIGURE 113. TIME TO RESOLVE EACH ISSUE (ISSUE ID IN X-AXIS) MEASURED IN DAYS (Y-AXIS) ................................. 238
FIGURE 114. OPENED, CLOSED AND (STILL) OPEN ISSUES PER MONTH ............................................................................. 238
FIGURE 115. LIGHTHOUSE TESTING BEFORE OPTIMIZATIONS ................ 239
FIGURE 116. SAMPLE OF SEO RECOMMENDATIONS PROVIDED BY LIGHTHOUSE .............................................................. 240
D5.3 Intermediate Validation Results

**Figure 117. Lighthouse testing after optimization and passed audits** ........................................ 241
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVAGE</td>
<td>ACTivating InnoVative IoT smart living environments for AGEing well</td>
</tr>
<tr>
<td>AHA</td>
<td>Active and Healthy Ageing</td>
</tr>
<tr>
<td>AloTES</td>
<td>ACTIVAGE IoT Ecosystem Suite</td>
</tr>
<tr>
<td>AIOTI</td>
<td>The Alliance for Internet of Things Innovation</td>
</tr>
<tr>
<td>AMQP</td>
<td>Advanced Message Queuing Protocol</td>
</tr>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
<tr>
<td>ARM</td>
<td>Architectural Reference Model</td>
</tr>
<tr>
<td>ASN</td>
<td>Abstract Syntax Notation</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CBOR</td>
<td>Concise Binary Object Representation</td>
</tr>
<tr>
<td>CEP</td>
<td>Complex Event Processing</td>
</tr>
<tr>
<td>CM</td>
<td>Communication Model</td>
</tr>
<tr>
<td>CoAP</td>
<td>Constrained Application Protocol</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DDS</td>
<td>Data Distribution Service</td>
</tr>
<tr>
<td>DM</td>
<td>Domain Model</td>
</tr>
<tr>
<td>DS</td>
<td>Deployment Site</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain Specific Language</td>
</tr>
<tr>
<td>ER</td>
<td>Exploitable Result</td>
</tr>
<tr>
<td>EXI</td>
<td>Efficient XML Interchange</td>
</tr>
<tr>
<td>FG</td>
<td>Functionality Groups</td>
</tr>
<tr>
<td>GDPR</td>
<td>General Data Protection Regulation</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Mark-up Language</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IM</td>
<td>Information Model</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IoT-A</td>
<td>Internet of Things – Architecture</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPSM</td>
<td>Inter Platform Semantic Mediator</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>JSON-LD</td>
<td>JSON for Linking Data</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>M2M</td>
<td>Machine-to-Machine</td>
</tr>
<tr>
<td>MQTT</td>
<td>Message Queue Telemetry Transport</td>
</tr>
<tr>
<td>MS</td>
<td>Milestone</td>
</tr>
<tr>
<td>MSE</td>
<td>Metadata Storage Exploder</td>
</tr>
<tr>
<td>MVC</td>
<td>Model View controller</td>
</tr>
<tr>
<td>MW</td>
<td>Middleware</td>
</tr>
<tr>
<td>NGN</td>
<td>Next Generation Networks</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organization for the Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>OC</td>
<td>Open Call</td>
</tr>
<tr>
<td>OCF</td>
<td>Open Connectivity Foundation</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>QoL</td>
<td>Quality of Live</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RA</td>
<td>Reference Architecture</td>
</tr>
<tr>
<td>REST</td>
<td>REpresentational State Transfer</td>
</tr>
<tr>
<td>RM</td>
<td>Reference Model</td>
</tr>
<tr>
<td>SEO</td>
<td>Search Engine Optimization</td>
</tr>
<tr>
<td>SIL</td>
<td>Semantic Interoperability Layer</td>
</tr>
<tr>
<td>sNa</td>
<td>sensiNact</td>
</tr>
<tr>
<td>SOTA</td>
<td>State Of The Art</td>
</tr>
<tr>
<td>uAAL</td>
<td>universAAL</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>US</td>
<td>Use Case</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>UX</td>
<td>User eXperience</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark-up Language</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WOQ</td>
<td>DS of Woquaz</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>XMPP</td>
<td>Extensible Messaging and Presence Protocol</td>
</tr>
</tbody>
</table>
1 About This Document

Deliverable D5.3 “Intermediate Validation Results” reports the results of the intermediate validation phase. This document reports about the tests performed at the technical and functional level in the various deployment sites.

1.1 Deliverable context

<table>
<thead>
<tr>
<th>Project item</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td>This deliverable is a first step for verifying the advancement and achievement level of the ACTIVAGE objectives and requirements. At this stage (initial tests) no KPI are significant, but the work described in the document definitely show the positive progress towards their completion. This deliverable is the basis for the evaluation of</td>
</tr>
<tr>
<td></td>
<td>Number of Value-Added services (Main)</td>
</tr>
<tr>
<td></td>
<td>Number of platforms integrated to ACTIVAGE (O1)</td>
</tr>
<tr>
<td></td>
<td>Number of 3rd party services integrated to ACTIVAGE (O6)</td>
</tr>
<tr>
<td>Exploitable results</td>
<td>This deliverable indirectly contributes to most of the project exploitable results (ERs) by assessing the technical achievement of the project objectives and showing that the goal is reachable.</td>
</tr>
<tr>
<td>Work plan</td>
<td>As per the DoA, the deliverable is mostly linked to the activity performed in WP5 and more precisely to the task 5.2. However, since it reports about the advancement of the whole IoT cluster, almost all tasks of WP3, 4 and 5 are involved. The return of experience as shown in this deliverable is critical for the adoption of the AIoTES in the DS and OC.</td>
</tr>
<tr>
<td>Milestones</td>
<td>This deliverable contributes to Milestone 2, since it demonstrate the appropriateness and effectiveness of the AIoTES with respect to the project objectives. It also contributes to the correct execution of the transition between the GROW and EXPAND phases, in particular for integrating the open callers, since it shows also the necessity for higher consistency across components and DS reporting (use of existing templates, guided implementation of the KPIs, etc.)</td>
</tr>
<tr>
<td>Deliverables</td>
<td>This deliverable is based on D5.1 “Integration plan and operational framework”, and serves as the basis for the D5.4 “Final Validation Results”. It must be read in conjunction with D6.3 “First evaluation report” and D6.4 “Second evaluation report”, although the KPIs (cf. D6.1) did not structure its reporting. It can be used as an informative input by the contributors to D9.4 “Updated KPI Evolution Report (I to IX)”.</td>
</tr>
<tr>
<td>Risks</td>
<td>This deliverable concretely addresses and lower the following risks: inappropriateness, in effectiveness of the AIoTES</td>
</tr>
</tbody>
</table>
technical approach and choices, impossibility of integration in the DS platforms. The corresponding risks identified by ACTIVAGE are R3 Accessibility, Transparency, New technologies & IoT-related equipment.

From the patient point of view, the methodology exhibited by the ACTIVAGE project, to which the present deliverable pertains, ensures the technical ground for a secure, safe and privacy preserving solution (R1: Loss of Privacy Control, R2: Data Security). Note however that these aspects will be evaluated in the deliverable D5.4.
2 Introduction

2.1 Verification and validation description

As described in Deliverable 5.1, Verification checks every single component of the ACTIVAGE AIoTES for its conformance to its specification, which in turn aims at checking the compliance to the technical requirements derived from the user requirements. Verification consists mostly in unit test of independent components and set of components (as depicted in Figure 1), before and after their integration into a consistent and functional system. As shown in Figure 2. These tests can be performed in lab, in testbeds or during field test over limited periods.

In Figure 1 shows the internal components of AIoTES, which are required to be tested to check their correct performance and to allow the performance of AIoTES as a whole.

As it can be seen in the block scheme, AIoTES is composed by three main blocks: the Semantic Interoperability Layer (SIL), the Service Layer and the Security and Privacy Management. Moreover, AIoTES will expose an external API that offers access to AIoTES functionality and services. Next, the AIoTES modules are described.

The Semantic Interoperability Layer is an AIoTES component that allows the integration and interoperation of DS IoT platforms. Currently in IoT there is a general lack of interoperability that hampers interconnection, interoperation and integration among IoT platforms and systems, which are unable to directly communicate among them. The Semantic
Interoperability Layer or SIL solves this problem providing interoperability across all the DS IoT platforms and AIoTES.

The Security and Privacy Module guarantees the security and privacy within AIoTES.

The Service Layer is composed by a set of independent components that provide additional functionality to the AIoTES environment. Components of the service layer are AIoTES Management system, Data Lake (an independent database) and Data Analytics. Furthermore there are specific AIoTES tools that support the development and component deployment in AIoTES environments.

Further description of the AIoTES components can be found in D3.7. In this deliverable the results of testing the aforementioned components of the ACTIVAGE environment are presented.

Figure 2: Verification and validation processes

The Validation corresponds to meeting the “user” requirements, first at the individual component level, whenever possible and most often, at the system level after a partial or full integration. The validation of the AIoTES aims therefore at checking the fulfilling of the user expectations. As such the validation is performed by pilot deployment on dedicated sites. In the case of ATIVAGE, depending on the user requirement, the user may be either the DS owner, the social service operator, the community, the regulation entities, etc.

Once validated the AIoTES or the ACTIVAGE system can be deployed to other pilot sites for extensive use, which will put the system under stress tests, up to a point where it can be declared as validated and therefore good enough to be generalised. Eventually, the system will enter the operational phase and it can be subject to maintenance (whenever necessary or periodically).

Note that stress test is a test that tests the boundaries of the capabilities of the system.

---

1 The notion of « user » is relative to the entity under evaluation. For example the user of development framework library is a application developer and the user of the outcome of the developed product maybe an operator. Similarly, the user of the machine tool is the machine operator and the user of the end product of the manufacturing process, e.g. a car is the driver.
Note that the sequence of verification and validation maybe iterated depending on the development approach that is taken either by the development team.

## 2.2 Verification and validation scope and strategy

This process deals with testing and validating the tangible outcome of WP 3 (ACTIVAGE Secure Interoperability Layer) and WP 4 (ACTIVAGE Application support Tools and Services Layer), mainly AoTIES including all layers and Bridges added to all created support tools in order to assess the readiness of integration with DS application. Furthermore, it deals with the level of stability and reliability when using it later in the different DSs. Finally, it allows for the traceability from the requirement to the test results, which show the compliance of every individual component with respect to their specification.

Deliverable 1.5 "Ethics and Privacy protection plan" and its reports 1.6 and 1.10 addresses the risk management. Even if the risks are identified from an ethical perspective, they still justify the need for testing. In this context, the first release of WP3 and WP4 tools, APIs, modules and bridges (called "α release", as defined in the lifecycle described in Deliverable D5.1) is put under stress testing in a field lab; as a result of fixing problems encountered in this phase, a second release ("β release") will be produced, which will go for DS exploitation. It aims mainly to provide evidence that AoTIES and all related APIs and Modules are really mature and stable and that the bridges between the platforms are smoothly working.

The Stability & Reliability Tests will be detailed to the following sub-phases:

- **Code validation (Unit test)**
  Through the usage of the unit tests, a set of automatic code validation and reliability will be generated (note that stress, stability and reliability tests are not yet reported).

- **Functional validation (Lab test \ bed scenarios)**
  Once the code is validated the Latbest\bed test aims to validate the expected layer, bridge and tools functionalities. Note that Labtesting is done with real users (invited for the purpose)

- **Stress test:**
  While remaining in a Lab environment, the stress test aims to expose the AoTIES and the related tools to harder conditions, mainly to validate and perform their stability. It will include putting it under stress phase for longer time, with a higher number of iteration, devices, tools and simultaneous connected platforms.

Note that the deployment tools are not evaluated in this deliverable.
3 Verification

This section compiles the results obtained by testing the AIoTES. It describes in detail the outcome of tests performed in lab and during the initial evaluation of the first deployments.

Note that the SIL is composed by two independent components that have different functions: the Interoperability Layer, that integrates the bridges, and the IPSM that employs alignment files to perform semantic translations. Both bridges and alignments were developed in ACTIVAGE order to include new platforms from DS.

3.1 Semantic Interoperability Layer

3.1.1 SIL Testing facilities

The testing of the Semantic Interoperability Layer consists of a set of In-lab tests that allow to test the correct performance of the SIL at different levels from the individual functions of a bridge to the whole SIL and platforms deployment. While the first tests allow to check the correct standalone behavior of the bridges, other more advanced tests make possible to check the correct functioning of the SIL as a whole.

The In-lab tests can be classified in the following categories:

- Bridge unit tests
- Message testing for semantic translation
- Integration tests
- Testing environment

The next paragraphs describe the different tests performed on the SIL.

3.1.2 Bridge unit tests

The first step in the development and testing of a new IoT platform bridge consists of the creation of JUnit test classes for bridge standalone testing. These unit tests make use of a real or simulated platform and the necessary classes from the SIL. The tests can be defined specifically for each bridge and allow testing different functionalities separately (communication with the IoT platform, syntactic translation, etc).

Since each platform has specific requirements, the tests to be performed when testing the bridges can vary greatly and should be defined by the bridge developer. However, the unit tests must at least ensure the correct functioning of the following common operations:

- registerPlatform: Registers the platform in the SIL and creates a bridge instance for this platform.
- unregisterPlatform: Unregisters the platform.
- subscribe: Subscribes client to observations provided by the platform. Creates a listener for sensor data and sends the received observations upstream in the common JSON-LD format.
- unsubscribe: Cancels specified subscription created by the Subscribe.
- platformCreateDevices: An instruction for the platform to create new virtual devices.
- platformDeleteDevices: An instruction for a platform to remove a virtual device.
– **observe**: Pushes a given observation message from SIL to platform (bridge acting as a publisher for platform)

These common tests include a set of predefined messages to call to the different methods of the bridge emulating SIL’s requests.

```json
{
  "@graph": [
    {
      "@id": "InterIoTMsg:meta/eddec4d4-4269-455c-983d-f0eace2bef3c",
      "@type": ["InterIoTMsg:meta", "InterIoTMsg:Platform_register" ],
      "InterIoTMsg:ReceiverPlatformId": {
        "@id": "InterIoT:example-platform1"
      },
      "InterIoTMsg:conversationID": "conv506c38b5-4ca4-47c3-9608-1a69f3ca9fba",
      "InterIoTMsg:dateTimeStamp": "2018-01-15T21:48:05.735+01:00",
      "InterIoTMsg:messageID": "msg06494427-76bb-4ade-8be7-2daaa0d61101",
      "InterIoTMsg:status": "myclient"
    }
  ],
  "@id": "InterIoTMsg:metadata"
},
{
  "@graph": [
    {
      "@id": "_b0",
      "@type": ["InterIoT:GOIoTP#Middleware", "InterIoT:GOIoTP#sofia2" ],
      "InterIoT:GOIoTP#hasBaseEndpoint": "http://localhost:4569/
    }
  ],
  "@id": "InterIoT:example-platform1",
  "@type": ["http://www.w3.org/ns/sosa/Platform", "InterIoT:GOIoTP#SoftwarePlatform" ],
  "InterIoT:GOIoTP#hasMiddleware": {
    "@id": "_b0"
  },
  "InterIoT:GOIoTP#hasName": "Example platform"
},
"@id": "InterIoTMsg:payload"
]
"@context": {
  "InterIoTMsg": "http://inter-iot.eu/message/",
  "InterIoT": "http://inter-iot.eu/"
}
}
```

**Figure 3: Unit test message example**

Also, the syntactic translation of the data between the JSON-LD format of the SIL and the platform’s format should be checked using unit tests.
3.1.3 Semantic Unit Test — (Messages, Semantic translation & Alignments)

The objective of the semantic unit tests is to check that the semantic translations are performed correctly. Semantic translations are performed after the syntactic translation process of the bridges. Messages are sent to the IPSM component (Inter Platform Semantic Mediator), and semantic conversion is performed following the rules for translation set on the alignment files.

These tests verify the definition of the semantic alignments and need data in the JSON-LD format of the SIL as an input. These messages are provided by the bridge. The semantic unit tests cover the different types of translations that will be performed (from platform ontology to central ontology and vice versa) and the different messages that will be translated (observations, device definitions, etc.). In order to test that messages are well-formed, and the semantic alignments for performing semantic translations are correct, it has been created a webtool for testing. This webtool is connected to a real instance of IPSM running in one server. In this way, through the webtool interface it is possible to configure an ontology translation, selecting the alignments and preparing a channel for this task. Then, it is possible to perform in an easy and direct way semantic translations of messages executing it through this interface. In this way it makes it possible to test alignments, the quality of the semantic translation and/or messages format.

[Image: Figure 4: Syntactic translation test]
IPSM REST Translation

Translation setup

Available alignments

- select alignment -

Add alignment to translation sequence

Translation sequence:

Figure 5: Semantic translation configuration

Figure 6: Semantic translation test
Also, it allows the monitoring of incoming flows of messages to test the correct performance and velocity of the semantic translator.

**Figure 7: Semantic translation monitoring**

### 3.1.4 Integration tests

The next step are the integration tests, which have been designed to check if the bridge can be integrated in the SIL. These tests are defined as JUnit classes that use the Interoperability Layer Java API to execute a standard scenario and check if the operations (register platform, create devices, subscribe to specific devices, etc.) have been executed correctly. The integration tests must ensure the correct operation of all the functions of the bridge when they are called by the SIL and check that the bridge generates the proper messages in each case.

These tests make use of a real or simulated platform together with the following components of the SIL:

- RabbitMQ
- Parliament triple store
- Inter-Platform Semantic Mediator (IPSM)

The IPSM should be configured to use the defined semantic alignments in order to check the operation of the bridge together with the semantic translation.
Figure 8: Integration test example

Figure 9: Integration test example: devices registration
3.1.5 Testing environment

A testing environment that simulates the use of the SIL in a real scenario has been defined. This scenario includes a full deployment of the SIL in a server. This deployment of the SIL is used along with real platforms with either real or simulated devices.

The different functions of the SIL are called making use of its REST API. The requests sent to the SIL API are converted into messages, which are routed by inter-MW. If the action involves a platform, the message is sent to the bridge, which translates it into the corresponding request to the platform. Then, a response message containing the result of the operation is generated by the bridge and sent to inter-MW. The IPSM should be configured with the corresponding semantic alignment to be able to perform the necessary semantic translation operations on the upstream and downstream messages.
3.1.6 IoTivity bridge testing

In this section, an example of bridge testing is presented for the specific case of the IoT platform IoTivity. In this section the testing process is described for this specific case, and it can be seen in more detail.

The testing of the IoTivity bridge consists of a set of different tests that allow the complete from end-to-end verification of the application and ensure that the application is fully functional. The implemented tests are classified in the following categories:

- Bridge unit tests
- Integration tests
- Testing environment

3.1.6.1 Bridge unit tests

Unit testing is a software testing method where individual units/components of a software are tested. The purpose is to validate that each unit of the software performs as designed. Before presenting in detail the unit tests that have been implemented for the IoTivity bridge, the next paragraphs explain how the bridge works and which are its main modules, shown in Figure 12:

![IoTivity bridge architecture](image)

The IoTivity platform supports information exchange and control based on messaging through the Constrained Application Protocol (CoAP). The messages are in CBOR format, which is a binary data serialization format loosely based on JSON, and follow the Open Connectivity Foundation’s (OCF) specifications. The IoTivity bridge utilizes 2 main modules: the CoAP client which is for communicating with the IoTivity server and the Syntactic translator which is responsible for the syntactic translation of the AIoTES format to IoTivity format and vice versa.

When an HTTP request is received by the bridge, the Syntactic translator translates the request to IoTivity format and the corresponding CoAP request is sent to the IoTivity server through the CoAP client. The response received by the IoTivity server is then translated to AIoTES format by the syntactic translator and the corresponding HTTP response is sent back by the bridge. The data flow of a received request along with how these two components cooperate is displayed in the figure above.

The testing of these components is made by unit testing, their basic functionalities and the framework adopted for these test is JUnit. For the CoAP Client the following functionalities are tested:

- DISCOVER: the client makes a DISCOVERY unicast request and discovers the port assigned to the IoTivity server along with all registered resources
D5.3 Intermediate Validation Results

- GET: the client makes a GET CoAP request to a specific resource and receives a response with the representation of the resource
- PUT: the client makes a PUT CoAP request and updates a registered resource
- POST: the client makes a POST CoAP request and creates a new resource to the server
- DELETE: the client makes a DELETE CoAP request and removes a resource from the server
- OBSERVE: the client makes an OBSERVE request and gets notified for any change related to a resource
- QUERY: the client makes a query request and retrieves all resources that match this query

A simple server example provided by IoTivity is used for these tests ([https://github.com/iotivity/iotivity/blob/master/java/examples-java/simpleclientserver/src/main/java/org/iotivity/base/examples/SimpleServer.java](https://github.com/iotivity/iotivity/blob/master/java/examples-java/simpleclientserver/src/main/java/org/iotivity/base/examples/SimpleServer.java)). For each functionality, a predefined set of files are used as input and the response received by the server is compared to the corresponding expected outcome. Figure 13 displays an example of successfully running JUnit tests. Each one of the tests of the figure verifies one of the aforementioned functionalities.

![JUnit test results](image)

**Figure 13: JUnit tests for the CoAP client module**

Regarding the Syntactic translator component, there are two main methods that are tested:

- toJenaModel: this is a method that translates an IoTivity message to AIoTES message
- toFormatX: this is a method that translates an AIoTES message to IoTivity message

The input files that are used simulate different type of messages that are usually translated according to the supported functionalities of the bridge (e.g. registerPlatform, unregisterPlatform, subscribe etc.) while, methods are asserted by comparing each time the actual outcome with the expected. The following table demonstrates an example of input and expecting output for testing the method “toJenaModel”.

<table>
<thead>
<tr>
<th>Input Message</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>registerPlatform</td>
<td>@Platform</td>
</tr>
</tbody>
</table>
Table 1: Input and expected outcome examples for testing the method ‘toJenaModel’

<table>
<thead>
<tr>
<th>Input example (IoTivity format)</th>
<th>Expected outcome (AloTES format)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>&quot;rt&quot;: [&quot;oic.wk.d&quot;],</td>
<td>&quot;@graph&quot; : [ {</td>
</tr>
<tr>
<td>&quot;if&quot;: [&quot;oic.if.ll&quot;, &quot;oic.if.r&quot;,</td>
<td>&quot;@id&quot; : &quot;_:b0&quot;,</td>
</tr>
<tr>
<td>&quot;oic.if.baseline&quot;],</td>
<td>&quot;@type&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td>&quot;id&quot;: &quot;device id&quot;,</td>
<td>iot.eu/syntax/Iotivity.owl#BloodPressure&quot;,</td>
</tr>
<tr>
<td>&quot;href&quot;: &quot;/mybloodpressure&quot;,</td>
<td>&quot;diastolic&quot;: &quot;80&quot;,</td>
</tr>
<tr>
<td>&quot;links&quot; : [</td>
<td>&quot;systolic&quot;: &quot;120&quot;,</td>
</tr>
<tr>
<td>&quot;href&quot;: &quot;/mybloodpressure/bloodpressure&quot;,</td>
<td>&quot;units&quot;: &quot;mmHg&quot;</td>
</tr>
<tr>
<td>&quot;oic.r.blood.pressure&quot;],</td>
<td>},</td>
</tr>
<tr>
<td>&quot;if&quot;: [&quot;oic.if.s&quot;,</td>
<td>&quot;@id&quot; : &quot;_:b1&quot;,</td>
</tr>
<tr>
<td>&quot;oic.if.baseline&quot;],</td>
<td>&quot;@type&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td>&quot;rep&quot; : {</td>
<td>iot.eu/syntax/Iotivity.owl#PulseRate&quot;,</td>
</tr>
<tr>
<td>&quot;systolic&quot;: 120,</td>
<td>&quot;pulserate&quot;: &quot;70&quot;</td>
</tr>
<tr>
<td>&quot;diastolic&quot;: 80,</td>
<td>},</td>
</tr>
<tr>
<td>&quot;units&quot;: &quot;mmHg&quot;</td>
<td>&quot;@id&quot; : &quot;_:b2&quot;,</td>
</tr>
<tr>
<td>} ],</td>
<td>&quot;@type&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td>{</td>
<td>iot.eu/syntax/Iotivity.owl#Glucose&quot;,</td>
</tr>
<tr>
<td>&quot;href&quot;: &quot;/mybloodpressure/pulse&quot;,</td>
<td>&quot;glucose&quot;: &quot;95&quot;,</td>
</tr>
<tr>
<td>&quot;rt&quot;: [&quot;oic.r.pulserate&quot;],</td>
<td>&quot;units&quot;: &quot;mg/dL&quot;</td>
</tr>
<tr>
<td>&quot;if&quot;: [&quot;oic.if.s&quot;,</td>
<td>},</td>
</tr>
<tr>
<td>&quot;oic.if.baseline&quot;],</td>
<td>&quot;@id&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td>&quot;rep&quot; : {</td>
<td>iot.eu/syntax/Iotivity#Instance/device id&quot;,</td>
</tr>
<tr>
<td>&quot;pulserate&quot;: 70</td>
<td>&quot;hasResource&quot; : [ &quot;_:b2&quot;,</td>
</tr>
<tr>
<td>} ],</td>
<td>&quot;<em>:b1&quot;, &quot;</em>:b0&quot; ],</td>
</tr>
<tr>
<td>{</td>
<td>&quot;<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td>&quot;href&quot;: &quot;/mybloodpressure/glucose&quot;,</td>
<td>iot.eu/syntax/Iotivity.owl#Device&quot;</td>
</tr>
<tr>
<td>&quot;rt&quot;: [&quot;oic.r.glucose&quot;],</td>
<td>} ],</td>
</tr>
<tr>
<td>&quot;if&quot;: [&quot;oic.if.s&quot;,</td>
<td>&quot;@context&quot; : {</td>
</tr>
<tr>
<td>&quot;oic.if.baseline&quot;],</td>
<td>&quot;units&quot; : {</td>
</tr>
<tr>
<td>&quot;rep&quot; : {</td>
<td>&quot;@id&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td>&quot;glucose&quot;: 95,</td>
<td>iot.eu/syntax/Iotivity#units&quot;</td>
</tr>
<tr>
<td>&quot;units&quot;: &quot;mg/dL&quot;</td>
<td>&quot;diastolic&quot; : {</td>
</tr>
<tr>
<td>} }</td>
<td>&quot;@id&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td></td>
<td>iot.eu/syntax/Iotivity#diastolic&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;@type&quot; : &quot;<a href="http://www.w3.org/2001/XMLSchema#long">http://www.w3.org/2001/XMLSchema#long</a>&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;systolic&quot; : {</td>
</tr>
<tr>
<td></td>
<td>&quot;@id&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td></td>
<td>iot.eu/syntax/Iotivity#systolic&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;@type&quot; : &quot;<a href="http://www.w3.org/2001/XMLSchema#long">http://www.w3.org/2001/XMLSchema#long</a>&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;pulserate&quot; : {</td>
</tr>
<tr>
<td></td>
<td>&quot;@id&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td></td>
<td>iot.eu/syntax/Iotivity#pulserate&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;@type&quot; : &quot;<a href="http://www.w3.org/2001/XMLSchema#long">http://www.w3.org/2001/XMLSchema#long</a>&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;glucose&quot; : {</td>
</tr>
<tr>
<td></td>
<td>&quot;@id&quot; : &quot;<a href="http://inter-">http://inter-</a></td>
</tr>
<tr>
<td></td>
<td>iot.eu/syntax/Iotivity#glucose&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;@type&quot; : &quot;<a href="http://www.w3.org/2001/XMLSchema#long">http://www.w3.org/2001/XMLSchema#long</a>&quot;</td>
</tr>
</tbody>
</table>

© ACTIVAGE
3.1.6.2 Integration tests

Integration testing is a software testing method where individual software modules are combined and tested as a group. In our case this involves the testing of the bridge as a whole and the verification that all modules co-operate properly.

A series of JUnit classes have been implemented where an IoTivity bridge emulator is used. Each supported functionality of the bridge is tested by creating HTTP requests with pre-defined payload. The bridge utilizes the Syntactic translator and the CoAP client modules, makes the corresponding request to the IoTivity server (a dedicated server used only for testing purposes) and returns an HTTP response in AIoTES format. The payload of the response is compared with the corresponding expected outcome. The following table displays all available test and the steps included in each one of them.

Table 2: Description of available tests and their steps

<table>
<thead>
<tr>
<th>Available tests</th>
<th>Steps for testing</th>
</tr>
</thead>
</table>
| Register platform                                      | 1) Send a platform registration message  
2) Check if the received response is correct                      |
| Unregister platform                                     | 1) Send a message for unregistering a platform  
2) Check if the received response is correct                      |
| Query all devices (no registered devices on the server) | 1) Send a “query” message with no device id specified  
2) Check if the received response contains a void list of devices          |
| Create device / Query device                           | 1) Send a message that creates a device in the IoTivity server  
2) Check if the received response is OK  
3) Send a “query” message that includes the id of the device that it was created before  
4) Check if the received response contains a device identical to the one created |
<p>| Query all devices (multiple devices are registered on the server) | 1) Send two messages that create corresponding devices in the IoTivity server |</p>
<table>
<thead>
<tr>
<th>Action</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check if responses</td>
<td>2) Check if the received responses are OK</td>
</tr>
<tr>
<td>are OK</td>
<td>3) Send a “query” message with no device id specified</td>
</tr>
<tr>
<td></td>
<td>4) Check if the received response contains a list with devices identical to those created in step 1</td>
</tr>
<tr>
<td>Delete device</td>
<td>1) Send a message that creates a device in the IoTivity server</td>
</tr>
<tr>
<td></td>
<td>2) Check if the received response is OK</td>
</tr>
<tr>
<td></td>
<td>3) Send a message that deletes the device that was just created</td>
</tr>
<tr>
<td></td>
<td>4) Check if the received response is OK</td>
</tr>
<tr>
<td></td>
<td>5) Send a query message that lists all devices</td>
</tr>
<tr>
<td></td>
<td>6) Check if the received response contains a void list of devices</td>
</tr>
<tr>
<td>Update device</td>
<td>1) Send a message that creates a device in the IoTivity server</td>
</tr>
<tr>
<td></td>
<td>2) Check if the received response is OK</td>
</tr>
<tr>
<td></td>
<td>3) Send a message that updates the device that was just created or one of its measurements</td>
</tr>
<tr>
<td></td>
<td>4) Check if the received response is OK</td>
</tr>
<tr>
<td></td>
<td>5) Send a query message that includes the id of the device that it was created before</td>
</tr>
<tr>
<td></td>
<td>6) Check if the device included in the response is updated according to the corresponding request made in the previous step</td>
</tr>
<tr>
<td>Subscribe device</td>
<td>1) Send a message that creates a device in the IoTivity server</td>
</tr>
<tr>
<td></td>
<td>2) Check if the received response is OK</td>
</tr>
<tr>
<td></td>
<td>3) Send a subscription request for this device</td>
</tr>
<tr>
<td></td>
<td>4) Send a message that updates one of the measurements of this device</td>
</tr>
<tr>
<td></td>
<td>5) Check if the received response is OK</td>
</tr>
<tr>
<td></td>
<td>6) Check if the subscriber received the corresponding measurement</td>
</tr>
<tr>
<td>Unsubscribe from</td>
<td>1) Send a message that creates a device in the IoTivity server</td>
</tr>
<tr>
<td>device</td>
<td></td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

<table>
<thead>
<tr>
<th>Number</th>
<th>Task Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2)</td>
<td>Check if the received response is OK</td>
</tr>
<tr>
<td>3)</td>
<td>Send a subscription request for this device</td>
</tr>
<tr>
<td>4)</td>
<td>Unsubscribe from the device</td>
</tr>
<tr>
<td>5)</td>
<td>Send a message that updates one of the measurements of this device</td>
</tr>
<tr>
<td>6)</td>
<td>Check if the received response is OK</td>
</tr>
<tr>
<td>7)</td>
<td>Check if the subscriber received no measurement</td>
</tr>
</tbody>
</table>

List devices

- Send a “list devices” request
- Check if the received response is OK
3) Send a message that creates a device in the IoTivity server
4) Check if the received response is OK
5) Check if a notification has been received indicating the creation of a device

The following table demonstrates an input that is used for testing the creation of a device. Similar inputs are used for testing all aforementioned functionalities.

**Table 3: Input example for integration testing**

```json
{
    "@graph": [ {
        "@id": "InterIoTMsg:meta/8fc4a7ed-d48c-44b0-8ae2-186ad24d88f4",
        "@type": [ "InterIoTMsg:meta", "InterIoTMsg:Platform_create_Device" ],
        "InterIoTMsg:ReceiverPlatformId": {
            "@id": "InterIoT:IoTivity"
        },
        "InterIoTMsg:clientID": "myclient",
        "InterIoTMsg:conversationID": "conv66c69530-3fbb-4bf1-ac07-d94656ff1178",
        "InterIoTMsg:dateTimeStamp": "2018-07-09T08:23:20.657Z",
        "InterIoTMsg:messageID": "msgffd95f48-f4de-4e36-adbe-a21dc4517214"
    },
    "@id": "InterIoTMsg:metadata"
}, {
    "@graph": [ {
        "@id": "_:b0",
        "@type": "http://inter-iot.eu/syntax/Iotivity.owl#PresenceSensor",
        "value": true
    },
    "@id": "_:b1",
    "@type": "http://inter-iot.eu/syntax/Iotivity.owl#Temperature",
    "temperature": 32,
    "temperatureUnits": "C"
    }
```
The above example is representative of all other functions.

3.1.6.3 Testing environment

The JUnit testing framework has been used for both unit and integration testing. It is a well-known Regression Testing Framework used by developers to implement unit testing in Java and it is one of a family of unit testing frameworks collectively known as xUnit. Each test is a set of assertions verifying that the results of a method are expected either in their type and content. JUnit test framework provides the following important features:

- **Fixtures** is a fixed state of a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that there is a well-known and fixed environment in which tests are run so that results are repeatable.
- **Test suites** bundle a few unit test cases and runs them together.
- **Test runners**, which are used for executing the test cases.

JUnit classes, which are important classes, used in writing and testing JUnits.
3.1.7 SIL API

The SIL exposes an API to externally access to its functionality which is already documented in D3.7. It has been performed in the frame of T5.2 tests on the SIL API for testing main operations. This operations are encompassed in the AloTES API, as it can be seen in section 3.7. This testing represents also a preliminary testing of AloTES API functions.

It has been performed a functional testing on main SIL API operations, which can be seen in Appendix A. Functional tests check that given a set of inputs to the API the expected outputs are received. The following images show the results of inputs to the API for testing specific operation. These images are complemented with prior information about the specific API operation. All operational inputs returned the expected outputs, thus the correct performance of the SIL API is demonstrated through these functional tests.

3.2 AloTES Management

The AloTES Management tool is an ACTIVAGE global dashboard application providing monitoring features to European stakeholders. The specified features are:

- IoT Platforms information, about their status, location of DS where they are deployed, their IP address.
- Live and historical data, providing information about the performance using several deployments: key performance indicators, and deployed resources
- A feature to manage user rights to access the ACTIVAGE dashboard.

This section reports on the results obtained by DSs with respect to technical Key Point Indicators (KPIs) employed for management and DS validation.

These technical KPIs reflect the level of achievement of the AloTES with respect to the expectations of DSs, since the KPIs are the translation of the DS requirements into technical terms (see also ACTIVAGE project Deliverable D5.1 “Integration plan and operational framework”). They also give a rough mean for the evaluation of interoperability. Every subsection also briefly explains how to compute the technical KPIs. Note that, although it seems an apparently trivial task, it proves difficult for the DSs to perform such tests since they have to implement the KPI computation or restitution.

A tool will be available for gathering the results of the computations in all DSs, thus collecting the DS KPIs. This tool, the AloTES Management, is a part of the AloTES framework and it is currently implemented and tested in the DS 6 in Isère.

Table 4 lists the KPIs monitored on DS 6 with Sensinact (excluding Interoperability, scalability, security and privacy). The table also shows the pending tests and KPI implementations. The table below first column lists the technical KPIs, by category. In the target column, the availability of the KPI target is given for each KPI:

- ‘TODO’ means value is not yet available, implementation is under progress
- A constant value means the target is known but not provided in the tool API (for instance 100% for a perfect rate)
- ‘API’ means the target value is provided through the AloTES management API
- ‘Computable’ means the target is not directly provided as constant or through the API, but can be computed from provided targets (for instance #installed devices/beneficiary target can be computed having total number of devices and total number of beneficiaries targets)
D5.3 Intermediate Validation Results

- ‘?’ means the way to implement the given KPI is not specified yet

In the live and history columns, the availability of the KPI live value for monitoring and history value for statistics are given for each KPI. Similarly to the target column, the monitor/statistics data can be in todo state, available as constant or through the API, computable and not yet specified.

Appendix B shows the results of tests done in DS6 Isère to check if information related with each sensor is detected by the DS IoT platform. Part of this testing has been done manually and part of it it has been performed through the AloTES management tool.
## D5.3 Intermediate Validation Results

### Table 4: KPIs used on DS 6 with sensINact

<table>
<thead>
<tr>
<th></th>
<th>DS6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Target</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td></td>
</tr>
<tr>
<td>Deployment degree:</td>
<td></td>
</tr>
<tr>
<td># installed elderly beneficiaries</td>
<td>API</td>
</tr>
<tr>
<td>% rate installed/planned elderly beneficiaries</td>
<td>100%</td>
</tr>
<tr>
<td># installed devices</td>
<td>API</td>
</tr>
<tr>
<td>% rate installed/planned devices</td>
<td>100%</td>
</tr>
<tr>
<td># installed devices / beneficiary</td>
<td>computable</td>
</tr>
<tr>
<td><strong>Uptime/SLA fulfilment:</strong></td>
<td>?</td>
</tr>
<tr>
<td><strong>Connectivity/availability rate:</strong></td>
<td></td>
</tr>
<tr>
<td># connected beneficiary gateways</td>
<td>API</td>
</tr>
<tr>
<td>% rate connected/installed gateways</td>
<td>API</td>
</tr>
<tr>
<td># connected devices</td>
<td>API</td>
</tr>
<tr>
<td>% rate connected/installed devices</td>
<td>100%</td>
</tr>
<tr>
<td>cumulated disconnection durations</td>
<td>TODO</td>
</tr>
<tr>
<td>% rate connection durations/#devices/installation duration</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Use (of API)</strong></td>
<td></td>
</tr>
<tr>
<td>API average usage</td>
<td></td>
</tr>
<tr>
<td># available functions</td>
<td>API</td>
</tr>
<tr>
<td># active functions</td>
<td>API</td>
</tr>
<tr>
<td># function deactivations</td>
<td>API</td>
</tr>
<tr>
<td>% rate deactivated/available functions</td>
<td>0%</td>
</tr>
</tbody>
</table>
cumulated durations of function deactivations
% rate deactivation duration/installation duration

Peak usage:
  peak used/ available infrastructure?

Frequency:
  # API requests
distribution of requests frequency

Throughput/bandwidth:
  # transactions
# transactions/sampling duration

Size of rough database:
  raw database size (in byte)
# sensor records

Alerts and notifications:
  # registered triggered alerts
# registered false alerts
% rate false/triggered alerts
# registered triggered notifications

AloTES use
  # AloTES instances installed (livinglab instance)
# interoperability use cases running (WOQ bed sensor in livinglab through universaal
  # AloTES tools used

<table>
<thead>
<tr>
<th></th>
<th>API</th>
<th>API</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td></td>
<td>computable</td>
<td>computable</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
<td></td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
<td></td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>API</td>
<td>API</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>API</td>
<td>API</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>API</td>
<td>API</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>API</td>
<td>API</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

Sparsity of historical data

Average storage frequency:
- total number of records / number of sensors / history sampling duration

Distribution of storage frequencies by sensors:
- mean frequency and standard deviation

Distribution of storage frequency by sensor types:
- mean frequency and standard deviation

User interface

Applications
- # available applications
- # installed applications
- % rate installed/planned applications
- # clicks in apps (to quantify level of use)

UI use
- # run tasks or AHA functions
- # completed tasks
- % rate completed/run tasks

Adaptation of the UI to the type of user (patient, medic, technician, etc.):
- # app/ui dedicated to elderly beneficiary
- # app/ui dedicated to medic staff
- # app/ui dedicated to technician
- # app/ui dedicated to other users

Adaptation of the UI to the type of device (input, rendering, modes, medias, etc.):

<table>
<thead>
<tr>
<th>API</th>
<th>computable</th>
<th>computable</th>
</tr>
</thead>
<tbody>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>API</th>
<th>API</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>API</th>
<th>API</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
</tbody>
</table>

| 2 | 2 | NO |
| 3 | 2 | NO |
| 2 | 1 | NO |
| 2 | 2 | NO |
D5.3 Intermediate Validation Results

# ui dedicated to input
# ui dedicated to rendering
# ui dedicated to modes
# ui dedicated to media
# ui dedicated to other type of device

Specialization of the UI to the task or AHA functions:
- # specified AHA functions
- # delivered AHA functions through apps
- % rate ui delivered/specified AHA functions

Support effectiveness
Resolution
- # declared issues
- # solved issues
- % rate solved/declared issues

Responsiveness:
- # issues attended
- % rate attended/declared issues

Power consumption
- # powered devices
- cumulated power consumption of powered devices
- average power / beneficiary

System performance (Time)
Response time:
- Average time to complete a request

<table>
<thead>
<tr>
<th></th>
<th>Specified AHA functions</th>
<th>Delivered AHA functions</th>
<th>% Rate UI Delivered/Specified AHA Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td># specified AHA</td>
<td>68</td>
<td>68</td>
<td>NO</td>
</tr>
<tr>
<td># delivered AHA</td>
<td>68</td>
<td>36</td>
<td>NO</td>
</tr>
<tr>
<td>% rate UI delivered</td>
<td>100</td>
<td>53%</td>
<td>NO</td>
</tr>
</tbody>
</table>

Version 1.0 | 2019-02-14 | ACTIVAGE ©
### Latency:
- Average packet delivery time

### Jitter:
- Average packet delivery time variation

### Discovery Time:
- Average time to discover a new system/resource

### Timeliness/fulfilment of real-time constraints:
- #met deadlines/#missed deadlines

### Autonomy:
- # discharged devices
- # running out of charge devices
- % discharged/installed devices
- % running out of charge/installed devices
- average time working without the need to recharge/change battery

### Lifetime-function properly:
- # replaced devices
- % replaced/installed devices
- Average lifetime without the need to replace the device

### Reliability
- Failure probability (error, loss, false alert rate):
  - Time failure state/Total time
  - #error/Total operations

### Mean Time between Failures:

<table>
<thead>
<tr>
<th>TODO</th>
<th>TODO</th>
<th>TODO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>API</td>
<td>API</td>
<td>API</td>
</tr>
<tr>
<td>API</td>
<td>API</td>
<td>API</td>
</tr>
<tr>
<td>0%</td>
<td>computable</td>
<td>computable</td>
</tr>
<tr>
<td>? 10%</td>
<td>computable</td>
<td>computable</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

Average operational time between failures

Mean Time to Repair:
Average time to return to production status

<table>
<thead>
<tr>
<th>TODO</th>
<th>TODO</th>
<th>TODO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TODO</td>
<td>TODO</td>
<td>TODO</td>
</tr>
</tbody>
</table>
3.3 Developments tools

The ACTIVAGE development tools allow the composition of existing applications and tools, in order to easily generate new applications. They are described in detail in deliverable D4.1 "Developers toolkit and deployment support".

Note that some of the testing performed in this section is presented in a brief and partially summarized manner, for the sake of clarity to the reader, as it represents a very long list of tests of many small components. Complex testing procedures and results are summarized providing a high level analysis, and the corresponding tools are described at the final part of this section.

3.3.1 ACTIVAGE Ontology Explorer

The ACTIVAGE Ontology Explorer is a web application used for visualizing the ACTIVAGE Ontologies. The application is verified through scenario testing, where realistic user activities that cover the most important components of the system are used as scenario tests. The following tables demonstrate these tests, along with the needed actions per test. The tests include: Navigate using tabs, Select a class, Search for a class, Select an annotation property, Select an object property, Select a data property, Select an instance and Graph navigation.

The appendix D.1 gives details about the tests.

3.3.2 Device Semantics Editor

The Device Semantics Editor is a web application used for defining/editing the semantics related to the devices. For its verification, the following scenario tests have been implemented.

The tests include: Create a new device sub-class, Delete a device system, Delete a device system, Add/edit/delete annotation property, Add/edit/delete annotation restriction, Add/edit/delete datatype property and Add/edit/delete object property.

Appendix D.2 gives details about the tests.

3.3.3 Service Semantics Editor

The Service Semantics Editor is a tool closely related to the Device Semantics Editor. Specifically, both applications support the same functionalities and similar user interface. They are both used for specifying or editing semantics but they target different group of classes. Therefore, the testing scenarios that were designed for the Device Semantics Editor and are demonstrated in section 3.3.2 are also utilised for the verification of this tool.

3.3.4 Data Analyser

The Data analyser is a web application for experimenting with data analytics methods on a loaded dataset. The data analytics methods that are utilized by this tool are part of the Analytics and their verification is described in detail in section 3.5. For the application and its main functionalities, a series of different scenarios has been developed in order to verify them. These scenarios include several steps and for each step there is a detailed description of the action needed and of the expected result. The aforementioned scenarios are demonstrated in the following tables.

The tests include Load data, Raw data table functionalities, Run analysis and Export analysis results.
Appendix D.3 gives details about the tests.

### 3.3.5 DataModel 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.

Appendix D.4 reports the tests that have been applied to the DataModel Workbench tool. For the DataModel Workbench tool, we have unit tests and integration tests (with Database tests, Table tests, Schema tests and Query tests), and we intend to continue testing as the development continues. The appendix also contains the inlab validation results. To this date all tests are passing, more tests will be added in case we will need to have more coverage to test updated and/or new features being developed.

### 3.3.6 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 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.

Appendix D.5 reports the tests that have been applied to the Metadata Storage Exploder tool. For the Metadata Storage Exploder tool, we have unit tests and integration tests (with Model tests) and we intend to continue testing as the development continues. The appendix also contains the inlab validation results. To this date all tests are passing, more tests will be added in case we will need to have more coverage to test updated and/or new features being developed.

### 3.3.7 AloTES IDE tool

AloTES IDE tool is an Integrated Development Environment for development in the frame of the ACTIVAGE ecosystem. This tool integrates 4 different AloTES development tools: Service Composer, ClickDigital, Code Generator and Code Templates. Also, it provides a guidance for their use for developers, as it includes an integrated wizard.

Regarding the testing procedures applied:

- Integration tests have been performed with the development tools that are encompassed within the IDE.
- Each tool has performed unitary tests on it (when applicable) separately.
• It has been performed functional testing to check that its performance under determined inputs is as expected.

### 3.3.8 AIoTES SIL tool

The AIoTES SIL tool is a development tool that eases the use of basic interactions with instances of the Semantic Interoperability Layer, and in addition represents a didactic tool that introduces the first steps on the use of the SIL API. This tool is composed by two independent parts: it has been developed a front-end and a back-end. Both parts of the tool have been tested separately with functional tests in order to see that the response to REST operations is correct. Since the back end communicates with the SIL, an integration test has been performed to verify the integration with the SIL. Finally a functional integration test on the 2 complementary parts of the tool was performed. Compatibility with different browsers has been tested as well, as this tool requires the use of a web browser for its visualization.

### 3.3.9 Service Composer

The service composition tool is responsible for the creation of composite services. Certain steps (basic functional testing) have to be followed to verify its correct operation, as well as the correct operation of the flow of services designed and executed through this tool:

- Validation of the correct access to services. The configuration of nodes should provide information about the connectivity of the service, in order to check if it is available.
- The nodes are responsible of wrapping the functionalities of the services. Therefore it is necessary to test that in addition to being able to access to these services, they are capable of executing the expected functions. Test with sample data provide feedback about the correct status of the functionalities.
- The nodes as wrappers of the services, introduce some inputs and give some outputs. It is necessary to validate that the messages are the expected ones.
- Access to the node with real data. Test the correct operation. Fix bugs and catch errors.
- The flows represent the interaction between nodes. It is necessary to validate if the message between services passing is working in the expected way. Testing the runtime.

### 3.3.10 Code generation tool

This section reports the tests that have been applied to the code generation tool, see Section 4.1.4.1 of D4.1. For the Code generation tool, we have only unit tests, and we intend to continue testing as the development continues. This testing will include usability testing. Note that most of these tests are automated.

Automatic unit tests for Code generation:

- XML validation (3 tests)
  - malformed XML
  - inconsistent XML
  - valid XML
- Macro validation (1 test)
- Variable management (3 test)
  - missing variables
  - non-defined variables
D5.3 Intermediate Validation Results

- Correct variables
- Ontology management (5 tests)
  - Inconsistent imports
  - Referenced missing imports
  - Missing remote ontology reference
  - Correct remote ontology reference
  - Correct local ontology reference
- Full test/integration test (5 tests) complete process.

Additionally the 3 presentations of the tool have been tested. The protégé plugin has been manually tested and validated; The Maven plugin has one integration tests, and has been manually tested and validated; The REST service has been manually tested and validated.

To this date all unit tests are passing, more unit test will be added to have more coverage, and to test new features being developed.

3.3.11 Wiki

The 2 Wikis in ACTIVAGE (for internal and external use respectively) consist on 2 accessible online services that display information about the project. Due to the nature of this tool, it does not proceed to perform testing procedures on them. Hosting services for Wiki content are already provided by an instance of Gogs GIT and an instance of Confluence, and they have proven to be reliable.

3.4 Data Lake

The Data Lake is responsible for providing access to raw data and analytics metadata that are needed for high-level applications, such as data/visual analytics services and development/deployment tools. The basic components of the Data Lake, as described in deliverable D4.5 "Data Layer Support Tools", are the following:

- Independent raw data storage
- Analytics metadata storage
- Data Integration Engine

Each component is tested through unit tests and integration tests are the same. The stable version of the APIs is continuously updated once new releases are available. Details about the testing procedures for each component are presented in Appendix D.

The Data Lake API is summarised in Table 5, p.46.

3.5 Analytics

The Analytics is responsible for the establishment of statistics and analysis of processus on the basis of the collected data. Details about the testing procedures of Analytics are presented in Appendix F (Unit and integration testing, testing environment).

The Analytics API is summarised in Table 5, p.46.
3.6 Security and Privacy management

Security and privacy are addressed across the different elements in ACTIVAGE. As of the date of the delivery of the present document, it has been performed an evaluation of the level of security and privacy of each ACTIVAGE component.

Each element of the ACTIVAGE system has been individually evaluated using the self-reporting questionnaire, see D3.3 and D3.8 update for the results of said evaluation.

Integrated security testing will be added in the next version of AIoTES, particularly testing end to end security features, and the whole Security and Privacy module. Deeper testing has not been done at the date of D5.3 final edition, as far as the Security and Privacy module is currently under development.

3.7 AIoTES API

The APIs of internal components of AIoTES (SIL, Data Lake, Data Analytics, etc.) are currently available. AIoTES API encompasses the functionality and operations of each AIoTES component API, and these functions have been tested separately.

Specifically, section 3.1.7 includes the testing of the methods related to the SIL API, section 3.4 includes all the implemented and tested methods related to the Data Lake API, while the section 3.5 tests the methods related to the Analytics API. These tests check in a primary way the AIoTES API functionality (as it is provided by the internal AIoTES components APIs).

Here is a table that summarizes all these methods with the AIoTES URI format:

<table>
<thead>
<tr>
<th>Table 5: AIoTES API summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic Interoperability Layer (SIL)</strong></td>
</tr>
<tr>
<td>POST /aiotes/api/mw2mw/clients</td>
</tr>
<tr>
<td>GET /aiotes/api/mw2mw/clients</td>
</tr>
<tr>
<td>GET /aiotes/api/mw2mw/clients/{clientId}</td>
</tr>
<tr>
<td>PUT /aiotes/api/mw2mw/clients/{clientId}</td>
</tr>
<tr>
<td>DELETE /aiotes/api/mw2mw/clients/{clientId}</td>
</tr>
<tr>
<td>GET /aiotes/api/mw2mw/platform-types</td>
</tr>
<tr>
<td>POST /aiotes/api/mw2mw/platforms</td>
</tr>
<tr>
<td>GET /aiotes/api/mw2mw/platforms</td>
</tr>
<tr>
<td>GET /aiotes/api/mw2mw/platforms/{platformId}</td>
</tr>
<tr>
<td>PUT /aiotes/api/mw2mw/platforms/{platformId}</td>
</tr>
<tr>
<td>DELETE /aiotes/api/mw2mw/platforms/{platformId}</td>
</tr>
</tbody>
</table>
### POST /aiotes/api/mw2mw/responses
Retrieve response messages concerning the client

### POST /aiotes/api/mw2mw/request
Send given JSON-LD message downstream

### GET /aiotes/api/mw2mw/devices
List all devices registered in the SIL (or in the specified platform)

### POST /aiotes/api/mw2mw/devices
Register (start managing) new virtual devices in the SIL

### PUT /aiotes/api/mw2mw/devices/{deviceId}
Update specified virtual device

### DELETE /aiotes/api/mw2mw/devices/{deviceId}
Remove specified device

### POST /aiotes/api/mw2mw/subscriptions
Subscribe to specific devices

### GET /aiotes/api/mw2mw/subscriptions
List subscriptions

### DELETE /aiotes/api/mw2mw/subscriptions/{conversationId}
Remove specified subscription

### POST /aiotes/api/mw2mw/devices/{deviceId}/actuation
Send data to an actuator

### POST /aiotes/alignments
Upload a new alignment

### POST /aiotes/convert
Convert an alignment from XML format to RDF/XML

### POST /aiotes/convert/TTL
Convert cells of an alignment to Turtle format

### GET /aiotes/alignments
List alignments

### GET /aiotes/alignments/{name}/{version}
Get an alignment

### DELETE /aiotes/alignments/{name}/{version}
Remove specified alignment

### GET /aiotes/channels
List all channels

### POST /aiotes/channels
Create a new channel

### DELETE /aiotes/channels/{channelId}
Remove specified channels

### POST /aiotes/translation
Translate a JSON-LD message

---

**Data Lake**

### POST /aiotes/independentStorage/createDB
Creates a new database.

### POST /aiotes/independentStorage/select
Selects data from a database.

### POST /aiotes/independentStorage/insert
Inserts data into a database.

### POST /aiotes/independentStorage/update
Updates the data contained in a database.
### POST /aiotes/independentStorage/delete
Deletes data from a database.

### POST /aiotes/metadataStorage/createModel
Creates a new analytics model in the metadata storage.

### GET /aiotes/metadataStorage/getModel
Returns the stored parameters for a requested model.

### PUT /aiotes/metadataStorage/editModel
Edits an existing model in the metadata storage.

### DELETE /aiotes/metadataStorage/deleteModel
Deletes a model from the metadata storage.

### POST /aiotes/api/createIndex
Creates an index for the distributed data, based on specific indexing options.

### DELETE /aiotes/api/deleteIndex/{id}
Deletes an index for the distributed data.

### GET /aiotes/api/getAllIndex
Get all indexes in the database

### GET /aiotes/api/getIndex/{id}
Get index by id

### PUT /aiotes/api/updateIndex
Updates an index for the distributed data, based on specific indexing options.

### Analytics

<table>
<thead>
<tr>
<th>POST /aiotes/summary</th>
<th>Computes summary statistics of the input data</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST /aiotes/histogram</td>
<td>Computes a histogram of the provided numeric data.</td>
</tr>
<tr>
<td>POST /aiotes/lof</td>
<td>Computes anomaly scores for the input data, using the LOF method</td>
</tr>
<tr>
<td>POST /aiotes/lof/train</td>
<td>Trains a LOF anomaly detection model</td>
</tr>
<tr>
<td>POST /aiotes/kmeans</td>
<td>Clusters the input data using the k-means clustering algorithm</td>
</tr>
<tr>
<td>POST /aiotes/dbscan</td>
<td>Clusters the input data using the DBSCAN clustering algorithm</td>
</tr>
</tbody>
</table>

Further testing of the AIoTES API will be provided in the next version of this deliverable (D5.4 Final Results). Further information about the AIoTES API and related APIs can be found on D3.7.
3.8 Marketplace

The ACTIVAGE Marketplace is a one-stop-shop for providing, discovering and deploying applications built on AoTES. As such, it is a high level deployment tool, very close to end-users. Specifically, it is intended for applications users and developers alike. Developers, either internal from the Deployment Sites, or external, from the Open Call and beyond, can upload, promote and monetize their applications. Users, including healthcare professionals, carers and deployers at the Deployment Sites, external third parties and more, can search, discover get free or buy applications.

The current version 1.0 of the Marketplace is a standard web portal providing the above functionality supporting (App) Users, Developers and (Marketplace) Administrators. The Marketplace homepage for version 1.0 is shown on Error! Reference source not found. while its specifications are presented in the respective version 1.0 of D4.3. For context, a list of its main functionality list is provided here on Table 6:. Notably, no distinction needs to be made between ACTIVAGE participants and general users for roles. Usually, Marketplace administrators are ACTIVAGE members but Marketplace developers can be external adopters while Marketplace users can be the general public. The Marketplace does not need to have knowledge of their ACTIVAGE membership.

Note that the Marketplace does not do any distinction between the Activage / non activage roles.

Appendix G gives details about the tests of the Marketplace. Note that the Marketplace provides facilities for Issue tracking and Performance, Accessibility, Best Practices and SEO testing.

<table>
<thead>
<tr>
<th>Marketplace User Role</th>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[User]</td>
<td>View Most Downloaded and Top Rated Apps</td>
<td>See overall Marketplace statistics</td>
</tr>
<tr>
<td></td>
<td>Search All Apps</td>
<td>Search with various criteria</td>
</tr>
<tr>
<td></td>
<td>View supported Platforms and respective Apps</td>
<td>View AoTES Platforms and information about them with respect to hosted Apps</td>
</tr>
<tr>
<td></td>
<td>View and Edit User Profile, Installed Apps, Wishlist, All Comments</td>
<td>Manage user profile, maintain lists of Apps, manage comments and comply to GDPR</td>
</tr>
<tr>
<td></td>
<td>Show Statistics Downloads and Top Downloaded Ratings and Top Rated</td>
<td>Explore more statistics of downloads and rating in the Marketplace</td>
</tr>
<tr>
<td>[Developer]</td>
<td>Show Developer Dashboard With Apps Published, Revenue, Top Downloads, Average Rating, Comments received</td>
<td>Explore Developer profile, revenue, ratings, downloads and user feedback</td>
</tr>
</tbody>
</table>
The full list of GUIs and functions integrated in AIoTES is given in D5.5.

The full description of the Marketplace component, including implementation and technology stack is given in D4.3 (1st & 2nd version).

The verification and testing procedure of the Marketplace follows that of a standard web portal.

The Marketplace coding follows the MVC paradigm and best practices for any web portal, placing emphasis on well-documented functionality, issues and tracking, versioning and iterative testing throughout development. This verification procedure is reported here through the git/GitHub statistics exported directly through the platform. No additional effort was invested in developing unit tests in order to speed up the process and deliver an Alpha (1.0) Phase version, a Beta and a Final (2.0) version to satisfy all involved parties (project partners, open calls and external adopters).

As functionality develops, emphasis is also placed on the front-end UI/UX in terms of Performance, Accessibility, Best (Web Development) Practices and SEO for outreach. This was verified through established online verification services and before and after certain optimization were performed.
4 Validation

This section analyses the results of the first phase and establishes a diagnosis based on the assessment of the AIoTES with respect to the KPIs and success criteria. It estimates the status of the achieved results relatively to the ultimate objectives of the ACTIVAGE project.

As such the validation is performed by pilot deployment on dedicated sites. In the case of ATIVAGE, depending on the user requirement, the user may be either the DS owner, the social service operator, the community, the regulation entities, etc.

Once validated the AIoTES or the ACTIVAGE system can be deployed to other pilot sites for extensive use, which will put the system under stress tests, up to a point where it can be declared as validated and therefore good enough to be generalised. Eventually, the system will enter the operational phase and it can be subject to maintenance (whenever necessary or periodically).

Note that the sequence of verification and validation might be iterated depending on the development approach that is taken either by the development team.

4.1 Semantic Interoperability Layer

As mentioned, the Validation corresponds to meeting the “user” requirements, first at the individual component level, whenever possible and most often, at the system level after a partial or full integration. The validation of the SIL and Bridges aims therefore at checking the fulfilling of the user expectations.

Following is showed the first validation made with the integration of all components involved in the process of collecting data (sensors) through the process of sending that information to the IoT platforms and sharing this information between different IoT platforms using the SIL and specific bridges.

The first test takes into account two scenarios of validation:

- The first scenario of validation consists in an external application that obtains information from one or more IoT platforms.
- The second scenario of validation consists in an application created to work with a specific IoT platform but it can work with another platform in transparent way.

Both scenarios of validation have been tested and validated during the 2nd ACTIVAGE Technical Review, being shown on 2 of the technical demonstrations performed.

4.1.1 First scenario of validation

As mentioned the first scenario of validation consists in an external application that obtains information from one or more IoT platforms (multiplatform application), therefore that application can offer several services but using several IoT platforms, that is possible using SIL and the Bridges of each platform.

The Validation takes into account two IoT platforms: Sofia2 and universAAL (uAAL). For the execution of the test the validation in that scenario must comply with the next prerequisites:

- Sofia2 deployment
- uAAL deployment
- SIL deployment
• uAAL and Sofia2 bridges

The next figure shows the general architecture of the first scenario of validation.

Figure 14: General architecture of the first scenario of validation.

4.1.1 Preparation of first scenario

The main steps for the execution of first scenario of validation are:

1) Create client in SIL and connect the application with the SIL. A Client of SIL can be used for many applications depending of the functionalities.
2) Create bridges and include platforms in registry to connect platforms with the SIL. That Step will be executed only once for each Platform instance.
3) Subscribe application to data, for it is necessary add devices to work in the registry and subscribe to device data.

Figure 17: Step 3 — Subscription of the application to IoT data

4.1.1.2 Execution of first scenario

The execution of the first scenario of validation consists in one external application (external to any IoT platform) that collects the bio-measurements of two IoT platforms: SOFIA2 and Universal but independently, namely the application collecting information from the two IoT platforms. Anyway these platforms (SOFIA2 and universal in this case) don’t need any extra information to bring that information, Bridges and SIL are responsible of all process.

The Client of SIL transforms the application in a multiplatform app that gathers information from two Platforms. Therefore, this scenario represents a third party application that can be used in more than one DS (Deployment Site)

For the validation of Multiplatform application in the case of use de SOFIA2 IoT Platform, we use Sofia2 gateways connected the following sensors:

- Digital scale
- Blood pressure monitor
- Coagulometer
Figure 18: Architecture Overview of execution with SOFIA2 of Validation of Multiplatform application.

For the validation of Multi-Platform application in the case of use of Universal IoT Platform, we use a Mobile Phone like gateway connected the following sensors:

- Digital scale
- Blood pressure monitor
As mentioned before, the second validation scenario consists in transparently using an application that was created for a specific IoT platform, with another platform. Thanks to the SIL, we can convert an application specifically designed and created for an IoT platform as a native application into a multiplatform application.

In the present test, the validation scenario considers two IoT platforms: SOFIA2 and uAAL (universAAL) on which the TELEA application is installed, deployed and run. The execution of the test procedure involves the following operations to be supported and present:

- SOFIA2 deployment
- uAAL deployment
- SIL deployment
- uAAL and SOFIA2 bridges

The TELEA application has been created for working with the SOFIA2 platform. It collects biological measurements of patients in real-time, analyses them and prescribes a specific medical treatment depending on the measurements.

By means of the SIL, SIL bridges and SIL alignments, the platform-specific TELEA application will work as a multiplatform application, capable of managing sensors and sensor information from other platforms than SOFIA2, such as in the present case the uAAL platform. The TELEA application can then share bio-measurements from sensors connected to two IoT platform and
not only from SOFIA2. In the demo scenario TELEA receives information from a sensor (weight scale) hosted in an instance of a UniversAAL platform.

Thanks to the interoperability provided by the SIL, this sensor is represented in SOFIA2 as a virtual sensor, and SOFIA2 is able to receive and understand sensor measurements, although they come from a different platform, UniversAAL. TELEA is an application that represents only the sensors and sensor data that the DS SOFIA2 platform provides to TELEA. Thus, it is platform-specific and cannot manage data from a different platform. But as far as this virtual sensor is handled internally in SOFIA2 platform as if it were a SOFIA2 sensor, TELEA receives and represents any information from this virtual sensor from UniversAAL in the same way as it is done with any SOFIA2 sensors in the DS, indistinctly.

The Figure 11 shows the architecture of the second scenario of validation.

![Figure 11: Architecture overview of the second scenario of validation.](image)

4.1.2.1 Preparation of the second scenario

The main steps for the execution of second scenario of validation are:

1) Create bridges and include platforms in registry to connect platforms with the SIL. Add the associated alignments to those bridges in the IPSM and configure this component to perform the semantic translation on them. That step will be executed only once for each Platform instance.
Figure 21: Step 1 – Use of SIL bridges/alignments and inclusion of platforms in the SIL

2) Creation of virtual devices in SOFIA2, these devices corresponding to real devices connected to Universal (uAAL) IoT platform.

Figure 22: Step 2 – Creation of virtual devices in SOFIA2 associated to real devices from an uAAL platform
4.12.2 Execution of second scenario

As mentioned the execution of Validation of the second scenario consists in use the TELEA application like a Multiplatform Application. TELEA application works with SOFIA2 to obtain the bio-measurements of patients and bring a specific medical treatment according with the measurements obtained in real time. TELEA is only able to work with SOFIA2, as it is the general case of applications for IoT platforms.

By means of the SIL, SIL bridges and SIL alignments, the platform-specific TELEA application works as a Multiplatform application, capable of managing sensors and sensor information from other platforms rather than SOFIA2. TELEA is in this way able to obtain bio-measurements from sensors connected from another external IoT platform and not only from SOFIA2, such as universAAL. In the demo scenario TELEA receives information from a sensor (weight scale) hosted in an instance of a universAAL platform. This is detailed below.

Thanks to the interoperability provided by the SIL, this sensor is represented in SOFIA2 as a virtual sensor, and SOFIA2 is able to receive and understand sensor measurements, although they come from a different platform, universAAL. TELEA is an application that represents only the sensors and sensor data that the DS SOFIA2 platform provides to TELEA. Thus, it is platform-specific and cannot manage data from a different platform. But as far as this virtual sensor is handled internally in SOFIA2 platform as if it were a SOFIA2 sensor, TELEA receives and represents any information from this virtual sensor from universAAL in the same way as it is done with any SOFIA2 sensors in the DS, indistinctly.

Thanks to the use of SIL and bridges, the TELEA application can obtain bio-measurements form sensors connected to Universal IoT platform. Therefore, with this deployment, it is possible to share bio-measurements collected in real time from DS 1 (Galicia DS) and a DS that employs universAAL, such as DS 3 (Madrid DS) – see Appendix C.

For the validation we use a Mobile Phone like gateway connected to the next sensors to universAAL IoT platform:

- Digital scale
- Blood pressure monitor

In addition, we have created these virtual sensors in SOFIA2 platform.
In Figure 23 we can see the flow of information from real sensors through the Universal platform and SIL using the bridge and alignment of universAAL. The SIL is responsible to send that information to SOFIA2 through the specific bridge and alignment of SOFIA2, and TELEA can work with that information in transparent way. UniversAAL data messages with sensor information are converted into SOFIA2 messages by means of the SIL alignments and bridges.
5 Conclusion and innovation path

This section serves as a conclusion by explaining the results and findings, and how they can drive innovation and benefit in AIoTES. The solutions and technological components proposed in open calls will be assessed in the controlled environments and then deployed (EXPAND and GROWTH phase). The assessment will be done using the same or adapted evaluation grid, method, metrics and KPIs. Note that liaison with WP6 has been established to ensure the continuity in the validation process.

5.1 Innovation path

This document reports about all steps in the verification and validation of the AIoTES components and experiment a first validation of the SIL.

The current status of the test and validation process corresponds to a key point at which the AIoTES components and integrated system have been tested in terms of compliance to technical specification (i.e. the technical requirements to which they correspond to). Some parts, e.g., the management, have been tested in a more representative environment, such as testbeds, which are closer to the DS than the laboratory. Eventually, the results obtained during the GROW phase are encouraging, in spite of the complexity and huge amount of specification, development and testing involved.

The reference model for the test and validation (as defined in D5.1 and also known as “the validation plan”) was however loosely applied and followed. The traceability of the original requirements and the corresponding tests is not explicitly ensured. The ACTIVAGE team will make sure in the future that the traceability will be implemented more faithfully, in particular by using the proposed templates, which makes an explicit reference to the user requirement or to technical specification. A specific training is planned for internal education.

The above methodology will also be used for the Open calls.

The next steps in the test and validation process are the following after the successful test of the individual component and the lab or testbed tests of the AIoTES:

- deployment of the AIoTES onto the DS platforms
- Validation of the AIoTES suing the test scenarios (not a user validation), which will
- Training
- Stress tests will test the system on load with all the actual processes involved, including physical actions.

The test and validation process as defined in D5.1 has proved to be effective and relevant, even though we have allowed for as much flexibility as possible to get as much results as possible. In addition, with respect to the above procedures and life cycle, the intermediate results have raised the need for amendments and revisions of the concepts. In particular, the validation is probably a two-stage notion that has a different aims and scope in a DS and in a laboratory or testbed. Given the cost of the deployment, we plan to have, as much as possible, an early validation performed in the testbeds or lab before the deployment in any DS.
5.2 Open call

Currently, there is no other DS that joined the project, on their own initiative or through Open calls. No DS has developed any Additional use cases. However, new DS will enter the ACTIVAGE user base through Open calls.

The next release of the document will take into account new scenarios proposed by the original DS and by the proposers to open calls. The solutions proposed in the context of the Open calls will be described and evaluated at the time of the Evaluation of OC solutions, in the context of the Use cases elaborated by and for the other DS.

Furthermore, the DS of the Open call will be trained using the format and material experimented on the ACTIVAGE team, for a deep understanding of the validation methodology, the tools that are provided to monitor, test and validate as well as the indicators that are used in the assessment.
References


17. Parliament [Internet]. [cited 2017 Dec 13]. Available from:
http://parliament.semwebcentral.org/

## Appendix A: SIL API functional tests

### List Alignments

<table>
<thead>
<tr>
<th>GET /alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>List alignments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Schema</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>- 200 OK</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;descId&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;id&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;date&quot;: 0,</td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;sourceOntologyURI&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;targetOntologyURI&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;version&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;creator&quot;: &quot;string&quot;,</td>
</tr>
</tbody>
</table>
|        |   "description": "string"
|        | }     |
|        | ]     |
Figure 24: IPSM get Alignments

Upload Alignment

**POST /alignments**

Upload a new alignment

<table>
<thead>
<tr>
<th>Input</th>
<th>Schema (Alignment in RDF/XML format)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example</td>
</tr>
</tbody>
</table>

**Implementation Notes**

Lists alignments uploaded to the IPSM.

**Response Class (Status 200)**

Information about alignments available in this IPSM instance.

Model Example Value

```
{
  "descld": "string",
  "set": false,
  "date": "2019-02-14",
  "name": "string",
  "sourceOntologyURI": "string",
  "targetOntologyURI": "string",
  "version": "3.0.1",
  "creator": "string"
}
```

**Response Content Type** application/json

Try it out  Hide Response

**Curl**

```
curl -X GET -H 'Accept: application/json' 'http://localhost:8080/alignments'
```

**Request URL**

http://localhost:8080/alignments

**Response Body**

```
[
  {
    "name": "UniversAAL_CO_align",
    "sourceOntologyURI": "http://ontology.universAAL.org/Context.owl#",
    "description": "ABORO user data - Alignment between UniversAAL messages with context events and INTER-IoT central ontology.
    "creator": "GRAPAS",
    "targetOntologyURI": "http://inter-IoT.eu/003IoTPeak",
    "version": "3.0.1",
    "id": 1,
    "date": "2019-02-14",
  }
]
```
D5.3 Intermediate Validation Results

Code
- 201 Success
- 208 Alignment already uploaded
- 400 Invalid request
- 409 Alignment with the same ID but different definition exists

Output

**Schema**

```json
{
    "message": "string"
}
```

**Example**

```json
{
    "message": "Alignment 1 uploaded"
}
```

**Figure 25: IPSM post Alignments**
### D5.3 Intermediate Validation Results

#### Request URL

```
http://localhost:8080/alignments
```

#### Response Body

```
{
  "message": "Alignment with AlignmentID(SOFI62_CO_0.1) uploaded successfully"
}
```

#### Response Code

201

#### Response Headers

```
{
  "access-control-allow-credentials": "true",
  "access-control-allow-origin": "http://localhost:8080",
  "content-length": "777",
  "content-type": "application/json",
  "date": "Mon, 11 Feb 2019 12:25:46 GMT",
  "server": "waka-http/4.0.1.5"
}
```

---

**Figure 26: IPSM post Alignments response**

### Delete Alignment

**DELETE /alignments/{name}/{version}**

Remove specified alignment

#### Input

- Schema
  -  
- Example
  -  

#### Code

- 204 Success
- 400 Invalid request
- 500 Deletion failed

#### Output

- Schema
  -  

```
{
  "message": "string"
}
```
### D5.3 Intermediate Validation Results

**DELETE** 
/alignments/(name)/(version)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>SOFIA2_CO</td>
<td>Name of the alignment to delete</td>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>version</td>
<td>0.1</td>
<td>Version of the alignment to delete</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

**Response Messages**

<table>
<thead>
<tr>
<th>HTTP Status Code</th>
<th>Reason</th>
<th>Response Model</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>Alignment deleted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>Alignment not found</td>
<td>Model</td>
<td></td>
</tr>
</tbody>
</table>

```json
{
  "message": "string"
}
```

<table>
<thead>
<tr>
<th>HTTP Status Code</th>
<th>Reason</th>
<th>Response Model</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>Alignment deletion failed</td>
<td>Model</td>
<td></td>
</tr>
</tbody>
</table>

```json
{
  "message": "string"
}
```

---

**Figure 27: IPSM delete Alignments**

**Curl**

curl -X DELETE \-header 'Accept: application/json' \-http://localhost:8888/alignments/SOFIA2_CO/0.1

**Request URL**

http://localhost:8888/alignments/SOFIA2_CO/0.1

**Response Body**

```json
{
  "message": "Alignment with AlignmentID(SOFIA2_CO,0.1) successfully deleted"
}
```

**Response Code**

200

**Response Headers**

```json
{
  "access-control-allow-credentials": "true",
  "access-control-allow-origin": "http://localhost:8888",
  "content-length": "76",
  "content-type": "application/json",
  "date": "Fri, 11 Feb 2011 11:28:37 GMT",
  "server": "akka-http/0.1.5"
}
```

---

**Figure 28: IPSM delete Alignments response**
Get Alignment by Name and Version

GET /alignments/{name}/{version}

Get an alignment

<table>
<thead>
<tr>
<th>Input</th>
<th>Schema</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th></th>
<th>- 200 Success</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>- 400 Invalid request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 500 Retrieval failed</td>
</tr>
</tbody>
</table>

| Output | Schema | (Alignment in RDF/XML format) |

Figure 29: IPSM get Alignment by name and version
Convert Alignment legacy format into RDF/XML

**POST /convert**

Convert an alignment from XML format to RDF/XML

<table>
<thead>
<tr>
<th>Input Schema</th>
<th>(Alignment in XML format)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td></td>
</tr>
</tbody>
</table>

| Code          | - 201 Success - 500 Error |

<table>
<thead>
<tr>
<th>Output Schema</th>
<th>(Alignment in RDF/XML format)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td></td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

**Figure 31: IPSM convert Alignment request**

**Request URL**

http://localhost:8888/convert

**Response Body**

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <Alignments rdf:about="http://inter-lot.eu/iotext2#sofiA2_0001">
    <dcterms:title>SOFI A2_0001</dcterms:title>
    <dcterms:creator>SOFI A2_0001</dcterms:creator>
    <dcterms:description>Alignment between SOFI A2 biomedical data and GIoT-Plex ontology</dcterms:description>
    <dcterms:date>Mon Feb 11 12:18:13 GMT 2019</dcterms:date>
  </Alignments>
</rdf:RDF>
```

**Response Code**

201

**Response Headers**

```json
{
  "access-control-allow-credentials": "true",
  "accept-control-allow-origin": "http://localhost:8888",
  "content-length": "12954",
  "content-type": "application/xml",
  "date": "Mon, 11 Feb 2019 12:18:13 GMT",
  "server": "akka-http/1.5"
}
```

**Figure 32: IPSM convert Alignment response**
### Convert Alignment legacy format into Turtle

**POST /convert/TTL**

Convert cells of an alignment to Turtle format

<table>
<thead>
<tr>
<th>Input</th>
<th>Schema</th>
<th>Alignment cells in XML format</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Output</th>
<th>(Alignment cells in turtle format)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example</td>
<td></td>
</tr>
</tbody>
</table>

#### Implementation Notes

Converts cells of an alignment to Turtle format

Response Class (Status 201)

string

Response Content Type: application/xml

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
</table>

`alignment`<br> Alignment source

<code>`<s2:hasAttribute>`
</code>

Figure 33: IPSM convert Alignment to Turtle format request
D5.3 Intermediate Validation Results

Request URL

http://localhost:8880/convert/TTL

Response Body

```
<align:entity rdf:about="http://www.inter-iot.eu/sriips#%_patient">
  <a rdf:ptype="http://www.w3.org/2001/XMLSchema#string">
    s2:hasAttribute [ a s2:Attribute , sriips:node_A ;
      s2:hasName "idProfesionales" ;
      s2:hasValue [ a sriips:node_doctor ] ] ;

    s2:hasAttribute [ a s2:Attribute , sriips:node_B ;
      s2:hasName "idPaciente" ;
      s2:hasValue [ a sriips:node_patient ] ]
  ].

  s2:Observation , sriips:node_obs ].
</align:entity1>

<align:entity2 rdf:datatype="https://www.w3.org/2001/XMLSchema#string">
  l物联网OrderByUser [ a s2:User ;
    s2:hasProperty "idPaciente" ;
    s2:hasValue [ a sriips:node_patient ]
  ].
</align:entity2>

Figure 34: IPSM convert Alignment to Turtle format response

Register SIL client

POST /api/mw2mw/clients

Register a new client

<table>
<thead>
<tr>
<th>Headers</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client-ID (String)</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>clientId</td>
</tr>
<tr>
<td></td>
<td>callbackUrl</td>
</tr>
<tr>
<td></td>
<td>receivingCapacity</td>
</tr>
<tr>
<td></td>
<td>responseFormat</td>
</tr>
<tr>
<td></td>
<td>responseDelivery</td>
</tr>
</tbody>
</table>

Response Code

201

Response Headers

```

{  "access-control-allow-credentials": "true",
   "access-control-allow-origin": "http://localhost:8880",
   "content-length": "996",
   "content-type": "application/xml",
   "date": "Mon, 11 Feb 2019 11:41:16 GMT",
   "server": "akka-http/0.1.5"
}
Example
{  "clientId": "testclient",  "callbackUrl": "http://localhost:1880/receiver",  "receivingCapacity": 5,  "responseFormat": "JSON_LD",  "responseDelivery": "SERVER_PUSH"}

Output Schema

{  "clientId": "string",  "callbackUrl": "string",  "receivingCapacity": int,  "responseFormat": "JSON_LD",  "responseDelivery": "CLIENT_PULL"}

Example
{  "clientId": "testclient",  "callbackUrl": "http://localhost:1880/receiver",  "receivingCapacity": 5,  "responseFormat": "JSON_LD",  "responseDelivery": "SERVER_PUSH"}

Figure 35: Register client (request)
List SIL clients

**GET /api/mw2mw/clients**

**List all clients**

<table>
<thead>
<tr>
<th>Input</th>
<th>Headers</th>
<th>Client-ID (String)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Schema</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- 200 Success</td>
</tr>
<tr>
<td></td>
<td>- 401 Unauthorized</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;clientId&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;callbackUrl&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;receivingCapacity&quot;: int,</td>
</tr>
<tr>
<td></td>
<td>&quot;responseFormat&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;responseDelivery&quot;: &quot;string&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;clientId&quot;: &quot;testclient&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;callbackUrl&quot;: &quot;<a href="http://localhost:1880/receiver">http://localhost:1880/receiver</a>&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;receivingCapacity&quot;: 5,</td>
</tr>
<tr>
<td></td>
<td>&quot;responseFormat&quot;: &quot;JSON_LD&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;responseDelivery&quot;: &quot;SERVER_PUSH&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>]</td>
</tr>
</tbody>
</table>
Get SIL client

GET /api/mw2mw/clients/{clientId}

Retrieve specified client

<table>
<thead>
<tr>
<th>Input</th>
<th>Headers</th>
<th>Client-ID (String)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Schema</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>-</td>
</tr>
</tbody>
</table>

| Code   | - 200 Success |
|        | - 401 Unauthorized |
|        | - 404 Not found  |

<table>
<thead>
<tr>
<th>Output</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
</tbody>
</table>
|        | "clientId": "string",
|        | "callbackUrl": "string",
|        | "receivingCapacity": int,
|        | "responseFormat": "string",
|        | "responseDelivery": "string"
|        | }                  |
Example

```
{
"clientId": "testclient",
"callbackUrl": "http://localhost:1880/receiver",
"receivingCapacity": 5,
"responseFormat": "JSON_LD",
"responseDelivery": "SERVER_PUSH"
}
```

Figure 39: Get client (request)

Figure 40: Get client (response)

List Supported Platform Types

**GET /api/mw2mw/platform-types**

List all supported platforms

<table>
<thead>
<tr>
<th>Input</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headers</td>
<td>Client-ID (String)</td>
</tr>
</tbody>
</table>

---

Version 1.0 | 2019-02-14 | ACTIVAGE ©
### Example

**Code**

- **200 Success**
- **401 Unauthorized**

**Output Schema**

```json
{
    "conversationId": "string"
}
```

**Example**

```json
{
    "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd"
}
```

---

**Register new platform**

**POST /api/mw2mw/platforms**

Register a new platform instance in the SIL

**Headers**

- **Client-ID (String)**

---

Figure 41: Get supported platform types (request)

Figure 42: Get supported platform types (response)
### Input Schema

```json
{
    "platformId": "string",
    "type": "string",
    "baseEndpoint": "string",
    "location": "string",
    "name": "string",
    "username": "string",
    "encryptedPassword": "string",
    "encryptionAlgorithm": "string",
    "downstreamInputAlignmentName": "string",
    "downstreamInputAlignmentVersion": "string",
    "downstreamOutputAlignmentName": "string",
    "downstreamOutputAlignmentVersion": "string",
    "upstreamInputAlignmentName": "string",
    "upstreamInputAlignmentVersion": "string",
    "upstreamOutputAlignmentName": "string",
    "upstreamOutputAlignmentVersion": "string"
}
```

### Example

```json
{
    "platformId": "http://inter-iot.eu/platforms/uaal",
    "type": "http://inter-iot.eu/UniversAAL",
    "baseEndpoint": "http://localhost:9000/uaal/",
    "location": "http://test.inter-iot.eu/TestLocation",
    "name": "uAAL Platform",
    "downstreamInputAlignmentName": "",
    "downstreamInputAlignmentVersion": "",
    "downstreamOutputAlignmentName": "CO_UniversAAL_align",
    "downstreamOutputAlignmentVersion": "3.0.1",
    "upstreamInputAlignmentName": "UniversAAL_CO_align",
    "upstreamInputAlignmentVersion": "3.0.1",
    "upstreamOutputAlignmentName": "",
    "upstreamOutputAlignmentVersion": ""
}
```

### Code

- 202 The request has been accepted for processing
- 400 Invalid request
- 401 Unauthorized
- 409 Platform already registered

### Output Schema

```json
{
    "conversationId": "string"
}
```

### Example

```json
{
    "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd"
}
List platforms

**GET /api/mw2mw/platforms**

List all platforms registered in the SIL

<table>
<thead>
<tr>
<th>Input</th>
<th>Headers</th>
<th>Schema</th>
<th>Example</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client-ID (String)</td>
<td>-</td>
<td>-</td>
<td>200 Success</td>
</tr>
</tbody>
</table>
### Output Schema

```json
[
  {
    "platformId": "string",
    "type": "string",
    "baseEndpoint": "string",
    "location": "string",
    "name": "string",
    "clientId": "string",
    "username": "string",
    "timeCreated": 0,
    "platformStatistics": {
      "deviceCount": 0,
      "subscribedDeviceCount": 0,
      "subscriptionCount": 0
    },
    "downstreamInputAlignmentName": "string",
    "downstreamInputAlignmentVersion": "string",
    "downstreamOutputAlignmentName": "string",
    "downstreamOutputAlignmentVersion": "string",
    "upstreamInputAlignmentName": "string",
    "upstreamInputAlignmentVersion": "string",
    "upstreamOutputAlignmentName": "string",
    "upstreamOutputAlignmentVersion": "string"
  }
]
```

### Example

```json
[
  {
    "platformId": "http://inter-iot.eu/platforms/uaal",
    "type": "http://inter-iot.eu/UniversAAL",
    "baseEndpoint": "http://localhost:9000/uaal/",
    "location": "http://test.inter-iot.eu/TestLocation",
    "name": "uAAL Platform",
    "clientId": "http://inter-iot.eu/clients#myclient",
    "username": "",
    "timeCreated": 1540464606238,
    "platformStatistics": {
      "deviceCount": 0,
      "subscribedDeviceCount": 0,
      "subscriptionCount": 0
    },
    "downstreamInputAlignmentName": "",
    "downstreamInputAlignmentVersion": "",
    "downstreamOutputAlignmentName": "CO_UniversAAL_align",
    "downstreamOutputAlignmentVersion": "3.0.1",
    "upstreamInputAlignmentName": "UniversAAL_CO_align",
    "upstreamInputAlignmentVersion": "3.0.1",
    "upstreamOutputAlignmentName": "",
    "upstreamOutputAlignmentVersion": ""
  }
]
```
List information of a platform

**GET /api/mw2mw/platforms/{platformId}**

Get information about the specified platform

**Input Schema**
- **Example**

**Code**
- 200 Success
- 401 Unauthorized

**Output Schema**
- ```
  {
    "platformId": "string",
  }
```
Figure 47: Get platform (request)

```
```

Figure 48: Get platform (response)
# Update platform information

**PUT /api/mw2mw/platforms/{platformId}**

**Update the specified platform**

<table>
<thead>
<tr>
<th>Headers</th>
<th>Client-ID (String)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Schema</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;baseEndpoint&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;location&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;username&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;encryptedPassword&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;encryptionAlgorithm&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;downstreamInputAlignmentName&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;downstreamInputAlignmentVersion&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;downstreamOutputAlignmentName&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;downstreamOutputAlignmentVersion&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

**Example**

```json
{
   "baseEndpoint": "http://localhost:9000/uaal/",
   "location": "http://test.inter-iot.eu/TestLocation",
   "name": "uAAL Platform",
   "downstreamInputAlignmentName": "",
   "downstreamInputAlignmentVersion": "",
   "downstreamOutputAlignmentName": "CO_UniversAAL_align",
   "downstreamOutputAlignmentVersion": "3.0.1",
   "upstreamInputAlignmentName": "UniversAAL_CO_align",
   "upstreamInputAlignmentVersion": "3.0.1",
   "upstreamOutputAlignmentName": "",
   "upstreamOutputAlignmentVersion": ""
}
```

**Code**

- 202 The request has been accepted for processing
- 400 Invalid request
- 401 Unauthorized
- 404 Not found

**Output Schema**

```json
{
   "conversationId": "string"
}
```

**Example**

```json
{
   "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd"
}
**Figure 49: Update platform (request)**

**Figure 50: Update platform (response)**

**Unregister platform**

```plaintext
DELETE /api/mw2mw/platforms/{platformId}
```

**Remove specified platform instance**

<table>
<thead>
<tr>
<th>Input</th>
<th>Headers</th>
<th>Schema</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client-ID (String)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Output

Schema

```
{
  "conversationId": "string"
}
```

Example

```
{
  "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd"
}
```

Figure 51: Unregister platform (request)

**Creation of virtual device**

**POST /api/mw2mw/devices**

Register (start managing) new virtual devices in the SIL

| Headers   | Client-ID (String) |
Input Schema

```json
{  
  "devices": [  
    {  
      "deviceTypes": [  
        "DEVICE"  
      ],  
      "deviceId": "string",  
      "hostedBy": "string",  
      "location": "string",  
      "name": "string",  
      "hosts": [  
        "string"  
      ],  
      "forProperty": [  
        "string"  
      ],  
      "madeActuation": "string",  
      "implementsProcedure": "string",  
      "observes": [  
        "string"  
      ],  
      "detects": "string",  
      "madeObservation": "string"  
    }  
  ]
}
```

Example

```json
{  
  "devices": [  
    {  
      "deviceId": "http://inter-iot.eu/dev/sensor",  
      "hostedBy": "http://inter-iot.eu/platforms/fiware1",  
      "location": "http://test.inter-iot.eu/Testlocation",  
      "name": "sensor"  
    }  
  ]
}
```

Code

- 201 Success
- 400 Invalid request
- 401 Unauthorized
- 409 Device already registered

Output Schema

```json
{  
  "conversationId": "string"
}
```

Example

```json
{  
  "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd"
}
```
List devices

**GET /api/mw2mw/devices**

List all devices registered in the SIL (or in the specified platform)

<table>
<thead>
<tr>
<th>Headers</th>
<th>Client-ID (String)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>platformId</td>
</tr>
<tr>
<td>Schema</td>
<td>-</td>
</tr>
<tr>
<td>Example</td>
<td>-</td>
</tr>
<tr>
<td>Code</td>
<td>- 200 Success</td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

Output Schema

```
[  {  
"deviceTypes": [  
"DEVICE"  
],  
"deviceId": "string",  
"hostedBy": "string",  
"location": "string",  
"name": "string",  
"hosts": [  
"string"  
],  
"forProperty": [  
"string"  
],  
"madeActuation": "string",  
"implementsProcedure": "string",  
"observes": [  
"string"  
],  
"detects": "string",  
"madeObservation": "string"  
}  
]
```

Example

```
[  {  
"deviceTypes": [  
"DEVICE"  
],  
"deviceId": "http://ontology.universAAL.org/InterIoT.owl#sensor99761296A000",  
"hostedBy": "http://inter-iot.eu/platforms/uaal",  
"location": "http://test.inter-iot.eu/TestLocation",  
"name": "example uAAL sensor",  
"hosts": [],  
"forProperty": [],  
"madeActuation": null,  
"implementsProcedure": null,  
"observes": [],  
"detects": null,  
"madeObservation": null  
}  
]
**Update device**

**PUT /api/mw2mw/devices/{deviceIds}**

**Update specified virtual device**

<table>
<thead>
<tr>
<th>Headers</th>
<th>Input Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client-ID (String)</td>
<td></td>
</tr>
</tbody>
</table>

```json
{
    "deviceTypes": [ "DEVICE"
    ],
    "deviceIds": "string",
    "hostedBy": "string",
    "location": "string",
    "name": "string",
    "hosts": [ "string"
    ],
    "forProperty": [ "string"
    ]
}```
Example
{
  "deviceId": "http://inter-iot.eu/dev/sensor",
  "hostedBy": "http://inter-iot.eu/platforms/fiware1",
  "location": "http://test.inter-iot.eu/TestLocation",
  "name": "sensor"
}

Code
- 202 The request has been accepted for processing
- 400 Invalid request
- 401 Unauthorized
- 404 Not found

Output Schema
{
  "conversationId": "string"
}

Example
{
  "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd"

Figure 57: Update device (request)
**Delete device**

**DELETE /api/mw2mw/devices/{deviceID}**

Remove specified device

<table>
<thead>
<tr>
<th>Headers</th>
<th>Client-ID (String)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>-</td>
</tr>
<tr>
<td>Example</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>202 The request has been accepted for processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>401 Unauthorized</td>
</tr>
<tr>
<td></td>
<td>404 Not found</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>{ &quot;conversationId&quot;: &quot;string&quot; }</td>
</tr>
<tr>
<td>Example</td>
<td>{ &quot;conversationId&quot;: &quot;conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd&quot; }</td>
</tr>
</tbody>
</table>
Subscribe to information from specified devices

**POST /api/mw2mw/subscriptions**

Subscribe to specified devices

<table>
<thead>
<tr>
<th><strong>Input</strong></th>
<th><strong>Headers</strong></th>
<th><strong>Client-ID (String)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong></td>
<td><strong>deviceIds</strong>: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;string&quot; ]</td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>&quot;deviceIds&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;<a href="http://inter-iot.eu/dev/sensor">http://inter-iot.eu/dev/sensor</a>&quot; ]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Code</strong></th>
<th>- 202 The request has been accepted for processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- 400 Invalid request</td>
</tr>
<tr>
<td></td>
<td>- 401 Unauthorized</td>
</tr>
<tr>
<td></td>
<td>- 409 The client is already subscribed to the devices</td>
</tr>
</tbody>
</table>

| **Output** | **Schema** | { |
|------------|------------|"conversationId": "string" |
| **Example** | { |
| | "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd" |} |
List subscriptions

**GET /api/mw2mw/subscriptions**

<table>
<thead>
<tr>
<th>Headers</th>
<th>Parameters</th>
<th>Schema</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client-ID (String)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

<table>
<thead>
<tr>
<th>Code</th>
<th>Output</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 200 Success</td>
<td>[ ]</td>
<td></td>
</tr>
<tr>
<td>- 401 Unauthorized</td>
<td>[ ]</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```
[{
  "conversationId": "string",
  "deviceIds": [
    "string"
  ],
  "clientId": "string"
}]
```

```
[{
  "conversationId": "conv62da934d-16c1-461e-b4b0-456b2d11ae3b",
  "deviceIds": [
    "http://inter-iot.eu/dev/sensor"
  ],
  "clientId": "myclient"
}]
```

**Figure 63: List subscriptions (request)**

GET /me/autocomplete/ - List subscriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientId</td>
<td>myclient</td>
</tr>
<tr>
<td>clientId</td>
<td>myclient</td>
</tr>
</tbody>
</table>

Request URL:

```
http://localhost:8080/api/me/autocomplete?clientId=myclient
```

Server response:

**Code: 200**

```
{
  "conversationId": "conv62da934d-16c1-461e-b4b0-456b2d11ae3b",
  "deviceIds": [
    "http://inter-iot.eu/dev/sensor"
  ],
  "clientId": "myclient"
}
```

**Response headers**

- `Content-Type: application/json`
- `Server: Flask/1.0`
- `Date: Thu, 14 Feb 2019 11:31:08 GMT`
Unsubscription

DELETE /api/mw2mw/subscriptions/{conversationId}

Remove specified subscription

<table>
<thead>
<tr>
<th>Input</th>
<th>Headers</th>
<th>Client-ID (String)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Schema</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;conversationId&quot;: &quot;string&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

Example

```
{ "conversationId": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd" }
```

Figure 64: List subscriptions (response)

Figure 65: Unsubscribe (request)
### Get client response messages

**POST /api/mw2mw/responses**

Retrieve response messages concerning the client

<table>
<thead>
<tr>
<th>Input</th>
<th>Headers</th>
<th>Client-ID (String)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Type: application/ld+json</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Schema</th>
<th>-</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Example</th>
<th>-</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>202</th>
<th>The request has been accepted for processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>400</td>
<td>Invalid request</td>
</tr>
<tr>
<td></td>
<td>401</td>
<td>Unauthorized</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Schema</th>
<th>Returns array of messages in JSON-LD format or empty array if none is available.</th>
</tr>
</thead>
</table>

D5.3 Intermediate Validation Results

Example

```json
{
    "@graph": [
        {
            "@id": "InterIoTMsg:meta/692246e4-9e60-4102-9908-000676b6f5991",
            "@type": [
                "InterIoTMsg:Response",
                "InterIoTMsg:Platform_register",
                "InterIoTMsg:meta"
            ],
            "InterIoTMsg:SenderPlatformId": {
                "@id": "InterIoT:platforms/fiware1"
            },
            "InterIoTMsg:conversationID": "conv0686c2b3-e16a-4957-80e7-a47fd2dd6edd",
            "InterIoTMsg:dateTimeStamp": "2018-07-04T15:29:41.652Z",
            "InterIoTMsg:messageID": "msg7907dcfe-d402-44e5-a9fc-60e64e3aff18",
            "InterIoTMsg:status": "OK"
        }
    ],
    "@id": "InterIoTMsg:metadata",
    "@context": {
        "InterIoTMsg": "http://inter-iot.eu/message/",
        "InterIoT": "http://inter-iot.eu/"
    }
}
```

Figure 67: Get response messages (request)
Figure 68: Get response messages (response)
Appendix B : Tests done on DS6 Isère

This section lists the tests performed in the Isère deployment site. Tests and results are briefly described. These tests were performed in a test bed in order to validate the technological solution for the Panel 3 of DS6. This solution uses the sensiNact platform with some IoT devices to perform a certain number of services to two kind of users: beneficiaries and caregivers.

Tests are defined in first requirements, then a set of test cases that verifies one or several requirements. At the end, tests are performed in a test campaign. Most of the test results presented in the following table were performed manually in the testbed, although some additional test can be conducted using the test automation framework in relation with an integration tool such as Jenkins.

1-Sensors > 1-Info sNa
Test to check whether information from sensor is seen by sensiNact

**sNa_Bed sensor**

<table>
<thead>
<tr>
<th>Created on</th>
<th>19 janv. 2018 16:29 (by claire)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified on</td>
<td>19 janv. 2018 16:31 (by claire)</td>
</tr>
<tr>
<td>ID</td>
<td>32</td>
</tr>
<tr>
<td>Nature</td>
<td>Functional</td>
</tr>
<tr>
<td>Type</td>
<td>Recevabilité</td>
</tr>
<tr>
<td>Status</td>
<td>Under edition</td>
</tr>
<tr>
<td>Importance</td>
<td>Very high</td>
</tr>
</tbody>
</table>

**Description**
Test case for bed sensor for information in sNa.

**Pré-requis :**
Services Eclipse sensiNact and OpenHab are started.
In OpenHab, sensor binary must be set for the device.

**Considered requirements :**

**STEP 1 :**

<table>
<thead>
<tr>
<th><strong>Action</strong></th>
<th><strong>Expected result :</strong> Value for the bed sensor is &quot;close&quot; in sensiNact platform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lie down on the bed and wait for 60s.</td>
<td></td>
</tr>
</tbody>
</table>
### D5.3 Intermediate Validation Results

- **Attachments**: None
- **Requirement**: None

### STEP 2:

<table>
<thead>
<tr>
<th>Action</th>
<th>Get up from the bed and wait 10s.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Expected result</th>
<th>Value for the bed sensor is &quot;open&quot; in sensiNact platform.</th>
</tr>
</thead>
</table>

- **Attachments**: None
- **Requirement**: None

### sNa Door contactor Bedroom

- **Created on**: 19 janv. 2018 16:27 (by claire)
- **Modified on**: 19 janv. 2018 16:29 (by claire)
- **ID**: 31
- **Nature**: Functional
- **Type**: Recevabilité
- **Status**: Under edition
- **Importance**: Very high

**Description**

Test case for door contactor of bedroom for information in sNa.

**Pré-requis**:

- Services Eclipse sensiNact and OpenHab are started.
- In OpenHab, sensor binary must be set for the device.

**Considered requirements**:

### STEP 1:

<table>
<thead>
<tr>
<th>Action</th>
<th>Close the door.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Expected result</th>
<th>Value for the door contactor is &quot;close&quot; in sensiNact platform.</th>
</tr>
</thead>
</table>

- **Attachments**: None
- **Requirement**: None
**STEP 2:**

| Action       | Open the door |

**Expected result:** Value for the door contactor is "open" in sensiNact platform.

- Attachments: None
- Requirement: None

**sNa_Hydrao showerhead**

*Created on:* 19 janv. 2018 16:35 (by claire)

*Modified on:* 19 janv. 2018 16:38 (by claire)

*ID:* 33

*Nature:* Functional

*Type:* Recevabilité

*Status:* Under edition

*Importance:* Very high

**Milestones:**

**Description**

Test case for Hydrao showerhead for information in sNa.

- ** Pré-requis:**
  - Service Eclipse sensiNact is started.
  - Python code for Hydrao is running.

**Considered requirements:**

**STEP 1:**

| Action | Start shower. |

**Expected result:** Value for the Hydrao shower is "true" in sensiNact platform.

- Attachments: None
- Requirement: None

**STEP 2:**

| Action | Stop the shower |
### Expected result:
Value for the Hydrao shower is "false" in sensiNact platform.

- Attachments: None
- Requirement: None

### sNa_Motion Bathroom

<table>
<thead>
<tr>
<th>Created on</th>
<th>17 janv. 2018 11:37 (by thomas)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified on</td>
<td>19 janv. 2018 13:18 (by claire)</td>
</tr>
<tr>
<td>ID</td>
<td>22</td>
</tr>
<tr>
<td>Nature</td>
<td>Functional</td>
</tr>
<tr>
<td>Type</td>
<td>Recevabilité</td>
</tr>
<tr>
<td>Status</td>
<td>Under edition</td>
</tr>
<tr>
<td>Importance</td>
<td>Very high</td>
</tr>
</tbody>
</table>

### Description
Test case for motion sensor of bathroom for information in sNa

### Pré-requis:
Services Eclipse sensiNact and OpenHab are started.
In OpenHab, sensor binary must be set for the device.
Waited time after a movement to send information of lack of movement, parameter "3" for Aeotec device or parameter "2" for Fibaro device, must be set to 10s.

### Considered requirements:

#### STEP 1:

| Action              | Move in front of the sensor. |

<table>
<thead>
<tr>
<th>Expected result:</th>
<th>Value for the motion sensor is &quot;on&quot; in sensiNact platform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachments:</td>
<td>None</td>
</tr>
<tr>
<td>Requirement:</td>
<td>None</td>
</tr>
</tbody>
</table>

#### STEP 2:

| Action              | Leave the room and wait for 20s. |

Expected result: Value for the motion sensor is "off" in sensiNact platform.

- Attachments: None
- Requirement: None

sNa_Motion Bedroom

Created on: 19 janv. 2018 14:32 (by claire)
Modified on: 19 janv. 2018 16:26 (by claire)
ID: 29
Nature: Functional
Type: Recevabilité
Status: Under edition
Importance: Very high
Milestones:

Description
Test case for motion sensor of bedroom for information in sNa

Pré-requis:
Services Eclipse sensiNact and OpenHab are started.
In OpenHab, sensor binary must be set for the device.
Waited time after a movement to send information of lack of movement, parameter "3" for Aeotec device or parameter "2" for Fibaro device, must be set to 10s.

Considered requirements:
STEP 1:
Action: Move in front of the sensor.

Expected result: Value for the motion sensor is "on" in sensiNact platform.

- Attachments: None
- Requirement: None

STEP 2:
Action: Leave the room and wait for 20s.
**Expected result:**
Value for the motion sensor is "off" in sensiNact platform.

- **Attachments:** None
- **Requirement:** None

### sNa_Motion Entrance

**Created on:** 19 janv. 2018 14:35 (by claire)
**Modified on:** 19 janv. 2018 16:26 (by claire)

**ID:** 30
**Nature:** Functional
**Type:** Recevabilité
**Status:** Under edition
**Importance:** Very high

**Description**
Test case for motion sensor of entrance for information in sNa

**Pré-requis:**
Services Eclipse sensiNact and OpenHab are started.

In OpenHab, sensor binary must be set for the device.

Waited time after a movement to send information of lack of movement, parameter "3" for Aeotec device or parameter "2" for Fibaro device, must be set to 10s.

**Considered requirements:**

**STEP 1:**

<table>
<thead>
<tr>
<th><strong>Action</strong></th>
<th>Move in front of the sensor.</th>
</tr>
</thead>
</table>

| **Expected result:** | Value for the motion sensor is "on" in sensiNact platform. |

- **Attachments:** None
- **Requirement:** None

**STEP 2:**

| **Action** | Leave the room and wait for 20s. |
Expected result:

Value for the motion sensor is "off" in sensiNact platform.

- Attachments: None
- Requirement: None

sNa_Temperature Bedroom

Created on: 19 janv. 2018 16:50 (by claire)
Modified on: 15 févr. 2018 10:29 (by thomas)
ID: 36
Nature: Functional
Type: Recevabilité
Status: Under edition
Importance: Very high
Milestones:

Description

Test case for temperature sensor of bedroom for information in sNa

Pré-requis:

Services Eclipse sensiNact and OpenHab are started.

In OpenHab, sensor temperature must be set for the device.

Periodicity of sending:

- For Fibaro, memorize parameters "62" and "64" and set them to 10.
- For aeotec, memorize parameter "41" and set it to 1. Also apply by default the parameters "101" to 241.

Considered requirements:

STEP 1:

Action

Note value of the temperature sensor in sensiNact platform.

Get the sensor in hands and wait for 1 minute.

Expected result:

Value for the temperature sensor is higher in sensiNact platform.

- Attachments: None
- Requirement: None
2-Actuators > 1-Action sNa
Test to check whether action on actuator can be done from sensiNact

sNa_Switch Bathroom

Created on : 19 janv. 2018 16:40 (by claire)
Modified on : 19 janv. 2018 16:43 (by claire)
ID: 34
Nature : Functional
Type: Recevabilité
Status : Under edition
Importance: Very high

Description
Test case for switch of bathroom in sNa

Premis : Services Eclipse sensiNact and OpenHab are started.
In OpenHab, sensor binary must be set for the device.
Light is off in the room.

Considered requirements :

STEP 1 :
Action Light of the room is off.

Expected result : Value for the switch is "off" in sensiNact platform.

• Attachments : None
• Requirement : None

STEP 2 :
Action Set the value of the switch to "on" in sensiNact platform.

Expected result : Light of the room is on.

• Attachments : None
D5.3 Intermediate Validation Results

**sNa_Switch Bedroom**

*Created on:* 19 janv. 2018 16:45 (by claire)

*Modified on:* 19 janv. 2018 16:45 (by claire)

*ID:* 35

*Nature:* Functional

*Type:* Recevabilité

*Status:* Under edition

*Importance:* Very high

**Description**

Test case for switch of bedroom in sNa.

*Pré-requis:*

Services Eclipse sensiNact and OpenHab are started.

In OpenHab, sensor binary must be set for the device.

Light is off in the room.

**Considered requirements:**

**STEP 1:**

*Action:* Light of the room is off.

*Expected result:* Value for the switch is "off" in sensiNact platform.

*Attachments:* None

*Requirement:* None

**STEP 2:**

*Action:* Set the value of the switch to "on" in sensiNact platform.

*Expected result:* Light of the room is on.

*Attachments:* None

*Requirement:* None
3-Network
Test on network

1-Box connexion

Created on : 26 janv. 2018 17:06 (by claire)
Modified on : 26 janv. 2018 17:07 (by claire)
ID: 67
Nature : Utilisateur
Type: Recevabilité
Status : Under edition
Importance: Very high
Milestones:

Description
Test connection of the box to the network

Pré-requis :

Considered requirements :
STEP 1 :

Action From server, ping the box.

Expected result : Box is responding.

• Attachments : None
• Requirement : None

4-Interfaces > 1-Parameters
Tests cases on parameters interfaces

Global default settings
Created on : 21 févr. 2018 17:02 (by thomas)
Modified on : 21 févr. 2018 17:20 (by thomas)
ID: 77
Nature : Utilisateur
Type: Recevabilité
Status : Under edition
** Importance: ** Moyenne

** Milestones:**

** Description **
Test case for settings default parameters for all functions in one room.

** Pré-requis :**

** Considered requirements :**

- Activage Panel 3
  - Global parameters (MAJOR)
  - Reset building (MINOR)

** STEP 1 :**

| Action | Go into main page of room "studio-PTL" (under Résident tab) and select "Nettoyer données suivi" then yes. It will reset by default all parameters of the studio by default.
|        | Go into the studio parameters interface.

** Expected result :**
The value displayed on the interface is identical to corresponding value stored into database and are the one by default.

- Attachments: None
- Requirement: None

** Global parameters building **

| Created on | 2 févr. 2018 11:36 (by claire) |
| Modified on | 2 févr. 2018 12:02 (by claire) |
| ID | 71 |
| Nature | Utilisateur |
| Type | Recevabilité |
| Status | Under edition |
| Importance | Very high |

** Description **
Test case for reading and writing of parameters global to several functions for the building.

** Pré-requis :**
Considered requirements:

- Activage Panel 3
  - Global parameters (MAJOR)

**STEP 1**

<table>
<thead>
<tr>
<th>Action</th>
<th>Open parameters interface, on building page, look at the value of parameter for <code>{parameter}</code>.</th>
</tr>
</thead>
</table>

**Expected result:** The value displayed on the interface is identical to corresponding value stored into database.

- Attachments: None
- Requirement: None

**STEP 2**

| Action | On parameters interface, change value of parameter for `{parameter}`.  
Press button "Validate". |
|--------|--------------------------------------------------------------------------|

**Expected result:** The corresponding value stored into database has changed to the chosen value.

- Attachments: None
- Requirement: None

**STEP 3**

<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, for each studio of the building, open its page and look, for each function that use the parameter for <code>{parameter}</code>, at its value.</th>
</tr>
</thead>
</table>

**Expected result:** Values displayed on the interface are identical to the chosen value.

- Attachments: None
- Requirement: None
Global parameters function

Created on: 2 févr. 2018 11:29 (by claire)
Modified on: 2 févr. 2018 12:02 (by claire)
ID: 70
Nature: Utilisateur
Type: Recevabilité
Status: Under edition
Importance: Very high
Milestones:

Description
Test case for reading and writing of parameters of a function global to all studios of a building.

Pre-requis: 

Considered requirements:
- Activage Panel 3
  - Functions parameters (MAJOR)

**STEP 1:**
<table>
<thead>
<tr>
<th>Action</th>
<th>Open parameters interface, on building page, look at the value of parameter for ${parameter}.</th>
</tr>
</thead>
</table>

Expected result: The value displayed on the interface is identical to corresponding value stored into database.

- Attachments: None
- Requirement: None

**STEP 2:**
<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, change value of parameter for ${parameter}. Press button &quot;Validate&quot;.</th>
</tr>
</thead>
</table>

Expected result: The corresponding value stored into database has changed to the chosen value.
STEP 3:

| Action | On parameters interface, for each studio of the building, open its page and look at the value of parameter for $\text{parameter}$.

---

| Expected result | The value displayed on the interface is identical to the chosen value.

---

Local default settings

- Created on: 21 févr. 2018 17:04 (by thomas)
- Modified on: 21 févr. 2018 17:17 (by thomas)
- ID: 78
- Nature: Utilisateur
- Type: Recevabilité
- Status: Under edition
- Importance: Moyenne

Description

Test case for reading and writing of parameters global to several functions for the building.

Pre-requis:

Considered requirements:

- Activage Panel 3
  - Functions parameters (MAJOR)
  - Reset studio (MINOR)

STEP 1:

| Action | Open parameters interface, on building page, look at the value of parameter for $\text{parameter}$.

---
### Expected result:
The value displayed on the interface is identical to corresponding value stored into database.

- Attachments: None
- Requirement: None

**STEP 2:**

| Action | On parameters interface, change value of parameter for \$\{parameter\}. Press button "Validate". |

### Expected result:
The corresponding value stored into database has changed to the chosen value.

- Attachments: None
- Requirement: None

**STEP 3:**

| Action | On parameters interface, for each studio of the building, open its page and look, for each function that use the parameter for \$\{parameter\}, at its value. |

### Expected result:
Values displayed on the interface are identical to the chosen value.

- Attachments: None
- Requirement: None

---

**Local parameters function**

- Created on: 19 janv. 2018 10:23 (by claire)
- Modified on: 2 févr. 2018 12:02 (by claire)
- ID: 26
- Nature: Utilisateur
- Type: Recevabilité
- Status: Under edition
- Importance: Very high

---
Description
Test case for reading and writing of local parameters of function

Pré-requis :

Considered requirements :

- Activage Panel 3
  - Functions parameters (MAJOR)

STEP 1 :

<table>
<thead>
<tr>
<th>Action</th>
<th>Open parameters interface, on studio page, look at the value of parameter for $\text{parameter}$.</th>
</tr>
</thead>
</table>

Expected result :
The value displayed on the interface is identical to corresponding value stored into database.

Attachments : None
Requirement : None

STEP 2 :

<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, change value of parameter for $\text{parameter}$. Press button &quot;Validate&quot;.</th>
</tr>
</thead>
</table>

Expected result :
The corresponding value stored into database has changed and is identical to the chosen value.

Attachments : None
Requirement : None

4-Interfaces > 3-Notification
Test case for notification that is triggered on the e-lio application
Notification walking area

Created on : 22 févr. 2018 11:30 (by thomas)
Modified on : 22 févr. 2018 13:46 (by thomas)
ID: 84
Nature: Functional
Type: Recevabilité
Status: Under edition
Importance: Very high
Milestones:

Description
Integration test for area of walking.

Pré-requis:
sNa is running on the local gateway.
The pedometer service is running on the local gateway. "systemctl status pedometer.service"
Service Eclipse sensiNact is started on the server.
Test on sensors and parameters interface were passed successfully.
On parameters interface :
  • Status parameter for the function "pedometer' is set to "active"
  • Status parameters of all other functions are set to "inactive"

Considered requirements:

  • Activage Panel 3
    o - Walking area on (MAJOR)
    o - Functions parameters (MAJOR)
    o - pedometer service (MINOR)
    o - D_Walking area (MAJOR)
    o - N_Appearance (CRITICAL)
    o - N_Acknoledgement (CRITICAL)

STEP 1:
Action  Set the time of "Heure charge podomètre" to 5 minutes after the actual time
### Expected result:
Check 5 minutes later whether the notification saying "Avez-vous pensz à mettre votre podomètre" triggers on the screen. Check for acquit-notification in the database to validate the API has received it from the application

<table>
<thead>
<tr>
<th>Attachments :</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

### STEP 2:

**Action**
Answer "NO" an wait for 15 minutes

### Expected result :
After 15 minutes, a new notification triggers

<table>
<thead>
<tr>
<th>Attachments :</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

### STEP 3:

**Action**
Don't answer and wait for X min before the notification disappears

**Expected result:**
The notificaiton is gone and the acquit-notification to false is raised in the database

<table>
<thead>
<tr>
<th>Attachments :</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

### STEP 4:

**Action**
Wait for another 15 minutes untill the last notification triggers and answer "OUI"

**Expected result:**
The notification is acquitted in the database and dissappears from the screen

<table>
<thead>
<tr>
<th>Attachments :</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

### STEP 5:

**Action**
Set parameters "Heure port podomètre" to 5 min after the actual time

<table>
<thead>
<tr>
<th>Attachments :</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

Version 1.0  I  2019-02-14  I  ACTIVAGE ©
D5.3 Intermediate Validation Results

<table>
<thead>
<tr>
<th>Expected result</th>
<th>The notification saying &quot;Avez-vous pensez à charger votre podomètre&quot; should pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachments</td>
<td>None</td>
</tr>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

**STEP 6:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Repeat test from 2 to 4</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Expected result</th>
<th>Observe the same results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachments</td>
<td>None</td>
</tr>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

5-Functions > Area of walking

Synchronize the system whenever the device is available on the air.

Check for notification on the screen to charge and wear the device during the day.

**Integration walking area**

- **Created on:** 21 févr. 2018 18:01 (by thomas)
- **Modified on:** 22 févr. 2018 13:45 (by thomas)
- **ID:** 79
- **Nature:** Functional
- **Type:** Recevabilité
- **Status:** Under edition
- **Importance:** Very high
- **Milestones:**

**Description**

Integration test for area of walking.

**Pré-requis:**

- sNa is running on the local gateway.
- The pedometer service is running on the local gateway. "systemctl status pedometer.service"
- Service Eclipse sensiNact is started on the server.
D5.3 Intermediate Validation Results

Test on sensors and parameters interface were passed successfully.

On parameters interface:

- Status parameter for the function "pedometer' is set to "active"
- Status parameters of all other functions are set to "inactive"

Considered requirements:

- Activage Panel 3
  - Functions parameters (MAJOR)
  - Walking area on (MAJOR)
  - D_Walking area (MAJOR)
  - pedometer service (MINOR)

STEP 1:

| Action | Wear the bracelet for all day and come in the room in order to synchronize the data of walking area. Data will be synchronized every 5 min |

Expected result:

Check once you come in the room on the dashboard the number of step you have done

- Attachments: None
- Requirement: None

STEP 2:

| Action | Go take a walk for 5 minutes and come back into the room to synchronize the data of walking area |

Expected result:

Verify that the data on the dashboard is updated and takes into account the number of step you have done within this 5 last minutes
Robustness walking area

Integration test for area of walking.

**Pré-requis :**

- sNa is running on the local gateway.
- The pedometer service is running on the local gateway. "systemctl status pedometer.service"
- Service Eclipse sensiNact is started on the server.
- Test on sensors and parameters interface were passed successfully.
- On parameters interface :
  - Status parameter for the function "pedometer" is set to "active"
  - Status parameters of all other functions are set to "inactive"

**Considered requirements :**

- Activage Panel 3
  - Functions parameters (MAJOR)
  - Walking area on (MAJOR)
  - D_Walking area (MAJOR)
  - pedometer service (MINOR)

**STEP 1 :**

<table>
<thead>
<tr>
<th>Action</th>
<th>Let sNa and the pedometer service running for 2 weeks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Synchronize at least 2 to 3 times the value with the system and check whether the value is updated</td>
</tr>
</tbody>
</table>
**Expected result:**
The results should be displayed and updated every time you come by the room for every day. If the function status is OFF, no data is displayed.

- Attachments: None
- Requirement: None

**Status walking area**

*Created on:* 22 févr. 2018 10:54 (by thomas)

*Modified on:* 22 févr. 2018 13:45 (by thomas)

*ID:* 82

*Nature:* Functional

*Type:* Recevabilité

*Status:* Under edition

*Importance:* Very high

**Milestones:**

**Description**
Integration test for area of walking.

*Pré-requis:*

sNa is running on the local gateway.

The pedometer service is running on the local gateway. "systemctl status pedometer.service"

Service Eclipse sensiNact is started on the server.

Test on sensors and parameters interface were passed successfully.

On parameters interface:

- Status parameter for the function "pedometer" is set to "active"
- Status parameters of all other functions are set to "inactive"

**Considered requirements:**

- Activage Panel 3
  - Functions parameters (MAJOR)
  - pedometer service (MINOR)
  - D_Walking area (MAJOR)
  - Walking area on (MAJOR)
<table>
<thead>
<tr>
<th>STEP 1</th>
<th><strong>Action</strong> Set the status of the function to ON on the parameters menu of the e-linoManager</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected result</strong></td>
<td>Status of the function is ON</td>
</tr>
<tr>
<td>Attachments</td>
<td>None</td>
</tr>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STEP 2</th>
<th><strong>Action</strong> Wear the bracelet for all day and come in the room in order to synchronize the data of walking area. Data will be synchronized every 5 min</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected result</strong></td>
<td>Check once you come in the room on the dashboard the number of step you have done</td>
</tr>
<tr>
<td>Attachments</td>
<td>None</td>
</tr>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STEP 3</th>
<th><strong>Action</strong> Go take a walk for 5 minutes and come back into the room to synchronize the data of walking area</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected result</strong></td>
<td>Verify that the data is not displayed on the dashboard</td>
</tr>
<tr>
<td>Attachments</td>
<td>None</td>
</tr>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STEP 4</th>
<th><strong>Action</strong> Go take a walk for 5 minutes and come back into the room to synchronize the data of walking area</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected result</strong></td>
<td>Verify that the data on the dashboard is updated and takes into account the number of step you have done within this 5 last minutes</td>
</tr>
<tr>
<td>Attachments</td>
<td>None</td>
</tr>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>
D5.3 Intermediate Validation Results

- Attachments: None
- Requirement: None

**STEP 5:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Set the status of the function to OFF on the parameters menu of the e-lioManager</th>
</tr>
</thead>
</table>

**Expected result:** Status of the function is OFF

- Attachments: None
- Requirement: None

5-Functions > Automatic light

Test cases for "automatic light" function

**Integration automatic light**

- Created on: 23 janv. 2018 09:46 (by thomas)
- Modified on: 22 févr. 2018 13:45 (by thomas)
- ID: 37
- Nature: Functional
- Type: Recevabilité
- Status: Under edition
- Importance: Very high

**Milestones:**

**Description**

Integration test for automatic light during night.

**Pré-requis:**

- sNa and Openhab2 are running on the local gateway.
- The rule for automatic light is running on OH2.
- Service Eclipse sensiNact is started on the server.
- Test on sensors, actuators and parameters interface were passed successfully.

On parameters interface:

- Status parameter for the function "automatic light" is set to "active"
- Parameter "night end" must be set to "06:00"
- Parameter "night start" must be set to "20:00"
### Status parameters of all other functions are set to "inactive"

Bedroom and bathroom lights are off.

### Considered requirements:

- Activage Panel 3
  - Automatic light off (MAJOR)
  - Automatic light on (MAJOR)
  - Functions parameters (MAJOR)
  - Global parameters (MAJOR)

#### STEP 1:

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lie on the bed and wait for 60s.</td>
</tr>
<tr>
<td>Get up and wait for 10s.</td>
</tr>
</tbody>
</table>

**Expected result:**

Check on OpenHab log that the rule wasn't executed with this command: `tail -f /var/log/openhab2/events.log`

Both light in bedroom and bathroom are off.

- Attachments: None
- Requirement: None

#### STEP 2:

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch on lights of bedroom and bathroom.</td>
</tr>
<tr>
<td>Turn off the switch of bedroom's light.</td>
</tr>
</tbody>
</table>

**Expected result:**

Only bedroom's light is off.

- Attachments: None
- Requirement: None

#### STEP 3:

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch off the bathroom's light.</td>
</tr>
<tr>
<td>On parameters interface, set parameter &quot;night end&quot; to &quot;20:00&quot; and &quot;night start&quot; to &quot;06:00&quot;.</td>
</tr>
</tbody>
</table>

- Attachments: None
- Requirement: None
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Lie on the bed and wait for 60s. Get up and wait for 10s.</td>
<td>Check on OpenHab log that the rule has been executed with this command: <code>tail -f /var/log/openhab2/events.log</code> Both lights in bedroom and bathroom turn on.</td>
</tr>
<tr>
<td>Attachments:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Requirement:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>STEP 4:</strong></td>
<td></td>
<td><strong>STEP 5:</strong></td>
</tr>
<tr>
<td>Action</td>
<td>Once you are up, turn off the switch of the bedroom.</td>
<td>On parameters interface, set parameter &quot;night end&quot; to &quot;20:00&quot; and &quot;night start&quot; to &quot;06:00&quot;. Lie on the bed and wait for 60s. Get up and wait for 10s.</td>
</tr>
<tr>
<td>Attachments:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Requirement:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Expected result:</td>
<td>Both lights (bedroom + bathroom) has to turn off.</td>
<td>Check on OpenHab log that the rule has been executed with this command: <code>tail -f /var/log/openhab2/events.log</code> Both lights in bedroom and bathroom turn on.</td>
</tr>
<tr>
<td>Attachments:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Requirement:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>STEP 6:</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Expected result:

Light of bathroom turns off.

Light of bedroom stays on.

• Attachments: None
• Requirement: None

**STEP 7:**

| Action | Turn off the switch of the bedroom light. |

Expected result:

Bedroom’s light is off.

• Attachments: None
• Requirement: None

**Robustness automatic light**

Created on: 23 janv. 2018 09:59 (by thomas)
Modified on: 22 févr. 2018 13:45 (by thomas)
ID: 39
Nature: Non défini
Type: Non défini
Status: Under edition
Importance: Very high
Milestones:

**Description**
Test the robustness of the function automatic lightning at night.

**Pré-requis:**

sNa and Openhab2 are running on the local gateway.

The rule for automatic light is running on OH2.

Service Eclipse sensiNact is started on the server.

Test on sensors, actuators and parameters interface were passed successfully.

On parameters interface:

- Status parameter for the function "automatic light" is set to "active"
- Parameter "night end" must be set to "06:00"
- Parameter "night start" must be set to "20:00"
- Status parameters of all other functions are set to "active"

Bedroom and bathroom lights are off.

**Considered requirements**:

- Activage Panel 3
  - Acknowledgement (CRITICAL)
  - Appearance (CRITICAL)
  - Functions parameters (MAJOR)
  - Global parameters (MAJOR)
  - Automatic light off (MAJOR)
  - Automatic light on (MAJOR)

**STEP 1**:

| Action | Let sNa and OH2 running for 2 weeks.  
Test the integration automatic light every day at least 2 times. |
|--------|------------------------------------------------------------------|

**Expected result**:
The results should be the same as observed in the integration test

- Attachments: None
- Requirement: None

**Status automatic light**

- Created on: 25 janv. 2018 14:13 (by claire)
- Modified on: 22 févr. 2018 13:45 (by thomas)
- ID: 62
- Nature: Functional
- Type: Recevabilité
- Status: Under edition
- Importance: Very high

**Description**
Test whether the status enables or disables the function.

**Pré-requis**:
sNa and Openhab2 are running on the local gateway.
The rule for automatic light is running on OH2.
Service Eclipse sensiNact is started on the server.

Test on sensors, actuators and parameters interface were passed successfully.

On parameters interface:

- Status parameter for the function "automatic light" is set to "active"
- Parameter "night end" must be set to "20:00"
- Parameter "night start" must be set to "6:00"
- Status parameters of all other functions are set to "inactive"

Bedroom and bathroom lights are off.

**Considered requirements:**

- Activage Panel 3
  - Global parameters (MAJOR)
  - Functions parameters (MAJOR)
  - Automatic light off (MAJOR)
  - Automatic light on (MAJOR)
  - Status function (MINOR)

**STEP 1:**

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lie on the bed and wait for 60s.</td>
</tr>
<tr>
<td>Get up and wait for 10s.</td>
</tr>
</tbody>
</table>

**Expected result:**

Check on OpenHab log that the rule has been executed with this command: `tail -f /var/log/openhab2/events.log`

Both light in bedroom and bathroom turn on.

**Attachments:** None

**Requirement:** None

**STEP 2:**

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Once you are up, turn off the switch of the bedroom.</td>
</tr>
</tbody>
</table>

**Expected result:**

Both lights (bedroom + bathroom) are off.

**Attachments:** None
D5.3 Intermediate Validation Results

- Requirement: None

**STEP 3:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set parameter &quot;Status&quot; of function to &quot;inactive&quot;.</td>
</tr>
<tr>
<td></td>
<td>Lie on the bed and wait for 60s.</td>
</tr>
<tr>
<td></td>
<td>Get up and wait for 10s.</td>
</tr>
</tbody>
</table>

- **Expected result:**

Check on OpenHab log that the rule wasn’t executed with this command: `tail -f /var/log/openhab2/events.log`

Both light in bedroom and bathroom are off.

- Attachments: None
- Requirement: None

**5-Functions > Bed statistics**

Test cases for "automatic light" function

**Integration bed statistics**

- **Created on:** 25 janv. 2018 14:37 (by claire)
- **Modified on:** 25 janv. 2018 16:33 (by claire)
- **ID:** 63
- **Nature:** Functional
- **Type:** Recevabilité
- **Status:** Under edition
- **Importance:** Very high
- **Milestones:**

**Description**

Integration test for bed statistics.

**Pré-requis:**

sNa and Openhab2 are running on the local gateway.

The rule for automatic light is running on OH2.

Service Eclipse sensiNact is started on the server.

Test on sensors, actuators and parameters interface were passed successfully.

On parameters interface:
• Status parameter for the function "bed statistics" is "active"
• Parameter "night end" must be set to "06:00"
• Parameter "night start" must be set to "20:00"
• Status parameters of all other functions are set to "inactive"

Nobody was on the bed during the current day.

Considered requirements:

- Activage Panel 3
  - Functions parameters (MAJOR)
  - Global parameters (MAJOR)

**STEP 1:**

<table>
<thead>
<tr>
<th>Action</th>
<th>From the dashboard, note:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• the number of night rising for the previous night</td>
</tr>
<tr>
<td></td>
<td>• the total duration of night rising for the previous night</td>
</tr>
<tr>
<td></td>
<td>• the total duration of day lying for the previous day</td>
</tr>
</tbody>
</table>

**Expected result:**
The number and total duration of night rising for the previous night should be null.

- Attachments: None
- Requirement: None

**STEP 2:**

<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, set parameter &quot;night end&quot; to the current time plus 12min.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lie on the bed and wait for 2min.</td>
</tr>
<tr>
<td></td>
<td>Get up and wait 5min.</td>
</tr>
<tr>
<td></td>
<td>Lie on the bed and wait for 60s.</td>
</tr>
<tr>
<td></td>
<td>Get up and wait 1 min.</td>
</tr>
<tr>
<td></td>
<td>Lie on the bed and wait to the time of the end of the night.</td>
</tr>
</tbody>
</table>
## Expected result:

### On dashboard:
- the number of night rising for the previous night is 2.
- the total duration of night rising for the previous night is around 6 min.
- History of night rising show a duration in bed of 5 min between the two risings.
- the total duration of day lying for the previous day is identical as the one noticed at the beginning of the test case.

### Step 3:

<table>
<thead>
<tr>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lie on the bed and wait for 2 min.</td>
</tr>
<tr>
<td>Get up and wait 5 min.</td>
</tr>
<tr>
<td>Lie on the bed and wait for 3 min.</td>
</tr>
<tr>
<td>Get up.</td>
</tr>
<tr>
<td>Nobody must lie again on the bed for the current day and the next night.</td>
</tr>
<tr>
<td>The day after, wait the time of the end of the night.</td>
</tr>
</tbody>
</table>

## Expected result:

### On dashboard:
- the number of night rising for the previous night is 0.
- the total duration of night rising for the previous night is around 0 min.
- History of night presence shows no presence for the whole night.
- the total duration of day lying for the previous day is between 3 and 5 min.

### Robustness bed statistics

| Created on | 23 janv. 2018 11:19 (by thomas) |
| Modified on | 23 janv. 2018 14:33 (by thomas) |
| ID | 54 |
| Nature | Non définie |
| Type | Non défini |
D5.3 Intermediate Validation Results

**Status:** Under edition

**Importance:** Very high

**Milestones:**

**Description**
This function is to test the robustness of the function automatic lightning at night.

**Pré-requis:**
- sNa and Openhab2 are running.
- Bed sensor is communicating with OH2.
- The bundle is active in sNa for several days without interruption.

**Considered requirements:**
- Activage Panel 3
  - Night and day activity (MAJOR)
  - D_Night and day activity (MAJOR)
  - Global parameters (MAJOR)
  - Status function (MINOR)

**STEP 1:**

**Action**
Let sNa and OH2 running for 2 weeks.
Test the data gathered every day in the morning and compare to what was noted the day before.

**Expected result:**
The results should be the same as observed the day before with the lowest error possible.

- **Attachments:** None
- **Requirement:** None
D5.3 Intermediate Validation Results

Description
This function is to test whether the status enable of disable the function.

 Pré-requis :
- sNa and Openhab2 are running.
- Bed sensor is communicating with OH2.
- The bundle is active in sNa.

Considered requirements :

- Activage Panel 3
  - Global parameters (MAJOR)
  - Status function (MINOR)
  - D_Night and day activity (MAJOR)
  - Night and day activity (MAJOR)

STEP 1 :

<table>
<thead>
<tr>
<th>Action</th>
<th>Turn off the status of the function and repeat what is done in the &quot;Integration bed statistics&quot;:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lie on the bed and wait until the sensor bed detects you. Then, the watch starts and stay for 30s.</td>
</tr>
<tr>
<td></td>
<td>Get up and wait until the bed sensor detects you out of the bed.</td>
</tr>
</tbody>
</table>

Expected result :
- Time spend in bed is not incremented.
- Number of rise up is not incremented.

- Attachments : None
- Requirement : None

STEP 2 :

<table>
<thead>
<tr>
<th>Action</th>
<th>Turn on the status of the function and repeat what it is done in the &quot;Integration bed statistics&quot;:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lie on the bed and wait until the sensor bed detects you. Then, the watch starts and stay for 30s.</td>
</tr>
<tr>
<td></td>
<td>Get up and wait until the bed sensor detects you out of the bed.</td>
</tr>
</tbody>
</table>
### Expected result:
- Time spent in bed is incremented.
- Number of rise up is incremented.

- Attachments: None
- Requirement: None

## 5-Functions > Shower alert

Test cases for "automatic light" function

### Integration shower alert

<table>
<thead>
<tr>
<th>Created on</th>
<th>23 janv. 2018 11:19 (by claire)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified on</td>
<td>23 janv. 2018 11:34 (by claire)</td>
</tr>
<tr>
<td><strong>ID:</strong></td>
<td>48</td>
</tr>
<tr>
<td><strong>Nature:</strong></td>
<td>Functional</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Recevabilité</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Under edition</td>
</tr>
<tr>
<td><strong>Importance:</strong></td>
<td>Very high</td>
</tr>
<tr>
<td><strong>Milestones:</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### Description
Integration test for "shower alert" function.

#### Pré-requis:
- Services Eclipse sensiNact and OpenHab are started on the local gateway.
- Service Eclipse sensiNact is started on the server.
- Test on sensors and parameters interface were passed successfully.

On parameters interface:
- Shower duration for alert is set to 20s.
- Status parameter for the function "shower alert" is set to "active"
- Status parameters of all other functions are set to "inactive"

#### Considered requirements:
- Activage Panel 3
  - Shower alert (MAJOR)
  - Acknowledgement (CRITICAL)
  - Appearance (CRITICAL)
  - D_Shower alert (MAJOR)
  - hydrao service (MINOR)
## D5.3 Intermediate Validation Results

- pedometer service (MINOR)

### STEP 1:

<table>
<thead>
<tr>
<th>Action</th>
<th>On dashboard interface, note the rate of proven shower alerts. In database, note the time of the last shower alert and the time of the last acknowledgement of shower alert.</th>
</tr>
</thead>
</table>

**Expected result:** Check in database that the stored rate of proven shower alerts is identical

- Attachments: None
- Requirement: None

### STEP 2:

<table>
<thead>
<tr>
<th>Action</th>
<th>Run shower for 10s and stop.</th>
</tr>
</thead>
</table>

**Expected result:** The value of resource "shower" from service "alert" of provider "shower alert" of API is "false". No shower alert appears on the nurse tablet for the corresponding studio. In database, no new shower alert is stored.

- Attachments: None
- Requirement: None

### STEP 3:

<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, set shower duration for alert to 5s Run shower for 10s and stop.</th>
</tr>
</thead>
</table>

**Expected result:** The value of resource "shower" from service "alert" of provider "shower alert" of API is "true". Shower alert appears on the nurse tablet for the corresponding studio. In database, a new shower alert is stored.

- Attachments: None
- Requirement: None
### STEP 4:

**Action**

Run shower for 10s and stop.

**Expected result:**

The value of resource "shower" from service "alert" of provider "shower alert" of API is "true".

No new shower alert appears on the nurse tablet for the corresponding studio.

In database, no new shower alert is stored.

- **Attachments:** None
- **Requirement:** None

### STEP 5:

**Action**

On nurse tablet, click on the shower alert and select unapproved alert.

**Expected result:**

The value of resource "acquit_alert" from service "control" of provider "shower alert" of API is "false".

The value of resource "shower" from service "alert" of provider "shower alert" of API is "false".

On nurse tablet, shower alert becomes inactive or disappears.

On dashboard interface, the rate of proven shower alerts has decreased.

In database, a new acknowledgement for shower alert is stored and value is "false".

- **Attachments:** None
- **Requirement:** None

---

**Robustness shower alert**

*Created on:* 23 janv. 2018 10:57 (by claire)

*Modified on:* 23 janv. 2018 11:21 (by claire)

*ID:* 45

*Nature:* Functional

*Type:* Recevabilité

*Status:* Under edition

*Importance:* Very high
Milestones:

Description
Test the robustness of the function "shower alert".

Pré-requis :

- Services Eclipse sensiNact and OpenHab are started on the local gateway.
- Service Eclipse sensiNact is started on the server.
- Test on sensors, parameters interface and integration test for function "shower alert" were passed successfully.
- On parameters interface :
  - Shower duration for alert is set to 5s.
  - Status parameter for the function "shower alert" is set to "active"

Considered requirements :

- Activage Panel 3
  - Shower alert (MAJOR)
  - Appearance (CRITICAL)
  - hydrao service (MINOR)
  - Status function (MINOR)
  - pedometer service (MINOR)

STEP 1 :

Action
Let sNa and OH2 running for 2 weeks.
Run the test "Global shower alert" every day at least 3 times.

Expected result :
At each try, results should be the same as observed in the Integration test.

Status shower alert

Created on : 23 janv. 2018 10:57 (by claire)
Modified on : 24 janv. 2018 16:30 (by claire)
ID : 47
Nature : Functional
Type : Recevabilité
Status : Under edition
Description
Test whether the status enables or disables the function.

Pré-requis:
- Services Eclipse sensiNact and OpenHab are started on the local gateway.
- Service Eclipse sensiNact is started on the server.
- Test on sensors, parameters interface and integration test for function "shower alert" were passed successfully.
- On parameters interface:
  - Shower duration for alert is set to 5s.
  - Status parameter for the function "shower alert" is set to "active"

Considered requirements:
- Activage Panel 3
  - Shower alert (MAJOR)
  - Appearance (CRITICAL)
  - Status function (MINOR)

STEP 1:
<table>
<thead>
<tr>
<th>Action</th>
<th>Run shower for 10s and stop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected result</td>
<td>Shower alert appears on nurse tablet for the corresponding studio.</td>
</tr>
<tr>
<td>Attachments</td>
<td>None</td>
</tr>
<tr>
<td>Requirement</td>
<td>None</td>
</tr>
</tbody>
</table>

STEP 2:
| Action                  | On the nurse tablet, click on the alert and acknowledge it. On parameters interface, set the status of the function "shower alert" to "inactive". |
| Expected result         | On parameters interface, other parameters of function dissapear exept the status. |
D5.3 Intermediate Validation Results

- Attachments: None
- Requirement: None

**STEP 3:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Run shower for 10s and stop.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Expected result</th>
<th>No shower alert appears on nurse tablet for the corresponding studio.</th>
</tr>
</thead>
</table>

- Attachments: None
- Requirement: None

5-Functions > Temperature alert

Test cases for "automatic light" function

Integration temperature alert

- Created on: 24 janv. 2018 16:35 (by claire)
- Modified on: 24 janv. 2018 17:22 (by claire)
- ID: 58
- Nature: Functional
- Type: Recevabilité
- Status: Under edition
- Importance: Very high
- Milestones:

**Description**

Integration test for "temperature alert" function.

**Pré-requis:**

Services Eclipse sensiNact and OpenHab are started on the local gateway.

Service Eclipse sensiNact is started on the server.

Test on sensors and parameters interface were passed successfully.

On parameters interface:

- Status parameter for the function "temperature alert" is set to "active".
- Parameter "Temperature max" is set to 32°C
- Parameter "Temperature max" is set to 16°C
- Parameter "Delay between two alerts" is set to 5 min.
- Status parameters of all other functions are set to "inactive"
Considered requirements:

- Activage Panel 3
  - Appearance (CRITICAL)
  - Acknowledgement (CRITICAL)
  - Temperature control (MAJOR)
  - Functions parameters (MAJOR)
  - netatmo service (MINOR)

**STEP 1:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Note the value of the current temperature in the room.</th>
</tr>
</thead>
</table>

**Expected result:**

- No temperature alert appears on the nurse tablet for the corresponding studio.

- Attachments: None
- Requirement: None

**STEP 2:**

<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, set parameter &quot;Temperature min&quot; to a value higher than the current temperature.</th>
</tr>
</thead>
</table>

**Expected result:**

- The value of resource "temperature" from service "alert" of provider "temperature alert" of API is "true".
- Temperature alert appears on the nurse tablet for the corresponding studio.
- In database, a new temperature alert is stored.

- Attachments: None
- Requirement: None

**STEP 3:**

<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, set parameter &quot;Temperature min&quot; to 16°C.</th>
</tr>
</thead>
</table>

**Expected result:**

- The value of resource "temperature" from service "alert" of provider "temperature alert" of API is still "true".
- Temperature alert is still on the nurse tablet for the corresponding studio.
### D5.3 Intermediate Validation Results

- **Attachments:** None
- **Requirement:** None

#### STEP 4:

<table>
<thead>
<tr>
<th>Action</th>
<th>On parameters interface, set parameter &quot;Temperature max&quot; to a value lower than the current temperature.</th>
</tr>
</thead>
</table>

**Expected result:**
- The value of resource "temperature" from service "alert" of provider "temperature alert" of API is still "true".
- No new temperature alert appears on the nurse tablet for the corresponding studio.
- In database, no new temperature alert is stored.

- **Attachments:** None
- **Requirement:** None

#### STEP 5:

<table>
<thead>
<tr>
<th>Action</th>
<th>On nurse tablet, click on the temperature alert and acknowledge it as a approved alert.</th>
</tr>
</thead>
</table>

**Expected result:**
- The value of resource "acquit_alert" from service "control" of provider "temperature alert" of API is "true".
- The value of resource "temperature" from service "alert" of provider "temperature alert" of API is "false".
- On nurse tablet, temperature alert becomes inactive or disappears.
- In database, a new acknowledgement for temperature alert is stored and its value is "true".

- **Attachments:** None
- **Requirement:** None

#### STEP 6:

<table>
<thead>
<tr>
<th>Action</th>
<th>Wait for 5 minutes.</th>
</tr>
</thead>
</table>
### Expected result:

The value of resource "temperature" from service "alert" of provider "temperature alert" of API is "true".

Temperature alert appears on the nurse tablet for the corresponding studio.

In database, a new temperature alert is stored.

- Attachments: None
- Requirement: None

### STEP 7:

**Action**

On nurse tablet, click on the temperature alert and acknowledge it as an unapproved alert.

### Expected result:

The value of resource "acquit_alert" from service "control" of provider "temperature alert" of API is "false".

The value of resource "temperature" from service "alert" of provider "temperature alert" of API is "false".

On nurse tablet, temperature alert becomes inactive or disappears.

In database, a new acknowledgement for temperature alert is stored and its value is "false".

- Attachments: None
- Requirement: None

### Robustness temperature alert

*Created on*: 24 janv. 2018 16:35 (by claire)

*Modified on*: 25 janv. 2018 09:40 (by claire)

*ID*: 59

*Nature*: Functional

*Type*: Recevabilité

*Status*: Under edition

*Importance*: Very high

*Description*

Test the robustness of the function "temperature alert".
D5.3 Intermediate Validation Results

**Pré-requis :**

Services Eclipse sensiNact and OpenHab are started on the local gateway.

Service Eclipse sensiNact is started on the server.

Test on sensors, parameters interface and integration test for function "temperature alert" were passed successfully.

On parameters interface :

- Status parameter for the function "temperature alert' is set to "active"
- Parameter "Temperature max" is set to 32°C
- Parameter "Temperature max" is set to 16°C
- Parameter "Delay between two alerts" is set to 5 min.
- Status parameters of all other functions are set to "active"

**Considered requirements :**

- Activage Panel 3
  - Appearance (CRITICAL)
  - Temperature control (MAJOR)
  - Acknowledgement (CRITICAL)
  - Functions parameters (MAJOR)
  - netatmo service (MINOR)

**STEP 1 :**

**Action**

Let sNa and OH2 running for 2 weeks.

Run the test "Global temperature alert" every day at least 3 times.

**Expected result:**

At each try, results should be the same as observed in the Integration test.

- Attachments : None
- Requirement : None

**Status temperature alert**

*Created on :* 25 janv. 2018 09:43 (by claire)

*Modified on :* 25 janv. 2018 09:43 (by claire)

*ID:* 61

*Nature :* Functional

*Type:* Recevabilité
D5.3 Intermediate Validation Results

Status: Under edition
Importance: Very high
Milestones:

Description
Test whether the status enables or disables the function.

Pré-requis:
Services Eclipse sensiNact and OpenHab are started on the local gateway.
Service Eclipse sensiNact is started on the server.
Test on sensors and parameters interface were passed successfully.
On parameters interface:
- Status parameter for the function "temperature alert' is set to "active"
- Parameter "Temperature max" is set to 32°C
- Parameter "Temperature max" is set to 16°C
- Parameter "Delay between two alerts" is set to 5 min.
- Status parameters of all other functions are set to "inactive"

Considered requirements:
- Activage Panel 3
  - Functions parameters (MAJOR)
  - Acknowledgement (CRITICAL)
  - Temperature control (MAJOR)
  - Appearance (CRITICAL)
  - Status function (MINOR)
  - netatmo service (MINOR)

STEP 1:

Action

Note the value of the current temperature in the room.
On parameters interface, set parameter "Status" for the function to "inactive".

Expected result:
No temperature alert appears on the nurse tablet for the corresponding studio.

Attachments: None
Requirement: None
D5.3 Intermediate Validation Results

**STEP 2 :**

| Action: On parameters interface, set parameter "Temperature min" to a value higher than the current temperature. |

| Expected result: 

The value of resource "temperature" from service "alert" of provider "temperature alert" of API is "false".

No temperature alert appears on the nurse tablet for the corresponding studio. |

- Attachments: None
- Requirement: None

**STEP 3 :**

| Action: On parameters interface, set parameter "Status" for the function to "active". |

| Expected result: 

The value of resource "temperature" from service "alert" of provider "temperature alert" of API is still "true".

Temperature alert appears on the nurse tablet for the corresponding studio. |

- Attachments: None
- Requirement: None

**STEP 4 :**

| Action: On nurse tablet, click on the temperature alert and acknowledge it as an approved alert. |

| Expected result: 

The value of resource "acquit_alert" from service "control" of provider "temperature alert" of API is "true".

The value of resource "temperature" from service "alert" of provider "temperature alert" of API is "false".

On nurse tablet, temperature alert becomes inactive or disappears. |
STEP 5:

Action

On parameters interface, set parameter "Temperature min" to 16°C.

Expected result:

The value of resource "temperature" from service "alert" of provider "temperature alert" of API is still "true".

No new temperature alert appears on the nurse tablet for the corresponding studio.
This section briefly describes the tests done on the Equimetrix sensor, with respect to integration and usability. They have been performed in Madrid on the Deployment Site 3. Other tests have been done on the universAAL platform. The results of the IoT Platform tests are reported in Deliverable 5.3 of the ReAAL project, devoted to validate universAAL IoT. The technical tests (such as unit tests) summary can be found in section 4.2.1. [22]

The testing for this sensor has been ongoing for over a year. There are basically 3 types of testing performed:

1. Integration testing: testing whether the communication with the universAAL local instance is correct, the data received is correct, and possible errors and security problems in this communication.
2. Functional testing: testing the equimetrix application for normal operations, with stress testing running for extended periods of time, how it deals with disconnection of the device, and non-uniform UI interaction (user not following the predefined UI flow), among others.
3. Real testing: testing with real users, usability is specifically targeted in this category, looking for the effectiveness of the overall system to interact with the user, and report possible problems in the data gathering process.

For all these tests there is no explicit testing procedure defined (as defined in D9.1; which defines the more abstract DS experimental and deployment plan), but it is a continuous testing process where known previous errors are checked, and possible new bugs are discovered. Current testing status is OK for the majority of the issues detected; even though some errors are hard to replicate and analyze and for which no reliable solution has yet been provided. Despite this, the sensor has entered the production phase, and testing will continue to be performed on site, as the performed tests guarantee interoperability between the application and the universAAL platform, which in turn facilitates the integration of other services with this data.

C.1 Equimetrix device

Equimetrix core technology provides a mean to measure the relative 3D positions of different body parts: the Center of Mass and the Base of Support (or CoP). CoM position is tracked using one wearable vision based module (usually worn on the trunk). Information related to the BoS is provided by thin sensorised matress (named Flexipad) positioned below the user’s feet (Figure 69).
In Activage, and when in use, the Equimetrix software does not request any user-specific information. The Equimetrix software is connected to the IoT platform universAAL (later AIOTES) for the DS Madrid (later used in a cross pilot with DS ISERE) and data measured by the Equimetrix device is sent to universAAL (Figure 72) which is taking care of privacy and security aspect. See section C.4 for the data transmitted to universAAL.

C.2 Balance Assessment

Balance assessment is performed by asking the user to perform a serie of 12 pre-defined activities (standing on two feet, one foot, tandem, trunk/head motion ,etc). During each task, the system records and stores biomechnical data (see section C.4)
C.3 Balance Training

The training game was designed to train balance anticipation skills. In concreto the user has to collect a set of virtual gems by translating its Center of Mass. During the game the stability Index is estimated and stored (see section C.4).
C.4 Equimetrix and AIOTES

As it is shown on Figure 72, The Equimetrix device (green frame) is connected to an Activage center (or dedicated computer) which sends data to the universAAL server and database.
C.5 Equimetrix data

The data generated by Equimetrix are of two kinds:

- Detailed data files containing low level data about Equimetrix measurements such as Base of Support position, Center of Mass position of the user, Stability Index (see Table 7 AND Figure 73).
- Data sent to the universAAL server. Currently only summarised data (Stability Index: minimum, average, and maximum values) are sent to universAAL and accessible via through the user dashboard.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Storage information</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Nothing. Data files have an identification number provided by universal</td>
</tr>
<tr>
<td>Assessment data</td>
<td>The assessment is made of 12 tasks. Each time a task is performed a file is generated.</td>
</tr>
</tbody>
</table>
Training data

The training has 3 levels and a tutorial level (level 0). Each time a level is performed a file is generated.

Content of Balance and training data file

Every file generated have the same information inside. It has a:

- a timestamp
- data required for stability index computation (area of base of support, relative position of CoM with respect to the BoS boundary), data
- stability index value
- data related to the camera 3D position
- data related to the Flexipad (points describing the BoS, position of our estimated CoP)

An example of a log file is provided on Figure 73.
C.6 Technical verification

C.6.1 Methodology

Development verification

Continuous integration

Developers practicing continuous integration merge their changes back to the main branch as often as possible. The developer’s changes are validated by creating a build and running automated tests against the build.
Deployment verification

Continuous delivery of Equimetrix_activage installation software.

We also have automated our release process and we can generate our software installer at any point of time by launching a script. This installer includes The Equimetrix application developed by Tecnalia but also universAAL communication layer.

Functional tests of Equimetrix. The Equimetrix software installer is used to deploy the application on a fresh new windows 10 machine where functional tests are performed (balance assessment, balance training without connection to universAAL).

Functional tests of the communication with UniversAAL.

In these test, we test the communication between Equimetrix and universAAL. During balance assessment the user is performing a serie of functional tasks (standing on two legs, standing on one leg, semitandem, ...). For each of these tasks, balance performance scores are sent to universAAL together with a log files (see Table 7. E.g. assessment1.step1). The test consists on checking that these data arrived properly.

C.6.2 Continuous testing

Tests are run continuously and from different machines (from Tecnalia, from UPM, from TEA).

Figure 69 shows universAAL repository with Equimetrix data. Each folder represents a machine sending Equimetrix data.

<table>
<thead>
<tr>
<th>Nombre</th>
<th>Tamaño</th>
<th>Modificado</th>
</tr>
</thead>
<tbody>
<tr>
<td>0e107ac1-a4b9-4abd-8138-327c4508d353</td>
<td>03/08/2018 16:32:12</td>
<td></td>
</tr>
<tr>
<td>8be8fc99-54df-4ab3-b7c5-90d714e9652c</td>
<td>14/09/2018 12:08:28</td>
<td></td>
</tr>
<tr>
<td>02c115f-6d33-4c3b-ae3c-726a0bb548dd5</td>
<td>12/09/2018 17:57:10</td>
<td></td>
</tr>
<tr>
<td>2314dd5b-5c6-43e6-26b5-6a480c1be04</td>
<td>13/09/2018 22:50:48</td>
<td></td>
</tr>
<tr>
<td>89130a-9f3-437e-4a3c-30d2d914f916</td>
<td>13/09/2018 15:20:40</td>
<td></td>
</tr>
<tr>
<td>01879c0-666b-469a-b951-d019368905c</td>
<td>31/10/2018 16:33:55</td>
<td></td>
</tr>
<tr>
<td>1399186-2e48-4d3e-9f43-4a91fb3311ef</td>
<td>14/09/2018 12:22:33</td>
<td></td>
</tr>
<tr>
<td>a665f4f-f7aa-4e4b-8c30-61e7248f25a</td>
<td>14/09/2018 12:08:29</td>
<td></td>
</tr>
<tr>
<td>bf4e4e5f-62aa-4b-4a9f-c6953d37c2</td>
<td>12/09/2018 17:57:18</td>
<td></td>
</tr>
<tr>
<td>dba38ee-a4c8-4189-9f60-9738942db51</td>
<td>13/09/2018 11:12:15</td>
<td></td>
</tr>
<tr>
<td>ea5c577-795a-481c-88b9-73f3d12a0ac07</td>
<td>28/09/2018 13:00:38</td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>31/10/2018 11:20:54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12/09/2018 16:22:32</td>
<td></td>
</tr>
</tbody>
</table>

Figure 74. Example of universAAL repository with Equimetrix data.

For each machine the data of each user is stored. Figure 75 shows the users who have used the machine d8aa38ee.
Figure 75. Users of a specific machine (ID: d8aa38ee)
Appendix D  Developments tools

This Annex gives the details about the verification tests done on Ontology Explorer, the Device Semantics Editor the Data analyser, the Data Model Workbench and the Metadata Storage Explorer.

D.1 ACTIVAGE Ontology Explorer

The ACTIVAGE Ontology Explorer is a web application used for visualizing the ACTIVAGE Ontologies. The application is verified through scenario testing, where realistic user activities that cover the most important components of the system are used as scenario tests. The following tables demonstrate these tests, along with the needed actions per test.

Table 8: Ontology explorer tool tests

<table>
<thead>
<tr>
<th>Test Case Description</th>
<th>Navigate using tabs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S.No</strong></td>
<td><strong>Step description</strong></td>
</tr>
<tr>
<td>1</td>
<td>Go to the Annotation properties</td>
</tr>
<tr>
<td>2</td>
<td>Go to the Object properties</td>
</tr>
<tr>
<td>3</td>
<td>Go to the Data properties</td>
</tr>
<tr>
<td>4</td>
<td>Go to the class instances</td>
</tr>
<tr>
<td>5</td>
<td>Go to the classes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case Description</th>
<th>Select a class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S.No</strong></td>
<td><strong>Step description</strong></td>
</tr>
<tr>
<td>1</td>
<td>Go to the classes</td>
</tr>
<tr>
<td>2</td>
<td>Select a class</td>
</tr>
</tbody>
</table>
### Test Case Description: Search for a class

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the classes</td>
<td>Click on the 'Classes' tab</td>
<td>Class panel is displayed. On the left 169 different classes are displayed in tree-like structure</td>
</tr>
<tr>
<td>2</td>
<td>Search for a class by using its name</td>
<td>Type the name of a class and click on ‘Seach’ e.g. JSON</td>
<td>The graph focuses on the class and the corresponding node is painted red. On the right all details related to the class are displayed</td>
</tr>
</tbody>
</table>

### Test Case Description: Select an annotation property

<table>
<thead>
<tr>
<th>S.No</th>
<th>Action</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the Annotation properties</td>
<td>Click on the 'Annotation Properties' tab</td>
<td>Annotation Properties panel is displayed. On the left 28 different property names are displayed</td>
</tr>
<tr>
<td>2</td>
<td>Select an annotation property</td>
<td>Click on one of the listed annotation properties e.g. hasSympol</td>
<td>The name of the selected property and its URI are displayed on the right</td>
</tr>
</tbody>
</table>
### Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action.RELATED (e.g., <code>Object Properties</code> tab)</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the Object properties</td>
<td>Click on the ‘Object Properties’ tab</td>
<td>Object Properties panel is displayed. On the left 56 different property names are displayed</td>
</tr>
<tr>
<td>2</td>
<td>Select an object property</td>
<td>Click on one of the listed properties e.g. actsOnProperty</td>
<td>The name of the selected property and its uri are displayed on the right. In the ‘Annotation’ section, all the annotation properties related to the selected property are displayed (if any). The ‘Description’ section demonstrates important characteristics of the property such as its ‘Domain’, ‘Range’ and ‘Inverse of’</td>
</tr>
</tbody>
</table>

### Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action.RELATED (e.g., <code>Data Properties</code> tab)</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the Data properties</td>
<td>Click on the ‘Data Properties’ tab</td>
<td>Data Properties panel is displayed. On the left 11 different property names are displayed</td>
</tr>
<tr>
<td>2</td>
<td>Select a data property</td>
<td>Click on one of the listed properties e.g. asWKT</td>
<td>The name of the selected property and its uri are displayed on the right. In the ‘Annotation’ section, all the annotation properties related to the selected property are displayed (if any). The ‘Description’ section demonstrates important characteristics of the property such as its ‘Domain’, ‘Range’ and ‘Inverse of’</td>
</tr>
</tbody>
</table>
## Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the class instances</td>
<td>Click on the ‘Individuals by class’ tab</td>
<td>‘Individuals by class’ panel is displayed. On the left all the class names are displayed in a tree-like structure</td>
</tr>
<tr>
<td>2</td>
<td>Select a class</td>
<td>Click on one of the listed classes e.g. Measurement -&gt; Unit -&gt; ComplexUnit -&gt; UnitDerivedByShifting</td>
<td>All the instances of the selected classes are displayed in the central panel</td>
</tr>
<tr>
<td>3</td>
<td>Select an instance</td>
<td>Click on one of the listed class individuals e.g. degree</td>
<td>Details related to the selected instance are shown on the right</td>
</tr>
</tbody>
</table>

## Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the class instances</td>
<td>Click on the ‘Classes’ tab</td>
<td>‘Class panel is displayed.</td>
</tr>
<tr>
<td>2</td>
<td>Panning through the graph</td>
<td>Press the left mouse button and drag the mouse pointer over the graph</td>
<td>You can move through the different nodes of the graph</td>
</tr>
<tr>
<td>3</td>
<td>Zoom in/out</td>
<td>While the mouse pointer is over the graph, scroll up for zoom in or scroll down for zoom out</td>
<td>Zoom in or Zoom out</td>
</tr>
<tr>
<td>4</td>
<td>Select a node</td>
<td>Click on one of the nodes of the graph</td>
<td>The selected node is painted red. The tree-like structure on the left updates and the corresponding class is selected. The panel on the right is also updates and shows details related to the selected class</td>
</tr>
</tbody>
</table>
## D.2 Device Semantics Editor

The Device Semantics Editor is a web application used for defining/editing the semantics related to the devices. For its verification, the following scenario tests have been implemented.

### Table 9: Semantics editor tool tests

<table>
<thead>
<tr>
<th>Test Case Description</th>
<th>Create a new device sub-class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S.No</strong></td>
<td><strong>Step description</strong></td>
</tr>
<tr>
<td>1</td>
<td>Select the parent class</td>
</tr>
<tr>
<td>2</td>
<td>Select to create a new class</td>
</tr>
<tr>
<td>3</td>
<td>Provide class details</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case Description</th>
<th>Delete a device system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S.No</strong></td>
<td><strong>Step description</strong></td>
</tr>
<tr>
<td>1</td>
<td>Create a new device sub-class</td>
</tr>
<tr>
<td>2</td>
<td>Select the class to be deleted</td>
</tr>
<tr>
<td>3</td>
<td>Delete class</td>
</tr>
</tbody>
</table>
## Test Case Description
Delete a device system

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a new device sub-class</td>
<td>Follow the instructions of the corresponding test case</td>
<td>A new device class is created</td>
</tr>
<tr>
<td>2</td>
<td>Select the class to be deleted</td>
<td>On the left, click on the class that was created in the previous step.</td>
<td>A ‘Delete device system’ button appears at the bottom of the screen</td>
</tr>
<tr>
<td>3</td>
<td>Delete class</td>
<td>Click on the ‘Delete device system’ button</td>
<td>The selected class is removed from the menu on the left. Its parent class is selected now</td>
</tr>
</tbody>
</table>

## Test Case Description
Add/edit/delete annotation property

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a new device sub-class</td>
<td>Follow the instructions of the corresponding test case</td>
<td>A new device class is created</td>
</tr>
<tr>
<td>2</td>
<td>Create a new annotation property for this class</td>
<td>Go the section ‘Annotations’ on the right and click on the ‘+’ button. Select from the menu an annotation property, provide its value and click on ‘Save’</td>
<td>A new row is added in the ‘Annotations’ section containing the property that was just created.</td>
</tr>
<tr>
<td>3</td>
<td>Edit annotation property</td>
<td>Go to the ‘Annotations’ section and click on the ‘edit’ button of the property to be edited. Provide a new value for the property and click on ‘Save’</td>
<td>The row of the selected annotation property gets updated with the new value that was just provided.</td>
</tr>
<tr>
<td>4</td>
<td>Delete annotation property</td>
<td>Go to the ‘Annotations’ section and click on the ‘x’ button of the property to be deleted.</td>
<td>The row of the selected annotation property is removed.</td>
</tr>
</tbody>
</table>
## Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a new device sub-class</td>
<td>Follow the instructions of the corresponding test case</td>
<td>A new device class is created</td>
</tr>
<tr>
<td>2</td>
<td>Create a new restriction</td>
<td>Go the section 'Restrictions' on the right and click on the ‘+’ button. Select from the menu the type of the property, the property, a restriction type and its value, and click on 'Save'</td>
<td>A new row is added in the 'Restrictions' section containing the restriction that was just created.</td>
</tr>
<tr>
<td>3</td>
<td>Edit a restriction</td>
<td>Go to the ‘Restrictions’ section and click on the ‘edit’ button of the restriction to be edited. Modify any of the given options and click on ‘Save’.</td>
<td>The row of the selected restriction gets updated according to the modifications made.</td>
</tr>
<tr>
<td>4</td>
<td>Delete a restriction</td>
<td>Go to the ‘Restrictions’ section and click on the ‘x’ button of the restriction to be deleted.</td>
<td>The row of the selected restriction is removed.</td>
</tr>
</tbody>
</table>

## Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a new device sub-class</td>
<td>Follow the instructions of the corresponding test case</td>
<td>A new device class is created</td>
</tr>
<tr>
<td>2</td>
<td>Create a new datatype property</td>
<td>Go the section 'Datatype properties' on the right and click on the ‘+’ button. Provide the property name, a comment and its range, and click on 'Save'</td>
<td>A new row is added in the 'Datatype properties' section containing the name of the property that was just created.</td>
</tr>
<tr>
<td>3</td>
<td>Edit a datatype property</td>
<td>Go to the 'Datatype properties' section and click on the 'edit' button of the property to be edited. Modify its comment or its range, and click on 'Save'.</td>
<td>The row of the selected property gets updated according to the modifications made.</td>
</tr>
<tr>
<td>4</td>
<td>Delete a datatype property</td>
<td>Go to the 'Datatype properties' section and click on the 'x' button of the property to be deleted.</td>
<td>The row of the selected property is removed.</td>
</tr>
</tbody>
</table>
### Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a new device sub-class</td>
<td>Follow the instructions of the corresponding test case</td>
<td>A new device class is created</td>
</tr>
<tr>
<td>2</td>
<td>Create a new object property</td>
<td>Go the section ‘Object properties’ on the right and click on the ‘+’ button. Provide the property name, a comment and its range, and click on ‘Save’</td>
<td>A new row is added in the ‘Object properties’ section containing the name of the property that was just created.</td>
</tr>
<tr>
<td>3</td>
<td>Edit an object property</td>
<td>Go to the ‘Object properties’ section and click on the ‘edit’ button of the property to be edited. Modify its comment or its range, and click on ‘Save’.</td>
<td>The row of the selected property gets updated according to the modifications made.</td>
</tr>
<tr>
<td>4</td>
<td>Delete an object property</td>
<td>Go to the ‘Object properties’ section and click on the ‘x’ button of the property to be deleted.</td>
<td>The row of the selected property is removed.</td>
</tr>
</tbody>
</table>

### D.3 Data Analyser

The Data analyser is a web application for experimenting with data analytics methods on a loaded dataset. The data analytics methods that are utilized by this tool are part of the Analytics and their verification is described in detail in section 3.5. For the application and its main functionalities, a series of different scenarios has been developed in order to verify them. These scenarios include several steps and for each step there is a detailed description of the action needed and of the expected result. The aforementioned scenarios are demonstrated in the following tables.

#### Table 10: Data analyzer tool tests

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Choose a file</td>
<td>Click on the ‘Choose File’ button and then select the file you want to load</td>
<td>The name of the chosen file is displayed right next to the button</td>
</tr>
<tr>
<td>2</td>
<td>Load the file</td>
<td>Click on the 'Upload' button</td>
<td>A table named ‘Raw data’ appears in the middle of the screen, containing the values of the loaded file</td>
</tr>
<tr>
<td>Test Case Description</td>
<td>Raw data table functionalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>S.No</strong></td>
<td><strong>Step description</strong></td>
<td><strong>Action</strong></td>
<td><strong>Expected result</strong></td>
</tr>
<tr>
<td>1</td>
<td>Load data</td>
<td>Follow the steps described in the corresponding scenario</td>
<td>A table named 'Raw data' appears in the middle of the screen, containing the values of the loaded file</td>
</tr>
<tr>
<td>2</td>
<td>Sort columns</td>
<td>Click on the name of each column</td>
<td>The values of the column are sorted</td>
</tr>
<tr>
<td>3</td>
<td>Change page</td>
<td>Click on the pagination buttons at the bottom of the table</td>
<td>A different page of data is loaded to the table</td>
</tr>
<tr>
<td>4</td>
<td>Change number of entries</td>
<td>Change the value of the select box at the top-left of the table</td>
<td>The number of rows shown per page is modified</td>
</tr>
<tr>
<td>5</td>
<td>Search for a value</td>
<td>Search for a specific value in the table by writing the value, or part of it, in the search box at the top right of the table</td>
<td>As soon as you begin typing, the table is filtered accordingly.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case Description</th>
<th>Run analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S.No</strong></td>
<td><strong>Step description</strong></td>
</tr>
<tr>
<td>1</td>
<td>Load data</td>
</tr>
<tr>
<td>2</td>
<td>Select analysis method</td>
</tr>
<tr>
<td>3</td>
<td>Configure analysis method and run it</td>
</tr>
</tbody>
</table>
# Test Case Description

<table>
<thead>
<tr>
<th>S.No</th>
<th>Step description</th>
<th>Action</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Run analysis</td>
<td>Follow the steps described in the corresponding scenario</td>
<td>A new table named ‘Analytics results’ appears right below the ‘Raw data’ table and displays the results of the analysis</td>
</tr>
<tr>
<td>2</td>
<td>Export results</td>
<td>At the right side of the screen, below the method parameters, there is the &quot;Export results&quot; area. Select one of currently supported formats and click on the ‘Export results’ button</td>
<td>The outcome of the analysis is downloaded to a file named &quot;results&quot;, with an extension dependent on the selected format.</td>
</tr>
</tbody>
</table>

## D.4 DataModel 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.

This section reports the tests that have been applied to the DataModel Workbench tool. For the DataModel Workbench tool, we have unit tests and integration tests, and we intend to continue testing as the development continues.

### D.4.1 Unit tests

Unit tests are run by Karma, Test Runner for JavaScript, and written with Jasmine, Behavior-Driven JavaScript. They’re located in [src/test/javascript/](src/test/javascript/) and can be run with:

- `gulp test`

Thus, testing has the following steps:

- Identify tasks that the software has to perform.
- Create input data based on the specifications.
- Determine the output based on the specifications.
D5.3 Intermediate Validation Results

- Execute the test
- Check the actual and expected outputs.

D.4.2 Integration Tests

UI end-to-end tests are powered by Protractor, framework for Angular applications, which is built on top of WebDriverJS. They're located in [src/test/javascript/e2e](src/test/javascript/e2e) and can be run by starting Spring Boot in one terminal with

- `./mvnw spring-boot:run`

and running the tests in a second one with

- `gulp test`

Database test cases

- should load databases
- should load create Database dialog
- show detail database/edit database
D5.3 Intermediate Validation Results

Figure 76: Database tests
Table test cases

- should load Tables
- should load create Table dialog
- show detail Table and edit Table
D5.3 Intermediate Validation Results

```javascript
'use strict';

describe('Table e2e test', function () {
    var username = element(by.id('username'));
    var password = element(by.id('password'));
    var entityMenu = element(by.id('entity-menu'));
    var accountMenu = element(by.id('account-menu'));
    var login = element(by.id('login'));
    var logout = element(by.id('logout'));

    beforeAll(function () {
        browser.get('');
        accountMenu.click();
        login.click();

        username.sendKeys('admin');
        password.sendKeys('admin');
        element(by.css('button[type=submit'])).click();
    });

    it('should load Tables', function () {
        entityMenu.click();
        element(by.id('ui-select-tables')).first().click().then(function () {
            expect(element(by.all(by.css('#l2s')).first()).getText().toMatch('Tables'));
        });
    });

    it('should load create Table status', function () {
        element(by.css('#[ui-id=table состоянии]').click).then(function () {
            expect(element(by.css('h4.modal-title')).getText()).toMatch('Create or edit a Table');
        });
    });

    afterAll(function () {
        accountMenu.click();
        logout.click();
    });
});

describe('Controller Tests', function () {

describe('Table Management Detail Controller', function () {
    var $scope, $rootScope;
    var MockEntity, MockPreviousState, MockTable;
    var createController;

    beforeEach(inject(function ($injector) {
        $rootScope = $injector.get('$rootScope');
        $scope = $rootScope.$new();
        MockEntity = jasmine.createSpy(MockEntity);
        MockPreviousState = jasmine.createSpy(MockPreviousState);
        MockTable = jasmine.createSpy(MockTable);

        var locals = {
            $scope: $scope,
            $rootScope: $rootScope,
            entity: MockEntity,
            previousState: MockPreviousState,
            'Table': MockTable
        };
        createController = function () { $injector.get('$controller')('TableDetailController', locals); });
    });

    describe('Root Scope Listening', function () {
        it('Unregisters root scope listener upon scope destruction', function () {
            var eventType = 'datatablesApp:table:update';
            createController();
            expect($rootScope.$listenerCount[eventType]).toEqual(1);
            $scope.$destroy();
            expect($rootScope.$listenerCount[eventType]).toBeUndefined();
        });
    });
});
```

Figure 77: Table tests
Schema Test Cases

- should load Schemas
- should load create Schema dialog
- show detail Schema/ edit Schema
Query Test Cases

- should load Queries
- should load create Query dialog
- show detail Query result

```javascript
'use strict';
describe('Query e2e test', function () {
  var username = element(by.id('username'));
  var password = element(by.id('password'));
  var entityMenu = element(by.id('entity-menu'));
  var accountMenu = element(by.id('account-menu'));
  var login = element(by.id('login'));
  var logout = element(by.id('logout'));

  beforeEach(function () {
    browser.get('/');
    accountMenu.click();
    login.click();
    username.sendKeys('admin');
    password.sendKeys('admin');
    element(by.css('button[type=submit]')).click();
  });

  it('should load queries', function () {
    entityMenu.click();
    element(by.css('#id-get-"query"')).first().click().then(function () {
      expect(element.all(by.css('td'))).first().getText().toMatch(/Queries/);
    });
  });

  it('should load create Query dialog', function () {
    element(by.css('#id-modal-create-query')).click().then(function () {
      expect(element(by.css('#id-modal-create-query')).getText()).toMatch(/create or edit a Query/);
      element(by.css('button[ng-click]')).click();
    });
  });

  afterAll(function () {
    accountMenu.click();
    logout.click();
  });
});
```

Figure 79: Query Test
D.4.3 Validation Result

The correct performance of this component is validated through a UI functional testing. Testing on which it is validated if the system fulfils the functional requirements/specifications. Functions are validated by feeding them with an input and examining the output.

![Database creation functional sample validation result.]

To this date all tests are passing, more tests will be added in case we will need to have more coverage to test updated and/or new features being developed.

D.5 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 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.

This section reports the tests that have been applied to the Metadata Storage Explorer tool. For the Metadata Storage Explorer tool, we have unit tests and integration tests, and we intend to continue testing as the development continues.
D.5.1 Unit Tests

Unit tests are run by Karma, Test Runner for JavaScript, and written with Jasmine, Behavior-Driven JavaScript. They’re located in [src/test/javascript/](src/test/javascript/) and can be run with:

- `gulp test`

Thus, testing has the following steps:

- Identify tasks that the software has to perform.
- Create input data based on the specifications.
- Determine the output based on the specifications.
- Execute the test
- Check the actual and expected outputs.

D.5.2 Integration Tests

UI end-to-end tests are powered by Protractor, framework for Angular applications, which is built on top of WebDriverJS. They’re located in [src/test/javascript/e2e](src/test/javascript/e2e) and can be run by starting Spring Boot in one terminal with

- `./mvnw spring-boot:run`

and running the tests in a second one with

- `gulp test`

Model test cases

- should load Models
- should load create Model dialog
- should create Model
- should delete Model
- should update Model
- show detail model
D5.3 Intermediate Validation Results

```
'use strict';

describe('Model e2e test', function () {
  var username = element(by.id('username!'));
  var password = element(by.id('password!'));
  var entityMenu = element(by.id('entity-menu!'));
  var accountMenu = element(by.id('account-menu!'));
  var login = element(by.id('login!'));
  var logout = element(by.id('logout!'));

  beforeEach(function () {
    browser.get('/');
    accountMenu.click();
    login.click();
    username.sendKeys('admin!');
    password.sendKeys('admin!');
    entityMenu.click();
  });

  it('should load Models', function () {
    entityMenu.click();
    element.all(by.css('li.ui-autocomplete'))
      .first().click().then(function () {
      expect(element.all(by.css('button')).first().getText()).toBeMatch('Models!');
    });
  });

  it('should load create model status', function () {
    element(by.css('li.ui-autocomplete')).click().then(function () {
      expect(element(by.css('button')).getText()).toBeMatch('Create or edit a Model!');
    });
  });

  afterAll(function () {
    accountMenu.click();
    logout.click();
  });
});
```

Figure 81: Model test
D.5.3 Validation Result

The correct performance of this component is validated through a UI functional testing.

Testing on which it is validated if the system fulfills the functional requirements/specifications. Functions are validated by feeding them with an input and examining the output.

Figure 82: Model creation functional sample validation result.

To this date all tests are passing, more tests will be added in case we will need to have more coverage to test updated and/or new features being developed.
Appendix E  Data Lake

E.1  Independent raw data storage

The Raw Data Integration component consists on a non-relational database (InfluxDB) and an alternative REST API developed in the ACTIVAGE project. This alternative API will be used in the ACTIVAGE ecosystem instead of the own REST API of InfluxDB databases, and supports 6 operations (createDB, deleteDB select, insert, update, delete). It represents a wrapper API, which means that it would be possible to change the database wrapper with minimal changes from the developer’s side, and no changes from the API user's side.

The data collected from the different IoT platforms are stored in a distributed manner: the data of each DS platform are stored into its own database -associated to the IoT platform. The ACTIVAGE independent data storage is offered to DS IoT platforms to have the option of employing this independent database, in case those platforms do not have data storage services or prefer to use an independent one which is accessible online. Then, IoT platform data can be stored in AIoTES online database. Further information about the ACTIVAGE Independent Data Storage services can be found in D4.5.

The correct performance of this component is tested through a functional testing.

Functional testing is a type of software testing on which it is tested if the system fulfills the functional requirements/specifications. Functions are tested by feeding them with an input and examining the output. This type of testing is not related with how processing occurs, but rather, with the results of processing. It simulates actual system use.

Thus, functional testing has the following steps:

- Identify functions that the software has to perform.
- Create input data based on the function’s specifications.
- Determine the output based on the function’s specifications.
- Execute the test
- Compare the actual and expected outputs.

All 5 operations in the ACTIVAGE InfluxDB REST API are tested, as they are identified as the functions that the REST API must perform:

- /createDB : creates a new database
- /deleteDB: deletes an existing database
- /select : select the set of registries that match the conditions specified in the instruction
- /insert : inserts a registry
- /update : updates a current registry
- /delete : deletes a current registry

On the following table a set of sample inputs and expected outputs can be seen, covering all operations and considering the content already stored in the tested instance of the database.
### Table 11: API Functional Test Inputs and expected Outputs

<table>
<thead>
<tr>
<th>Query (Input)</th>
<th>Expected Result (Output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST&lt;br&gt;<a href="http://localhost:4567/independentStorage/createDB">http://localhost:4567/independentStorage/createDB</a></td>
<td>{&lt;br&gt;  &quot;message&quot;: &quot;The database test was created successfully.&quot;}&lt;br&gt;}</td>
</tr>
<tr>
<td>POST&lt;br&gt;<a href="http://localhost:4567/independentStorage/deleteDB">http://localhost:4567/independentStorage/deleteDB</a></td>
<td>{&lt;br&gt;  &quot;message&quot;: &quot;The database test was deleted successfully.&quot;}&lt;br&gt;}</td>
</tr>
</tbody>
</table>
| POST<br>http://localhost:4567/independentStorage/insert<br>BODY<br>{<br>    "db": "test",
    "table": "meas",
    "data": {<br>      "platformId": "FIWARE",
      "device": "termometro",
      "observation": {<br>        "@graph": [{<br>          "@graph": [{<br>            "@id": "InterIoTMsg:meta308c3987-b69e-4672-890b-3f3d6229596d",
            "@type": [<br>              "InterIoTMsg:meta",
              "InterIoTMsg:Observation"
            ],
            "InterIoTMsg:conversationID": "conv85"
          }]<br>        }<br>      }
    },
| "message": "Data inserted successfully."<br>} |
"InterIoTMsg:dateTimeStamp": "2017-05-08T13:48:19.428+02:00",

"InterIoTMsg:messageID":"msg204d0405-a6da-4054-a6db-96d20c413746"
}]

"@id": "InterIoTMsg:metadata"
},
{
"@graph": [
{
"@id": "_:b0",
"@type": "http://www.w3.org/2006/time#TimePosition",

"http://www.w3.org/2006/time#numericPosition": {
"@type": "xsd:decimal",
"@value": "1418143893015"
}
},
{
"@id": "_:b3",
"@type": "owl:Restriction",
"owl:allValuesFrom": {
"@id": "http://ontology.universAAL.org/Perso
nalHealthDevice.owl#HeartRateSensor"
}
},
}
D5.3 Intermediate Validation Results

```
"owl:onProperty": {"@id": "rdf:subject"}
}
{
  "@id": "ns1:hr",
  "@type": [

  "http://www.w3.org/ns/sosa/Result",
    "ns3:HeartRate"
  ],

  "InterIoT:GOIoTP#hasValue": {
      "@type": "xsd:int",
      "@value": "33"
  }
},
{
  "@id": "urn:org.universAAL.middleware.context.rdf:ContextEvent#:_9e2aa729ac420ba3:182a",
      "@type": "http://www.w3.org/ns/sosa/Observation",
      "http://www.w3.org/ns/sosa/hasResult": {
                     "@id": "ns1:hr"},

      "http://www.w3.org/ns/sosa/madeBySensor": {
                      "@id": "ns1:hrs"},

      "http://www.w3.org/ns/sosa/phenomenonTime": {
                      "@id": "_b1"}
  }
},
"@id": "InterIoTMsg:payload"
```
D5.3 Intermediate Validation Results

```
    },
    "@context": {
      "ns": "http://ontology.universaal.org/PhThin g.owl#",
      "owl": "http://www.w3.org/2002/07/owl#",
      "InterIoTMsg": "http://inter-iot.eu/message/",
      "InterIoTInst": "http://inter-iot.eu/inst/",
      "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
      "xsd": "http://www.w3.org/2001/XMLSchema#",
      "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
      "InterIoT": "http://inter-iot.eu/",
      "ns2": "http://ontology.universaal.org/Measure ment.owl#",
      "ns1": "http://ontology.universAAL.org/Simpl eHealthclient.owl#",
      "ns4": "http://ontology.universAAL.org/Devic e.owl#",
      "ns3": "http://ontology.universaal.org/Health Measurement.owl#"
    }
  }
}
```

POST http://localhost:4567/independentStorage/select
"platformId": "FIWARE",
"device": "termometro",
"observation": 
{""@graph": [{""@type": ["InterIoTMsg:meta", "InterIoTMsg:Observation"], "InterIoTMsg:conversationID": "conv85e0f5d2-cf65-4d23-84b1-ff1381ae01fc"}, "@id": "InterIoTMsg:meta308c3987-b69e-4672-890b-3f3d6229596d"}, "InterIoTMsg:dateTimeStamp": "2017-05-08T13:48:19.428+02:00", "InterIoTMsg:messageID": "msg204d0405-a6da-4054-a6db-96d20c413746"}], "@id": "InterIoTMsg:paylod"}
"@context": {"ns": "http://ontology.universAAL.org/PhThing.owl#", "owl": "http://www.w3.org/2002/07/owl#", "InterIoTMsg": "http://interiot.eu/message/", "InterIoTInst": "http://interiot.eu/inst/", "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-
POST http://localhost:4567/independentStorage/update

BODY

{
  "db": "prueba",
  "table": "meas",
  "query": "SELECT * FROM meas",
  "data": {
    "platformId": "FIWARE",
    "device": "termometro",
    "observation": {
      "@graph": [{
        "@graph": [{
          "@id": "InterIoTMsg:meta308c3987-b69e-4672-890b-3f3d6229596d",
          "@type": [
            "InterIoTMsg:meta",
            "InterIoTMsg:Observation"
          ],
        }
      ]
    }
  }
}
"InterIoTMsg:conversationID": "conv85e0f5d2-cf65-4d23-84b1-ff1381ae01fc",

"InterIoTMsg:dateTimeStamp": "2017-05-08T13:48:19.428+02:00",

"InterIoTMsg:messageID": "msg204d0405-a6da-4054-a6db-96d20c413746"
}
],
"@id": "InterIoTMsg:metadata"
},
{
"@graph": [
{
"@id": "_:b0",
"@type": "http://www.w3.org/2006/time#TimePosition",
"http://www.w3.org/2006/time#numericPosition": {
"@type": "xsd:decimal",
"@value": "1418143893030"
}
},
{
"@id": "_:b3",
"@type": "owl:Restriction",
"owl:allValuesFrom": {
"@id": "http://ontology.universAAL.org/Perso
nalHealthDevice.owl#HeartRateSensor",
"@type": "owl:Restriction",
"owl:allValuesFrom": {
"@id": "http://ontology.universAAL.org/Perso
nalHealthDevice.owl#HeartRateSensor"
}
© 2019 ACTIVAGE

D5.3 Intermediate Validation Results

```
},
  "owl:onProperty": {"@id": "rdf:subject"}
},
{
  "@id": "ns1:hr",
  "@type": ["http://www.w3.org/ns/sosa/Result",
            "ns3:HeartRate"
          ],
  "InterIoT:GOIoTP#hasValue": {
    "@type": "xsd:int",
    "@value": "33"
  }
},
{
  "@id": "urn:org.universAAL.middleware.context.rdf:ContextEvent#:9e2aa729ac420ba3:182a",
  "@type": "http://www.w3.org/ns/sosa/Observation",
  "http://www.w3.org/ns/sosa/hasResult": {"@id": "ns1:hr"},
  "http://www.w3.org/ns/sosa/madeBySensor": {"@id": "ns1:hrs"},
  "http://www.w3.org/ns/sosa/phenomenonTime": {"@id": "_:b1"}
}
},
"@id": "InterIoTMsg:payload"
```
D5.3 Intermediate Validation Results

```json
{
   "@context": {
      "ns": "http://ontology.universaal.org/PhThin
g.owl#",
      "owl": "http://www.w3.org/2002/07/owl#",
      "InterIoTMsg": "http://inter-
io.t.eu/message/",
      "InterIoTInst": "http://inter-
io.t.eu/inst/",
      "rdf": "http://www.w3.org/1999/02/22-rdf-
syntax-ns#",
      "xsd": "http://www.w3.org/2001/XMLSchema #",
      "rdfs": "http://www.w3.org/2000/01/rdf-
schema#",
      "InterIoT": "http://inter-iot.eu/",
      "ns2": "http://ontology.universaal.org/Measur
ement.owl#",
      "ns1": "http://ontology.universAAL.org/Simpl
eHealthclient.owl#",
      "ns4": "http://ontology.universAAL.org/Devic
e.owl#",
      "ns3": "http://ontology.universaal.org/Health
Measurement.owl#"
   }
}
```
POST
http://localhost:4567/independentStorage/delete
BODY
{
    "db": "test",
    "table": "meas",
    "query": "DELETE FROM meas"
}

Table 12: Independent data storage API general information

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base URL</td>
<td>/independentStorage</td>
</tr>
<tr>
<td>Method</td>
<td>POST</td>
</tr>
<tr>
<td>Headers</td>
<td>Content-Type: application/json</td>
</tr>
</tbody>
</table>
| Sub-services | /createDB  
|             | /deleteDB  
|             | /select  
|             | /insert  
|             | /update  
|             | /delete |

The following images are complemented with the API operation description. These images show the functional testing performed to each of these operations. Each image represents a REST API interaction using an Http Client program; on each interaction, an input is set through the Raw Independent Storage REST API, and an output is received from the API. Finally, the results of the use of this inputs can be seen below. Those results confirm that is achieved the expected performance from a functional point of view with the sample inputs and outputs. Other inputs and outputs were tested, but for the sake of simplicity and to avoid redundant testing reports, only one set is shown in this section, covering the whole set of operations.

Test CreateDB operation

Table 13: CreateDB operation

/createDB

Creates a new database.
D5.3 Intermediate Validation Results

Input JSON schema

```
{  
  "type": "object",  
  "properties": {  
    "db": {  
      "type": "string",  
      "description": "A unique identifier for the DB."  
    }  
  }  
}
```

Input JSON example

```
{  
  "db": "myPlatformData"  
}
```

Output JSON schema

```
{  
  "type": "object",  
  "properties": {  
    "message": {  
      "type": "string",  
      "description": "A success or error message."  
    }  
  }  
}
```

Output JSON example

```
{  
  "message": "The database was created successfully."  
}
```

Figure 83: CreateDB functional sample test

Test DeleteDB operation
### Table 14: DeleteDB operation

Delete a database.

<table>
<thead>
<tr>
<th>Input</th>
<th>JSON schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;query&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;description&quot;: &quot;DB to delete.&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;db&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;description&quot;: &quot;The unique identifier of the DB.&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

| JSON example | { |
|              |   "db": "myPlatformData" |
|              | } |

<table>
<thead>
<tr>
<th>Output</th>
<th>JSON schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;message&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;description&quot;: &quot;A success or error message.&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

| JSON example | { |
|              |   "message": "Success" |
|              | } |
**Test Select operation**

Table 15: Select operation

/\select

Selects data from a database.

<table>
<thead>
<tr>
<th>Input</th>
<th>JSON schema</th>
</tr>
</thead>
</table>
|       | {
|       |   "type": "object",
|       |   "properties": {
|       |     "db": {
|       |       "type": "string",
|       |       "description": "The unique identifier of the DB."
|       |     },
|       |     "table": {
|       |       "type": "string",
|       |       "description": "The name of the table."
|       |     },
|       |     "query": {
|       |       "type": "string",
|       |       "description": "A query to select data from the database"
|       |     }
|       | } |

<table>
<thead>
<tr>
<th>JSON example</th>
</tr>
</thead>
</table>
| {
|   "db": "myPlatformData",
|   "table": "table1",
| } |
D5.3 Intermediate Validation Results

JSON schema

```
{
  "type": "object",
  "properties": {
    "data": {
      "type": "object",
      "description": "The data obtained from the query."
    }
  }
}
```

JSON example

```
{
  "data": [
    {
      "@graph": [
        {
          "@id": "InterIoTMsg:meta308c3987-b69e-4672-890b-3f3d6229596d",
          "@type": ["InterIoTMsg:meta", "InterIoTMsg:Observation"],
          "InterIoTMsg:conversationID": "conv85e0f5d2-cf65-4d23-84b1-ff1381ae01fc",
          "InterIoTMsg:dateTimeStamp": "2017-05-08T13:48:19.428+02:00",
          "InterIoTMsg:messageID": "msg204d0405-a6da-4054-a6db-96d20c413746"
        },
        {
          "$id": "InterIoTMsg:metadata"
        }
      ],
      "$id": "InterIoTMsg:metadata"
    }
  ]
}
```
D5.3 Intermediate Validation Results

Figure 85: Select functional sample test
### Test Insert operation

**Table 16: Insert operation**

<table>
<thead>
<tr>
<th>Input</th>
<th>JSON schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>/insert</td>
<td>Inserts data into a database.</td>
</tr>
</tbody>
</table>

Inserts data into a database.

**JSON example**

```json
{
  "db": "myPlatformData",
  "table": "table1",
  "data": {
    "@graph": [{
      "@id": "InterIoTMsg:meta308c3987-b69e-4672-890b-3f3d6229596d",
      "@type": ["InterIoTMsg:meta", "InterIoTMsg:Observation"
        ],
      "InterIoTMsg:conversationID": "conv85e8f5d2-cf65-4d23-84b1-ff1381ae01fc",
      "InterIoTMsg:dateTimeStamp": "2017-05-08T13:48:19.428+02:00",
      "InterIoTMsg:messageID": "msg204d0405-a6da-4054-a6db-96d20c413746"
    }
      ],
    "@id": "InterIoTMsg:metadata"
  }
}
```

```json
{
  "@graph": [{
    "@id": "_:b0",
    "@type": "http://www.w3.org/2006/time#TimePosition",
    "http://www.w3.org/2006/time#numericPosition": {
      "@type": "xsd:decimal",
      "@value": "1418143893015"
    }
  },
  {
    "@id": "_:b3",
    "@type": "http://www.w3.org/2006/time#TimePosition",
    "http://www.w3.org/2006/time#numericPosition": {
      "@type": "xsd:decimal",
      "@value": "1418143893015"
    }
  }
}
### Output

**JSON schema**

```json
{
  "type": "object",
  "properties": {
    "message": {
      "type": "string",
      "description": "A success or error message."
    }
  }
}
```

**JSON example**

```json
{
  "message": "Data inserted successfully."
}
```
Test Update operation

Table 17: Update operation

Updates the data contained in a database.

```
{  
  "type": "object",  
  "properties": {  
    "db": {  
      "type": "string",  
      "description": "The unique identifier of the DB."  
    },  
    "table": {  
      "type": "string",  
      "description": "The name of the table."  
    },  
    "query": {  
      "type": "string",  
      "description": "Condition to update element."  
    },  
    "data": {  
      "type": "string",  
      "description": "The data that will be inserted"  
    }  
  }  
}
```
```json
{
  "db": "myPlatformData",
  "table": "table1",
  "data": {
    "@graph": [
      {
        "@id": "_:b0",
        "@type": "http://www.w3.org/2006/time#TimePosition",
        "http://www.w3.org/2006/time#numericPosition": {
          "@type": "xsd:decimal",
          "@value": "1418143893015"
        }
      },
      {
        "@id": "_:b3",
        "@type": "owl:Restriction",
        "owl:allValuesFrom": {
          "@id": "http://ontology.universAAL.org/PersonalHealthDevice.owl#HeartRateSensor"
        },
        "owl:onProperty": {
          "@id": "rdf:subject"
        }
      },
      {
        "@id": "ns1:hr",
        "@type": [
          "http://www.w3.org/ns/sosa/Result",
          "ns3:HeartRate"
        ],
        "InterIoT:GOIoTP#hasValue": {
          "@type": "xsd:int",
          "@value": "33"
        }
      },
      {
        "@id": "ns1:pub2",
        "@type": [
          "InterIoT:medex#Gauge",
          "http://www.w3.org/ns/sosa/Platform"
        ],
        "http://ontology.universAAL.org/Context.owl#hasType": {
          "@id": "http://ontology.universAAL.org/Context.owl#gauge"
        },
        "http://ontology.universAAL.org/Context.owl#myClassesOfEvents": {
          "@id": "_:b2"
        }
      }
    ],
    "@id": "InterIoTMsg:payload"
  }
}
```

Version 1.0  |  2019-02-14  |  ACTIVAGE ©
Output

JSON schema

```json
{
  "type": "object",
  "properties": {
    "message": {
      "type": "string",
      "description": "A success or error message."
    }
  }
}
```

JSON example

```json
{
  "message": "Success"
}
```

Figure 87: Update functional sample test

Test Delete operation
Table 18: Delete operation

<table>
<thead>
<tr>
<th>/delete</th>
<th>Deletes data from a database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>JSON schema</td>
</tr>
<tr>
<td></td>
<td>JSON example</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
</tbody>
</table>
|          |   "type": "object",
|          |   "properties": { |
|          |     "query": { |
|          |       "type": "string",
|          |       "description": "Condition to delete element."
|          |     }
|          |   }
|          | } |

|          | JSON example |
|          | { |
|          |   "db": "myPlatformData",
|          |   "table": "table1",
|          |   "query": "DELETE WHERE { GRAPH <http://devices> { ?d mdw:id 11 ; ?property1 ?value1 } } "
|          | } |

| **Output** | JSON schema |
|            | JSON example |
|            | { |
|            |   "type": "object",
|            |   "properties": { |
|            |     "message": { |
|            |       "type": "string",
|            |       "description": "A success or error message."
|            |     }
|            | } |
|            | { |
|            |   "message": "Success"
|            | } |
E.2 Analytics metadata storage

The analytics metadata storage is a database used to store models created and used by the AIOTES data analytics methods (see Deliverable D5.4 Data Layer Support Tools for details about the metadata). Specific analysis types, such as anomaly detection or prediction, require that appropriate models are trained using historical data and used to assess new data. The data analytics methods can be used to train such models and use existing models to operate on new data. The analytics metadata storage is used as a storage mechanism to hold these pre-trained models to make them available for online usage.

The analytics metadata storage is implemented as a MongoDB\(^2\) database accompanied by a Node.js\(^3\) server offering a RESTful interface. The operations that were implemented and tested for the ACTIVAGE Analytics metadata storage API are the following:

- `/metadataStorage/createModel`
- `/metadataStorage/getModel`
- `/metadataStorage/editModel`
- `/metadataStorage/deleteModel`

A model is essentially a MongoDB document, i.e. a JSON object, containing a unique ID and a set of parameters that may be different depending on the model. On the following tables sample inputs and expected outputs, considering the content already stored in the tested instance of the database, can be seen.

\(^{2}\) [https://www.mongodb.com/](https://www.mongodb.com/)
\(^{3}\) [https://nodejs.org/en/](https://nodejs.org/en/)
## API description:

### /createModel operation

**Table 19 API /createModel**

**/createModel**

Creates a new analytics model in the metadata storage. This is implicitly called when training a data analytics method.

<table>
<thead>
<tr>
<th>Input JSON schema</th>
</tr>
</thead>
</table>
| `{  
  "type": "object",  
  "properties": {  
    "modelID": {  
      "type": "string",  
      "description": "A unique ID for the model."  
    },  
    "modelParams": {  
      "type": "object",  
      "description": "The model parameters."  
    }  
  }  
} |

<table>
<thead>
<tr>
<th>JSON example</th>
</tr>
</thead>
</table>
| `{  
  "modelID": "arimaModel_1",  
  "modelParams": {  
    "arParams": [1.5, 0.3, 0.7, 0.1],  
    "maParams": [0.2, 0.3, 0.1, 0.1]  
  }  
} |

<table>
<thead>
<tr>
<th>Output JSON schema</th>
</tr>
</thead>
</table>
| `{  
  "type": "object",  
  "properties": {  
    "message": {  
      "type": "string",  
      "description": "A success or error message."  
    }  
  }  
} |

<table>
<thead>
<tr>
<th>JSON example</th>
</tr>
</thead>
</table>
| `{  
  "message": "The model was created successfully."  
} |
Returns the stored parameters for an requested model.

**Input**

**JSON schema**

```json
{
  "type": "object",
  "properties": {
    "modelID": {
      "type": "string",
      "description": "The unique ID of the model to retrieve."
    }
  }
}
```

**JSON example**

```json
{
  "modelID": "arimaModel_1"
}
```

**Output**

**JSON schema**

```json
{
  "type": "object",
  "properties": {
    "modelID": {
      "type": "string",
      "description": "The model's unique ID."
    },
    "modelParams": {
      "type": "object",
      "description": "The model parameters."
    }
  }
}
```

**JSON example**

```json
{
  "modelID": "arimaModel_1",
  "modelParams": {
    "arParams": [1.5, 0.3, 0.7, 0.1],
    "maParams": [0.2, 0.3, 0.1, 0.1]
  }
}
```

/`editModel` operation

Table 21 API /getModel

/`editModel`

Edits an existing model in the metadata storage.

**Input**

**JSON schema**

```json
{
  "type": "object",
  "properties": {
    "modelID": {
      "type": "string",
      "description": "The model's unique ID."
    },
    "modelParams": {
      "type": "object",
      "description": "The new model parameters."
    }
  }
}
```


**/deleteModel operation**

Table 22 API /deleteModel

**/deleteModel**

Deletes a model from the metadata storage.

### Input

<table>
<thead>
<tr>
<th>JSON schema</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;type&quot;: &quot;object&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;properties&quot;: {</td>
<td></td>
</tr>
<tr>
<td>&quot;modelID&quot;: {</td>
<td></td>
</tr>
<tr>
<td>&quot;type&quot;: &quot;string&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;description&quot;: &quot;The unique ID of the model to delete.&quot;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSON example</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{ &quot;modelID&quot;: &quot;arimaModel_1&quot; }</td>
<td></td>
</tr>
</tbody>
</table>
Testing for the analytics metadata storage is performed by testing the RESTful API offered by the implemented Node.js server, and specifically by testing whether the provided input produces expected output. This type of testing is described graphically in Figure 89.

Since MongoDB documents acquire a unique MongoDB identifier that is not deterministic, in cases where the expected output is a MongoDB result, the comparison of the produced output to the expected output is performed in terms of whether the produced output contains the properties and values of the expected output, rather than comparing the two outputs in an absolute manner.

The Mocha\(^4\) testing environment is used for testing, with the Chai\(^5\) JavaScript library used for the composition of the assert statements in a format closer to natural language. An example Mocha test description block is presented in Figure 90.

\(^4\) https://mochajs.org/
\(^5\) https://www.chaijs.com/
D5.3 Intermediate Validation Results

```javascript
describe('Tests', function() {
  describe('createModel', function() {
    it('should return correct output in example input', function(done) {
      let input = {
        "modelID": "test_model_1",
        "modelParameters": {
          "$": 1,
          "$": 2
        }
      };
      let expectedOutput = {"message": "The model was created successfully"};
      chai.request(server)
        .post('/createModel')
        .send(input)
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.an('object');
          expect(res.body).to.eql(expectedOutput);
          done();
        });
    });
  });

  describe('getModel', function() {
    it('should return output that contains expected properties', function(done) {
      let input = {
        "modelID": "test_model_1"
      };
      chai.request(server)
        .post('/getModel')
        .send(input)
        .end((err, res) => {
          expect(res).to.have.status(200);
          expect(res.body).to.be.an('object');
          expect(res.body).to.have.property("modelID").to.eql("test_model_1");
          expect(res.body).to.have.property("modelParameters").to.eql({
            "$": 1,
            "$": 2
          });
          done();
        });
    });
  });
});
```

Figure 90: Example Mocha testing block for the "createModel" and "getModel" analytics metadata storage services.

The Mocha tests are embedded in the source code of the analytics metadata storage server and can be initiated through the command line. The results of an example unit test for the "createModel" and "getModel" services are depicted in Figure 91.

![Mocha test output](image)

Figure 91: The output of the Mocha integration testing for the "createModel" and "getModel" analytics metadata storage services.

Details about the testing environment used can be found below, in Section Error! Reference source not found., since the same environment is also used for testing the Data Analytics services.
E.3 Data Integration Engine

The Data Integration Engine is responsible for providing a common point of access to the distributed storage of ACTIVAGE data. The Data Integration Engine is built on top of the ACTIVAGE Semantic Interoperability Layer, handled by Task 3.4, which is responsible for providing common semantics to the diverse data namespaces and schemas of the IoT platforms databases. Using this common naming infrastructure, namely the unified ACTIVAGE data model, the Data Integration Engine is responsible for providing higher level applications, such as user queries and data analytics, with a location-agnostic means of accessing the data, through web APIs. In other words, it provides mechanisms to retrieve data for a specific purpose, regardless of whether they are stored in a single database or in a multitude of databases.

The implementation of the Data Integration Engine addresses the requirements of the DSs regarding data access through a common Web interface. The architecture of the ACTIVAGE project is federated: There are a number of distributed databases, under different IoT platforms, where sensor data are collected. The IoT platforms and associated databases can be added or removed from the federation. In such architecture, in order to provide the upper layers with a seamless common point of data access, certain dedicated mechanisms need to be employed, that are not commonly present in a single-database storage model.

The correct performance of this component is tested through a functional testing. As mentioned in a previous section, functional testing has the following steps:

- Identify functions that the software has to perform.
- Create input data based on the function’s specifications.
- Determine the output based on the function’s specifications.
- Execute the test
- Compare the actual and expected outputs.

The operations in the ACTIVAGE Data Integration API are tested, as they are identified as the functions that the REST API must perform regarding Indexing:

- /api/createIndex
- /api/deleteIndex/{id}
- /api/getAllIndex
- /api/getIndex/{id}
- /api/updateIndex

![Figure 92: The operations in the ACTIVAGE Data Integration API indexing tested](image)
On the following tables sample inputs and expected outputs, considering the content already stored in the tested instance of the database, can be seen.

API description:

**Test /createIndex operation**

<table>
<thead>
<tr>
<th>Input</th>
<th>JSON schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;properties&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;options&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;properties&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;indexBy&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;description&quot;: &quot;The field or property by which to index the data.&quot;</td>
</tr>
<tr>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td>&quot;parameters&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;description&quot;: &quot;Any parameters that the specific index field needs in order to create the index.&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSON example</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{</td>
</tr>
<tr>
<td>&quot;options&quot;: {</td>
</tr>
<tr>
<td>&quot;indexBy&quot;: &quot;time_window&quot;,</td>
</tr>
<tr>
<td>&quot;parameters&quot;: {</td>
</tr>
<tr>
<td>&quot;windowSize&quot;: 100</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>JSON schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;properties&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;indexID&quot;: {</td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;string&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;description&quot;: &quot;A unique ID identifying the created index.&quot;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

Table 23. API /createIndex

/createIndex

Creates an index for the distributed data, based on specific indexing options. The described API will be further modified after database implementation proceeds further.
### D5.3 Intermediate Validation Results

#### JSON Example

```json
{
  "indexID": "index_134"
}
```

#### Test validation

**Figure 93: /createIndex functional sample test**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataIndex</td>
<td>{ &quot;options&quot;: { &quot;indexBy&quot;: &quot;string&quot;, &quot;parameters&quot;: {} }, &quot;indexBy&quot;: &quot;string&quot;, &quot;parameters&quot;: {} }</td>
</tr>
</tbody>
</table>

Parameter content type: [application/json](application/json)

<table>
<thead>
<tr>
<th>Response Messages</th>
<th>HTTP Status Code</th>
<th>Reason</th>
<th>Response Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>201</td>
<td>Created</td>
<td></td>
</tr>
<tr>
<td></td>
<td>401</td>
<td>Unauthorized</td>
<td></td>
</tr>
<tr>
<td></td>
<td>403</td>
<td>Forbidden</td>
<td></td>
</tr>
<tr>
<td></td>
<td>404</td>
<td>Not Found</td>
<td></td>
</tr>
</tbody>
</table>

Response success

- Http Status Code 201
- Body
Figure 94: /createIndex functional Response validation test

Test /updateIndex operation

Table 24. API /updateIndex

/updateIndex

Updates an index for the distributed data, based on specific indexing options. The described API will be further modified after database implementation proceeds further.

```
{  
  "type": "object",  
  "properties": {  
    "options": {  
      "type": "object",  
      "properties": {  
        "indexID": {  
          "type": "string",  
          "description": "The unique ID of the index to update."  
        },  
        "indexBy": {  
          "type": "string",  
          "description": "The field or property by which to index the data."  
        }  
      }  
    }  
  }  
}
```
D5.3 Intermediate Validation Results

```json
{
    "parameters": {
        "type": "object",
        "description": "Any parameters that the specific index field needs in order to create the index."
    }
}
```

```json
{
    "options": {
        "indexID": "index_134",
        "indexBy": "time_window",
        "parameters": {
            "windowSize": 100
        }
    }
}
```

```json
{
    "indexID": "index_134"
}
```

Test validation
### Response body success

- **Http Status code:** 200
- **Body**
### /deleteIndex/{id} operation

Deletes an index for the distributed data given by index id

**Input**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>5e578e13eb799ab50001d33dcc</td>
<td>id</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

**Output**

- No body response
- Header response

**Request URL**

http://activage.datascienceinstitute.ie:8025/api/deleteIndex/5c454aae799ab50001d33dcc
D5.3 Intermediate Validation Results

"x-dataintergrationapp-alert": "A dataIndex is deleted with identifier 5c454aae799ab50001d33dcc"

Test validation

Figure 97: /deleteIndex/{id} functional sample test

Response

- **Http status code success: 200**
- **Body (not reponse body)**

Header response: "x-dataintergrationapp-alert": "A dataIndex is deleted with identifier 5c454aae799ab50001d33dcc"
D5.3 Intermediate Validation Results

Fig. 98: `/deleteIndex/{id}` functional Response validation test

Test `/getAllIndex` operation

Table 26. API `/getAllIndex`

`/getAllIndex`

Get all indexes in the database with input parameters: page, size

<table>
<thead>
<tr>
<th>Query params</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>offset</td>
<td></td>
<td>Page number of the requested page</td>
<td>query</td>
<td>long</td>
</tr>
<tr>
<td>page</td>
<td></td>
<td></td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>pageSize</td>
<td></td>
<td>Size of a page</td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>pageNumber</td>
<td></td>
<td></td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>pagePath</td>
<td></td>
<td></td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>pageId</td>
<td></td>
<td></td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>offset</td>
<td></td>
<td></td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>size</td>
<td></td>
<td></td>
<td>query</td>
<td>integer</td>
</tr>
<tr>
<td>sort</td>
<td></td>
<td>Sorting criteria in the format property as (desc). Default sort order is ascending. Multiple sort criteria are supported.</td>
<td>query</td>
<td>Any(string)</td>
</tr>
</tbody>
</table>

Output: JSON schema

```json
[
]
```
D5.3 Intermediate Validation Results

```json

[  
  {  
    "id": "5b717da510ce59059c140955",  
    "indexBy": "Hung Nguyen",  
    "parameters": {  
      "param1": "some data here"  
    }  
  },  
  {  
    "id": "5b9bd203799ab50001b5219d",  
    "indexBy": "Hung Nguyen",  
    "parameters": {  
      "windows_time": 700,  
      "other_params": "data"  
    }  
  }  
]
```

Tests validation
D5.3 Intermediate Validation Results

Figure 99: /getAllIndex functional sample test

Response validation
D5.3 Intermediate Validation Results

Curl

curl -k GET —header 'Accept: application/json' —header 'Authorization: Bearer null' 'http://activage.datascienceinstitute.let:8025/api/getAllIndex'

Request URL

http://activage.datascienceinstitute.let:8025/api/getAllIndex

Response Body

```
"parameters": {
  "windows_time": 700,
  "other_params": "data"
},
{"id": "5b9bd264f709ba50001b5219e",
"indexBy": "Hoan Nguyen",
"parameters": {
  "windows_time": 800
},
{"id": "5b9b51a697bd50001b5219f",
"indexBy": "Mr. Ku",
"parameters": {
  "param": "param"
},
{"id": "5b761b799ba50001673558",
```

Response Code

200

Response Headers

```
"pragma": "no-cache",
"date": "Mon, 11 Jan 2019 05:38:55 GMT",
"content-encoding": "gzip",
"x-content-type-options": "nosniff",
"transfer-encoding": "chunked",
"content-type": "application/json;charset=UTF-8",
"cache-control": "no-cache, no-store, max-age=0, must-revalidate",
"connection": "keep-alive",
"link": "<https://api/activage.datascienceinstitute.let:8025/api/getAllIndex?page=0&size=20>; rel="last",<https://api/activage.datascienceinstitute.let:8025/api/getAllIndex?page=0&size=20>; rel="first",<https://api/activage.datascienceinstitute.let:8025/api/getAllIndex?page=0&size=20>; rel="next",<https://api/activage.datascienceinstitute.let:8025/api/getAllIndex?page=0&size=20>; rel="prev",x-total-count": "6",x-page": "0",
```

Figure 100: /getAllIndex functional Response validation test

Test /getIndex/{id} operation

Table 27. API /getIndex/{id}

/getIndex/{id}

Get Index by id to retrieve database index information given by id

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>5b761b799ba50001673558</td>
<td>id</td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

Response Messages
Output

JSON

schema

```
{
  "id": "string",
  "indexBy": "string",
  "parameters": {}
}
```

JSON

example

```
{
  "id": "5bff761b799ab50001673550",
  "indexBy": "Hung Nguyen",
  "parameters": {
    "windows_time": 200
  }
}
```

Test validation
D5.3 Intermediate Validation Results

Figure 101: `/getIndex/{id}` functional sample test

Response validation
D5.3 Intermediate Validation Results

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer null' 'http://activage.datascienceinstitute.ie/8025/api/getIndex/5bffe7eb799a5b00001673550'
```

Request URL
```
http://activage.datascienceinstitute.ie/8025/api/getIndex/5bffe7eb799a5b00001673550
```

Response Body
```
{
   "id": "5bffe7eb799a5b00001673550",
   "indexBy": "Hung Nguyen",
   "parameters": {
      "win\dows\-line": 200
   }
}
```

Response Code
```
200
```

Response Headers
```
{
   "pragma": "no-cache",
   "date": "Mon, 21 Jan 2019 05:47:55 GMT",
   "content-encoding": "gzip",
   "transfer-encoding": "chunked",
   "content-type": "application/json;charset=UTF-8",
   "cache-control": "no-cache, no-store, must-revalidate",
   "connection": "keep-alive",
   "x-ssl-protocol": "SSLv3",
   "user-agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36",
   "expires": "0"
}
```

Figure 102: /getIndex/{id} functional Response validation test
Appendix F  Analytics

F.1 Unit testing

The units to be tested in the AIoTES Analytics component are the analytics methods supported (see deliverable D4.5 "Data Layer Support Tools"). Specifically, the operations that were implemented and tested so far for the ACTIVAGE Analytics API are the following:

- /summary
- /histogram
- /lof
- /lof/train
- /kmeans
- /dbscan

On the following tables sample inputs and expected outputs can be seen for each one of the aforementioned operations:

/summary operation

Table 28 API /summary

<table>
<thead>
<tr>
<th>Input</th>
<th>JSON schema</th>
<th>JSON example</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON schema</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;properties&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;dataDesc&quot;: {$ref: &quot;/definitions/dataDesc&quot;}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>JSON example</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;dataDesc&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;query&quot;: &quot;select * from env_sensors where ds = 'DS_GRC'&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;expression&quot;: &quot;temperature&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>Output</th>
<th>JSON schema</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON schema</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;object&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;properties&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;min&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;type&quot;: &quot;number&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;description&quot;: &quot;The minimum value of the data.&quot;</td>
<td></td>
</tr>
</tbody>
</table>
The JSON example:

```json
{
  "min": 17.8,
  "max": 28.5,
  "mean": 25.3,
  "median": 26.1,
  "perc25": 21.2,
  "perc75": 27.9,
  "stdev": 2.5,
  "skewness": -0.6,
  "kurtosis": 0.8
}
```

The histogram operation:

**Table 29 API /histograms**

/ histogram

Computes a histogram of the provided numeric data.
### Input JSON schema

```json
{
    "type": "object",
    "properties": {
        "dataDesc": {"$ref": "#/definitions/dataDesc"},
        "options": {
            "type": "object",
            "properties": {
                "nbins": {
                    "type": "number",
                    "description": "The number of bins to use."
                },
                "normalized": {
                    "type": "boolean",
                    "description": "Whether the histogram is normalized."
                }
            }
        }
    }
}
```

### JSON example

```json
{
    "dataDesc": {
        "query": "select * from env_sensors where ds = 'DS_GRC'",
        "expression": "temperature + humidity * 100"
    },
    "options": {
        "nbins": 10,
        "normalized": true
    }
}
```

### Output JSON schema

```json
{
    "type": "object",
    "properties": {
        "values": {
            "type": "array",
            "description": "The computed histogram as a vector.",
            "items": {"type": "number"}
        }
    }
}
```

### JSON example

```json
{
    "values": [0.00, 0.09, 0.13, 0.22, 0.26, 0.17, 0.09, 0.04, 0.00, 0.00]
}
```

### /lof operation

The `/lof operation` provides anomaly scores for the input data, using the LOF method.
Input JSON schema

```json
{
  "type": "object",
  "properties": {
    "dataDesc": {
      "$ref": "#/definitions/dataDesc",
      "modelParams": {
        "type": "object",
        "description": "The model parameters. Either this or 'modelID' should be used.",
        "properties": {
          "normalValues": {
            "type": "array",
            "description": "The set of normal values to compare against."
          }
        }
      }
    },
    "modelID": {
      "type": "string",
      "description": "The ID of a pre-trained model, stored in the analytics metadata storage. Either this or 'modelParams' should be used."
    },
    "options": {
      "numNeighbors": {
        "type": "number",
        "description": "The number of neighbors to use."
      }
    }
  }
}
```

JSON example

```json
{
  "dataDesc": {
    "query": "select * from env_sensors where ds = 'DS_GRC'",
    "expression": "temperature"
  },
  "modelParams": {
    "normalValues": [25.2, 24.8, 26.9, 23.3, 27.2, 24.6]
  },
  "options": {
    "numNeighbors": 5
  }
}
```

Output JSON schema

```json
{
  "type": "object",
  "properties": {
    "dataDesc": {
      "query": "select * from env_sensors where ds = 'DS_GRC'",
      "expression": "temperature"
    },
    "modelID": "lof_model_1",
    "options": {
      "numNeighbors": 5
    }
  }
}
```
D5.3 Intermediate Validation Results

### JSON example

```json

"anomalyScores": {
  "type": "array",
  "description": "The anomaly scores for the input data."
},

}```

### `/lof/train` operation

**Table 31 API /lof/train**

#### `/lof/train`

Trains a LOF anomaly detection model

**Input**

**JSON Schema**

```json

{
  "type": "object",
  "properties": {
    "trainingData": {"$ref": "/#definitions/dataDesc"},
    "modelID": {
      "type": "string",
      "description": "A unique ID to use when referring to the model."
    }
  }
}

```

**JSON Example**

```json

{
  "trainingData": {
    "query": "select temp from temp_sensors where date < '2018-05-10' and ds = 'DS_GRC'",
    "expression": "temp"
  }
}

```

#### Output

**JSON Schema**

```json

{
  "type": "object",
  "properties": {
    "modelID": {
      "type": "string",
      "description": "The ID of the trained model, stored in the analytics metadata storage."
    },
    "modelParams": {
      "type": "object",
      "description": "The computed model parameters.",
      "properties": {
        "normalValues": {
          "type": "array",
          "description": "The set of normal values to compare against.",
          "items": {"type": "number"}
        }
      }
    }
  }
}

```
### /kmeans

Clusters the input data using the k-means clustering algorithm

<table>
<thead>
<tr>
<th>Input</th>
<th>JSON schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{</td>
</tr>
</tbody>
</table>
|       |   "type": "object",
|       |   "properties": { |
|       |     "dataDesc": {"$ref": "#/definitions/dataDesc"},
|       |     "options": { |
|       |       "type": "object",
|       |       "properties": { |
|       |         "numClusters": { |
|       |           "type": "number",
|       |           "description": "The number of clusters to create."
|       |         },
|       |         "distanceType": { |
|       |           "type": "string",
|       |           "description": "The distance function to use between points."
|       |         },
|       |         "maxIterations": { |
|       |           "type": "number",
|       |           "description": "The maximum number of iterations for which to run the algorithm."
|       |         }
|       |     },
|       |   },
|       | }            |
|       | }            |

<table>
<thead>
<tr>
<th>JSON example</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;dataDesc&quot;: {</td>
<td></td>
</tr>
<tr>
<td>&quot;data&quot;: [</td>
<td></td>
</tr>
</tbody>
</table>
|       [5.1, 3.5, 1.4, 0.2]],
|       [5.8, 4.0, 1.2, 0.2]],
|       [5.7, 4.4, 1.5, 0.4]],
|       [5.4, 3.9, 1.3, 0.4]],
|       [5.1, 3.5, 1.4, 0.3]],
|       [5.7, 3.8, 1.7, 0.3]],
|       [5.1, 3.8, 1.5, 0.3]],
|       [5.4, 3.4, 1.7, 0.2]],
|   ]           |

### Table 32 API /kmeans

<table>
<thead>
<tr>
<th>JSON example</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td></td>
</tr>
</tbody>
</table>
|   "modelID": "lof_model_1",
|   "modelParams": { |
|     "normalValues": [25.2, 24.8, 26.9, 23.3, 27.2, 24.6]
| }            |

```
{            |
  "modelID": "lof_model_1",
  "modelParams": { |
    "normalValues": [25.2, 24.8, 26.9, 23.3, 27.2, 24.6]
  }            |
}            |
```

<table>
<thead>
<tr>
<th>JSON example</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td></td>
</tr>
</tbody>
</table>
|   "modelID": "lof_model_1",
|   "modelParams": { |
|     "normalValues": [25.2, 24.8, 26.9, 23.3, 27.2, 24.6]
| }            |

```
{            |
  "modelID": "lof_model_1",
  "modelParams": { |
    "normalValues": [25.2, 24.8, 26.9, 23.3, 27.2, 24.6]
  } |
} |
```
Output JSON schema

```
{
  "type": "object",
  "properties": {
    "clusterLabels": {
      "type": "array",
      "description": "An array of labels, one for each input record, denoting the cluster it belongs to."
    }
  }
}
```

JSON example

```
{
  "clusterLabels": [0, 1, 1, 2, 0, 2, 0, 1, 0]
}
```

/dbscan operation

**Table 33 API/dbscan**

Clusters the input data using the DBSCAN clustering algorithm

Input JSON schema

```
{
  "type": "object",
  "properties": {
    "dataDesc": {"$ref": "#/definitions/dataDesc"},
    "options": {
      "type": "object",
      "properties": {
        "eps": {
          "type": "number",
          "description": "The neighborhood radius to use."
        },
        "minPts": {
          "type": "number",
          "description": "The minimum number of points to use for determining cluster density."
        },
        "distanceType": {
          "type": "string",
          "description": "The distance function to use between points."
        }
      }
    }
  }
}
```
Each of the aforementioned methods is a function that takes data and options as its input and produces the output of the analysis, as depicted in Figure 103.

![Diagram](image)

Figure 103: The basic unit of the Data Analytics component.
The data analytics methods are implemented as JavaScript functions in Node.js\(^6\), which may call other modules developed in C++ or Python, as necessary, in order to support all relevant analytics functionalities. The input (data/options) and output are in JSON format, which is native to the JavaScript language and facilitates the exposition of the methods as web services.

Unit testing is performed using the Mocha\(^7\) testing framework for Node.js. For each data analytics method, a Mocha testing block is constructed. Testing is mainly performed by providing a fixed input to the method and comparing the output to the expected output for the given input, as depicted in Figure 104. An example of a Mocha testing block used for the k-means data analytics method is presented in Figure 105.

![Diagram of data analytics method with input (data/options) and output (expected output)](image)

**Figure 104:** Comparison of method output to expected output.

---

\(^6\) [https://nodejs.org/en/](https://nodejs.org/en/)

\(^7\) [https://mochajs.org/](https://mochajs.org/)
D5.3 Intermediate Validation Results

Version 1.0  I  2019-02-14  I  ACTIVAGE ©

The Mocha unit tests are embedded in the source code of the data analytics methods and can be initiated through the command line. The results of an example unit test for the k-means method are depicted in Figure 106.

Figure 105: Example Mocha unit testing block for the k-means analytics method.
F.2 Integration testing

The data analytics methods of the AIoTES Analytics component are exposed to the rest of the AIoTES environment as web services. Each data analytics method is wrapped as a RESTful web service within Node.js, accepting its input (both data and options) through the request’s body, in JSON format. Note that the Web services are described in D4.5. There is no link, because they are currently run in a local server. The output of each service is again formatted as output and returned as the response to the caller, as depicted in Figure 107.

Integration testing of the data analytics web API is performed by testing each analytics web service through its RESTful API. Similar to unit testing, the tests are performed using the Mocha testing framework for Node.js, by comparing the output of each web service to the expected output, provided a fixed input, as depicted in Error! Reference source not found.. Given the same input, the output produced by the data analytics services is the same, so the current output can at all times be compared to the known output. The expected output used for testing has been produced by running the first version of the services, with the output checked by an expert regarding its correctness. Once correctness of the output is established, it is further used in testing. An example of a Mocha testing block used in integration testing is presented in Error! Reference source not found..

* [https://mochajs.org/](https://mochajs.org/)
The Mocha integration tests are embedded in the source code of the data analytics services and can be initiated through the command line. The results of an example unit test for the k-means service are depicted in Error! Reference source not found..
F.3 Testing environment

The Mocha\(^9\) testing framework has been used for both unit and integration testing. Mocha runs from within Node.js and allows the description of testing blocks, facilitating the organization of the tests in categories. Each test is a set of assertions verifying that specific steps of execution produce results that are expected either in their type (e.g. number, array, etc.) or in their content (e.g. by comparing to expected outputs).

The assertions themselves are written in the mini-language of the Chai\(^{10}\) assertion library. Chai offers a more human-friendly way to write assertions compared to the standard assertion functions provided by JavaScript. As an example, one can assert that an output is equal to an expected value using the following statement:

\[
\text{expect(output).to.eql(expectedOutput)}
\]

Chai offers such syntax for more complex types of assertions, such as for verifying the result of asynchronous operations, which significantly facilitates testing web service outputs.

---

\(^9\) https://mochajs.org/

\(^{10}\) https://www.chaijs.com/
Appendix G  Marketplace

The ACTIVAGE Marketplace is a one-stop-shop for providing, discovering and deploying applications built on AIoT ES. As such, it is a high level deployment tool, very close to end-users. Specifically, it is intended for applications users and developers alike. Developers, either internal from the Deployment Sites, or external, from the Open Call and beyond, can upload, promote and monetize their applications. Users, including healthcare professionals, carers and deployers at the Deployment Sites, external third parties and more, can search, discover get free or buy applications.

The current version 1.0 of the Marketplace is a standard web portal providing the above functionality supporting (App) Users, Developers and (Marketplace) Administrators. The Marketplace homepage for version 1.0 is shown on Error! Reference source not found. while its specifications are presented in the respective version 1.0 of D4.3. For context, a lists of its main functionality list is provided here on Table 6:. Notably, no distinction needs to be made between ACTIVAGE participants and general users for roles. Usually, Marketplace administrators are ACTIVAGE members but Marketplace developers can be external adopters while Marketplace users can be the general public. The Marketplace does not need to have knowledge of their ACTIVAGE membership.

Note that the Marketplace does not do any distinction between the Activage / non activage roles.
Figure 111. Marketplace Homepage

Table 34: Marketplace Functionality per User Role

<table>
<thead>
<tr>
<th>Marketplace User Role</th>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[User]</td>
<td>View Most Downloaded and Top Rated Apps</td>
<td>See overall Marketplace statistics</td>
</tr>
<tr>
<td></td>
<td>Search All Apps</td>
<td>Search with various criteria</td>
</tr>
<tr>
<td></td>
<td>View supported Platforms and respective Apps</td>
<td>View AloTES Platforms and information about them with respect to hosted Apps</td>
</tr>
<tr>
<td></td>
<td>View and Edit User Profile, Installed Apps, Wishlist, All Comments</td>
<td>Manage user profile, maintain lists of Apps, manage comments and comply to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GDPR</td>
</tr>
</tbody>
</table>
For a full list of GUIs and functions integrated in AloTES please refer to D5.5.

For a full description of the Marketplace component, including implementation and technology stack, please refer to D4.3 (1st & 2nd version).

The verification and testing procedure of the Marketplace follows that of a standard web portal.

The Marketplace coding follows the MVC paradigm and best practices for any web portal, placing emphasis on well-documented functionality, issues and tracking, versioning and iterative testing throughout development. This verification procedure is reported here through the git/GitHub statistics exported directly through the platform. No additional effort was invested in developing unit tests in order to speed up the process and deliver an Alpha (1.0) Phase version, a Beta and a Final (2.0) version to satisfy all involved parties (project partners, open calls and external adopters).

As functionality develops, emphasis is also placed on the front-end UI/UX in terms of Performance, Accessibility, Best (Web Development) Practices and SEO for outreach. This was verified through established online verification services and before and after certain optimization were performed.

### 6.1 Issue Tracking

Issue tracking helps both project managers and developers to create, monitor and resolve issues throughout the engineering process. Issue is a generic term that in our case can refer to both features and bugs for the Marketplace. In our process, a new feature represents new functionality or new UI/UX elements to be added. Bugs naturally refer to abnormal behavior of the application or errors.

Issue tracking can be maintained over a variety of project management software or even documents. However, since Marketplace development employs the git version control
D5.3 Intermediate Validation Results

system\textsuperscript{11}, the GitHub platform\textsuperscript{12} allows Issue Tracking with issues linked to git versions of code. Using this platform, and its exporting function, the actual creation, documentation and resolution of Marketplace issues, features and bugs, is presented below.

To begin with, Error! Reference source not found. shows the overall GitHub Issue Statistics from March 2018 to November 2018.

<table>
<thead>
<tr>
<th>GitHub Issue Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Issues</td>
<td>187</td>
</tr>
<tr>
<td>Resolved</td>
<td>176</td>
</tr>
<tr>
<td>Open</td>
<td>11</td>
</tr>
</tbody>
</table>

Each of those issues was always assigned a label, i.e. category, to classify it as a Bug, a Code optimization issue, an Enhancement with new features or a Design improvement. The distribution between those categories was equal, as shown on Error! Reference source not found., asserting the homogeneous development of new features, front-end, code quality and testing. Meanwhile, bugs were discovered, yet kept to a minimum and always resolved as the majority of issues – no remaining issues belong to the Bug category.

Regarding the time to resolve each issue, Error! Reference source not found. plots the duration from opening an issue to its closing, measured in days (y-axis). All issues are displayed (on the x-axis, although not all not all issue identification numbers are apparent, due to space limitations) on an increasing duration order. This clarifies the majority of issues were resolved in well under 5 days maximum time, through intense development while few larger issues (features) last longer than 50 and up to 100 days.

\textsuperscript{11} The Git version control system: https://git-scm.com/
\textsuperscript{12} GitHub: https://github.com/
Issues were closed quickly after they were opened. A view of issues opened, closed and those that remained open (pending) per month of development in a timeline fashion, from March 2018 to November 2018, is shown on Figure 113. Even in months were up to 75 issues were opened, that many were closed, leaving open issues as low as up to 11 every month.

These insights show for a consistent and active development of features while constantly testing and resolving errors through testing.
6.2 Performance, Accessibility, Best Practices and SEO Testing

To objectively verify performance, accessibility, best practice compliance and SEO in web development, the web.dev platform\textsuperscript{13} was employed. Web.dev employs the Lighthouse tool provided by Google\textsuperscript{14} for improving the quality of web pages.

The tools automatically check:

- **Performance**: Audits for metrics like first paint and time to interactive to determine lag. Also, PWA, which assesses your page against the baseline Progressive Web App Checklist.
- **Best Practices**: Looks for everything from HTTPS usage to correct image aspect ratios.
- **SEO**: Checks for best practices to ensure your site is discoverable.
- **Accessibility**: Checks for common issues that may prevent users from accessing your content.

Two rounds of testing were performed.

The first round was performed after all functionality for version 1.0 was implemented and all bugs resolved but no intended optimization for accessibility and SEO was considered. The results are shown on Error! Reference source not found.

The portal also offers specific feedback for each audit test performed giving recommendations and coding hints on improving. Such a sample from the SEO section is shown on Error! Reference source not found.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{lighthouse-testing.png}
\caption{Lighthouse testing before optimizations}
\end{figure}

\textsuperscript{13} web.dev: \url{https://web.dev/}

\textsuperscript{14} Lighthouse tools for Web Developers: \url{https://developers.google.com/web/tools/lighthouse/}
Figure 116. Sample of SEO recommendations provided by Lighthouse

Optimizations were indeed performed such as:

- Including metadata and title elements for SEO
- Optimizing visual elements such as contrast and sizing for accessibility
- Adding alternative titles and tags for accessibility and mobile ports
- Performance improvements
- Best practices in coding

The outcome version 1.0 managed to pass all audits 100% according to Lighthouse as shown on Error! Reference source not found..

This ensures a high-level of verification of the Marketplace’s Performance, Accessibility, Best Practices and SEO, according to an established web development tool.
## D5.3 Intermediate Validation Results

### Figure 117. Lighthouse testing after optimization and passed audits

<table>
<thead>
<tr>
<th>Category</th>
<th>Score</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Practices</td>
<td>100</td>
<td>Passed audits: 14 audits</td>
</tr>
<tr>
<td>Accessibility</td>
<td>100</td>
<td>Additional items to manually check: 11 audits, Passed audits: 11 audits, Not applicable: 24 audits</td>
</tr>
<tr>
<td>SEO</td>
<td>100</td>
<td>Additional items to manually check: 2 audits, Passed audits: 9 audits, Not applicable: 2 audits</td>
</tr>
<tr>
<td>Performance</td>
<td>90</td>
<td>Metrics: First Contentful Paint: 1.9 s, First Meaningful Paint: 2.0 s, Speed Index: 3.5 s, First CPU Idle: 3.5 s, Time to Interactive: 4.2 s, Estimated Input Latency: 10 ms</td>
</tr>
</tbody>
</table>

These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

These checks ensure that your page is optimized for search engine results ranking. There are additional factors Lighthouse does not check that may affect your search ranking. Learn more.