## Project Deliverable

<table>
<thead>
<tr>
<th>Project Number:</th>
<th>Project Acronym:</th>
<th>Project Title:</th>
</tr>
</thead>
<tbody>
<tr>
<td>287901</td>
<td>BUTLER</td>
<td>Ubiquitous, secure internet-of-things with Location and context awareness</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instrument:</th>
<th>Thematic Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Project</td>
<td>Internet of things</td>
</tr>
</tbody>
</table>

### Title

**D3.2 Integrated System Architecture and Initial Pervasive BUTLER proof of concept**

<table>
<thead>
<tr>
<th>Contractual Delivery Date:</th>
<th>Actual Delivery Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2013</td>
<td>October 2013</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start date of project:</th>
<th>Duration:</th>
</tr>
</thead>
<tbody>
<tr>
<td>October, 1st 2011</td>
<td>36 months</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Organization name of lead contractor for this deliverable:</th>
<th>Document version:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERC</td>
<td>V1.50</td>
</tr>
</tbody>
</table>

Dissemination level (Project co-funded by the European Commission within the Seventh Framework Programme):

- **PU** Public
- **PP** Restricted to other programme participants (including the Commission)
- **RE** Restricted to a group defined by the consortium (including the Commission)
- **CO** Confidential, only for members of the consortium (including the Commission)

<table>
<thead>
<tr>
<th>Dissemination level</th>
<th>X</th>
</tr>
</thead>
</table>
Authors (organizations):

ERC, CEA, GTO, KUL, INNO, ISMB, TIL, ZIGPOS

Abstract:

“Integrated System Architecture and Initial Pervasive BUTLER proof of concept” is the deliverable D3.2 produced by the project BUTLER in its second year of activity.

The project BUTLER is a 3-year integrated research project whose research purpose is to enable the development of secure and smart life assistant applications thanks to a context and location aware, pervasive information system. For that purpose, project BUTLER is:

a. Improving/creating enabling technologies to implement a well-defined vision of secure, pervasive and context-aware IoT, where links are inherently secure (from PHY to APP layers) applications cut across different scenarios (Home, Office, Transportation, Health, etc.), and the network reactions to users are adjusted to their needs (learned and monitored in real time).

b. Integrating/developing a new flexible Device-centric network architecture where platforms (devices) function according to three well-defined categories: SmartObject (sensors, actuators, gateways to integrate the former), SmartMobile (user’s personal device) and SmartServers (providers of contents and services)/

c. Building a series of field trials, which progressively integrate and enhance state-of-the-art technologies to showcase BUTLER’s secure, pervasive and context-aware vision of IoT.

The current report is a major milestone of the project making the link between the users’ requirements produced in the inception Task 1.1 (T1.1) of the project and the field implementations initiated in the second period.

Keywords:

BUTLER, IoT, I-T-A, FI-WARE, SmartObject, SmartObject Gateway, SmartServer, SmartMobile, context management, behavior prediction, localization management, resource, service, virtual entity, data source, data offerings

Disclaimer

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Any liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No license, express or implied, by estoppels or otherwise, to any intellectual property rights are granted herein. The members of the project BUTLER do not accept any liability for actions or omissions of BUTLER members or third parties and disclaims any obligation to enforce the use of this document. This document is subject to change without notice.
# Revision History

The following table describes the main changes done in the document since it was created.

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.0</td>
<td>Jan 2013</td>
<td>Creation (ERC)</td>
</tr>
<tr>
<td>V0.01</td>
<td>April 2013</td>
<td>Index update after preliminary agreement on the BUTLER architecture (ERC, CEA)</td>
</tr>
<tr>
<td>V0.02</td>
<td>9 May 2013</td>
<td>Minor fixing (ERC)</td>
</tr>
<tr>
<td>V0.1</td>
<td>10 May 2013</td>
<td>BUTLER Information model added (ERC)</td>
</tr>
</tbody>
</table>
| V0.2     | 01 July 2013| Context Management (ERC)  
Localization Management (ISMB)  
User Behavior Capture (KUL)  
Mapping Requirements (ZIGPOS)  
SW Updated in SmartObject (ZIGPOS) |
| V0.21    | 18/07/2013 | Context Management (Updated) (ERC)                                           |
| V0.3     | 06/08/2013 | 4.5.7 – Localization Management (ISMB) – reviewed  
4.5.8 – SmartObject Addressing (ISMB) – new  
4.6.1.2 – Network Monitoring (ISMB) – new  
4.6.2.7 – Localization Manager (ISMB)  
SW Updated in SmartObject, Mapping requirements (ZIGPOS)  
4.12.2.3 – IoT Protocol Adapter (ST) |
| V0.4     | 09/08/2013 | 4.5.3 – Security Services (GTO)  
Energy forecasting (ERC) |
| V0.5     | 28/08/2013 | 4.5.10 – Data Discovery and Data Marketplace (ERC) |
| V0.6     | 12/09/2013 | 4.12.1.6 – IoT Protocol Adapter (ISMB Review)  
4.12.2.3 – Persistent Storage (ERC) |
| V0.7     | 16/09/2013 | 4.5.6 and 4.12.2.9 – User Profile Manager (TIL)  
4.5.2 Butler Service and Resource Model (CEA) |
| V0.8     | 25/05/2013 | Reviewed section – 4.3.4 (TST)  
New section – 3.5.3.5 NFC Service (TST)  
Reviewed section – 4.12.2.4 User Behavior Capture (KUL)  
Reviewed – 4.12.3.11 Behavior Exposition (KUL)  
New Section 5.4 SmartHealth Scenario (KUL) |
| V0.9     | 03/10/2013 | 3.5.1.1 – Overview communication later (ISMB)  
4.3 – SmartParking (TST)  
4.6 – SmartShopping (TST)  
General Review (KUL)  
Updated versions – 3.4.5 Context Management (ERC) 3.4.9 Resource discovery and exposition (ERC), Complex Event Processing (ERC), Persistent Storage (ERC) |
| V1.0     | 07/10/2013 | 3.4.4 – SmartObject Lifecycle (CEA)  
3.4.8 – Resource discovery and exposition (CEA)  
3.5.1.4 – Device Directory (CEA/ISMB)  
3.5.2.6 – Resource Access (CEA)  
3.5.2.7 – Simple Event Processing (CEA)  
3.5.2.10 – Device Data Collector/ Actuator (CEA)  
3.5.3.3 – Monitoring Services (CEA)  
3.5.3.4 – Multimedia Services (CEA/ MAYA)  
3.5.3.7 – Management Portal (CEA)  
3.5.3.8 – Data and Service Directory (CEA) |
<p>| V1.1   | 09/10/2013 | 3.5.3.8.8.4. – Data Discovery (ERC) |
|       |           | 3.5.4.2. – Data Management and Marketplace portal (ERC) |
|       |           | 3.5.4.3. – Context Management Portal (ERC) |
|       |           | 3.5.3.5.1. – Energy Forecasting (IHomeLab/ERC) |
|       |           | 4.1. – SmartHome Scenario (Energy forecasting) (IHomeLab) |
|       |           | 3.4.3. – Security Services (updated) (GTO) |
| V1.11  | 14/10/2013 | 4.5.7 – Updated: Localization Management (ISMB) – reviewed |
|       |           | 4.5.8 – Updated: SmartObject Addressing (ISMB) – new |
|       |           | 4.6.1.2 – Updated: Network Monitoring (ISMB) – new |
|       |           | 4.6.2.7 – Localization Manager (ISMB) |
|       |           | SW Updated in SmartObject, Mapping requirements (ZIGPOS) |
|       |           | 3.4.1. – Deployment Model (TIL) |
|       |           | 3.5.1.7 – User directory (TIL) |
|       |           | 3.5.1.10 – Server Directory (TIL) |
|       |           | 3.5.2.9 – User profile Management |
|       |           | 3.5.3.12 – User profile exposition (TIL) |
|       |           | 3.5.3.14 – Authorization Server (ERC) |
|       |           | 3.4.3 – Security Services (ERC on behalf of GTO) |
|       |           | 3.5.3.9 – Resource Exposition (ERC) |
|       |           | 3.5.3.10 – Context Exposition (ERC) |
|       |           | 3.5.2.7 – Full rework of Simple Event Processing (ERC) |
| V1.12  | 15/10/2013 | Initial overall review by document editor |
|       |           | 3.5.3.1 – Services Layer introduction |
| V1.2   | 18/10/2013 | Reviewed at the Torino Plenary Meeting. |
|       |           | Contributions: |
|       |           | 3.5.1.8 – User Authentication (ERC) |
|       |           | Section “NFC Services” (3.5.36) moved to D4.2 |
|       |           | Section “SmartObject Lifecycle” (3.4.4) removed |
|       |           | 3.4.3 – Security Services (reviewed by GTO according to comments) |
|       |           | 3.5.1.6 – Device Authentication (ERC on behalf of GTO) |
|       |           | 3.5.3.14 (Authorization Server) moved to 3.5.1.5 (to ease understanding) |
| V1.21  | 22/10/2013 | 2.3 – Document Structure (ERC) |
|       |           | 3.4.1 – BUTLER Deployment Models (TIL) |
|       |           | 3.4.2 – BUTLER Service and Resource Model (CEA) |
|       |           | 3.4.4 – User Profile Manager (TIL) |
|       |           | 3.4.7 – Resource Discovery and Exposition (CEA) |
|       |           | 3.5.1.8 – User Device Directory (TIL) |
|       |           | 3.5.1.11 – Server Directory (refurbished) (ERC) |
|       |           | 3.5.1.12 – Server Authentication (ERC) |
|       |           | 3.5.1.13 – Server Connectivity Manager (ERC) |
|       |           | 3.5.1.14 – Device Directory (CEA) |
|       |           | 3.5.2.6 – Resource Access (CEA) |
|       |           | 3.5.2.7 – Simple Event Processing (refurbished) (ERC) |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.2.10</td>
<td>Device Data Collector/Actuator (CEA)</td>
</tr>
<tr>
<td>3.5.2.11</td>
<td>Context/Behavior Information Provider (ERC)</td>
</tr>
<tr>
<td>3.5.3.3</td>
<td>Monitoring Services (CEA)</td>
</tr>
<tr>
<td>3.5.3.4</td>
<td>Multimedia Service (CEA)</td>
</tr>
<tr>
<td>3.5.3.6</td>
<td>Management Portal (CEA)</td>
</tr>
<tr>
<td>3.5.3.7.1</td>
<td>Service Registration (CEA)</td>
</tr>
<tr>
<td>3.5.3.7.2</td>
<td>Service Discovery (CEA)</td>
</tr>
<tr>
<td>3.5.3.7.3</td>
<td>Service Composition (CEA)</td>
</tr>
<tr>
<td>3.5.3.7.4</td>
<td>Service Rebinding (CEA)</td>
</tr>
<tr>
<td>3.5.3.11</td>
<td>User Profile Exposition (TIL)</td>
</tr>
<tr>
<td>3.6.5</td>
<td>SmartShopping – Context-aware sparkdeals (TST)</td>
</tr>
<tr>
<td></td>
<td>Refurbishing of overall functional component images (ERC)</td>
</tr>
<tr>
<td>Section 3.5.3.3 moved to 3.5.4.5.3 (now 3.5.3.5.2)</td>
<td></td>
</tr>
<tr>
<td>Additional updates according to Torino review (CEA, ERC, KUL, ISMB, TIL, ST, ZIGPOS)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.3</td>
<td>25/10/2013</td>
<td>Ready for internal review</td>
</tr>
<tr>
<td>V1.4</td>
<td>28/10/2013</td>
<td>Renumbering of the whole document as a result of the introduction of section 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5.3.4 – BUTLER Application Repository (INNO)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5.2 – Localization Service renamed to Localization Exposition and moved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Movements in Key Concepts and Interactions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4.7 – BUTLER Resource Discovery and Exposition merged with 4.4.2 – BUTLER Service and Resource Model (now 4.4.2.3.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4.6 – BUTLER SmartObject Addressing renumbered as 4.4.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4.4 – BUTLER User Profile Management renumbered as 4.4.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5.1.3 – Device Management (CEA). The rest of the section has been renumbered accordingly</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updates by TST, INNO, ISMB, KUL, GTO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ready for final review</td>
</tr>
<tr>
<td>V1.5</td>
<td>29/10/2013</td>
<td>Final Review - INNO</td>
</tr>
</tbody>
</table>
# Table of Content

1. ACRONYMS AND DEFINITIONS ........................................... 13

2. INTRODUCTION .............................................................. 15

   2.1. PURPOSE OF THE DOCUMENT .................................. 15
   2.2. INTENDED AUDIENCE ............................................. 15
   2.3. DOCUMENT STRUCTURE ........................................... 15
   2.4. SUGGESTED PREVIOUS READINGS .............................. 16

3. EXECUTIVE SUMMARY .................................................... 17

4. BUTLER ARCHITECTURE ................................................... 20

   4.1. BUTLER ARCHITECTURAL LAYERING ........................... 20
   4.2. HIGH-LEVEL ARCHITECTURE ..................................... 21
   4.3. BUTLER INFORMATION MODEL ................................... 24

       4.3.1. Introduction .................................................. 24
       4.3.2. The foundation: The IoT Domain Model from IoT-A ....... 26
       4.3.3. The context complement: Fi-WARE Data/Context Management .... 32

   4.4. KEY CONCEPTS AND INTERACTIONS ............................. 35

       4.4.1. BUTLER Deployment Models ................................. 35
       4.4.2. BUTLER Service and Resource Model ..................... 37
       4.4.3. BUTLER Security Services ................................ 48
       4.4.4. BUTLER SmartObject Addressing .......................... 56
       4.4.5. BUTLER Localization Management ........................ 59
       4.4.6. BUTLER Data Discovery and Data Marketplace ................... 60
       4.4.7. BUTLER Context Management ............................... 80
       4.4.8. BUTLER User Profile Management ......................... 92

   4.5. ARCHITECTURE SPECIFICATION .................................. 96

       4.5.1. Communications Layer ..................................... 96
       4.5.2. Data/Context Management Layer ............................ 114
       4.5.3. Services Layer ............................................... 132
       4.5.4. System/Device Management Layer ......................... 152

   4.6. MATCHING OF BUTLER REQUIREMENTS TO ARCHITECTURE . 161

       4.6.1. SmartHome – Monitoring and controlling home appliances 161
       4.6.2. SmartHome – Multimedia everywhere ........................ 163
       4.6.3. SmartHealth – Personalized diabetes self-healthcare .......... 164
       4.6.4. SmartCity – Smart parking space management .............. 165
       4.6.5. SmartShopping – Context-aware sparkdeals .................. 167
       4.6.6. SmartTransport – Safe transportation of school kids ....... 168

   4.7. REVIEWED USE CASES AND REQUIREMENTS .................... 169

       4.7.1. SmartHome – Monitoring and controlling home appliances 169
       4.7.2. SmartHome – Multimedia everywhere ........................ 170
4.7.3. SmartHealth – Personalized diabetes self-healthcare ................................................................. 171
4.7.4. SmartCity – Smart parking space management ...................................................................... 172
4.7.5. SmartShopping – Context-aware sparkdeals ......................................................................... 173
4.7.6. SmartTransport – Safe transportation of school kids .............................................................. 174

5. INITIAL PERVASIVE BUTLER PROOF-OF-CONCEPT ............................................................. 176
5.1. SMARTHOME SCENARIO (ENERGY FORECASTING) .................................................................. 176
5.2. SMARTHOME SCENARIO (FOLLOW ME) ................................................................................. 178
5.3. SMARTPARKING SCENARIO .................................................................................................... 179
5.4. SMARTHEALTH SCENARIO ...................................................................................................... 182
5.5. SMARTTRANSPORT SCENARIO .............................................................................................. 184
5.6. SMARTSHOPPING SCENARIO .................................................................................................. 185

6. CONCLUSION ............................................................................................................................... 188

7. REFERENCES .............................................................................................................................. 189
List of figures

Figure 1: BUTLER Architectural Layering ................................................................. 20
Figure 2: High-level overview of the BUTLER Architecture ............................... 21
Figure 3: IoT-A Functional Model ........................................................................... 22
Figure 4: Architectural Layering and Functional Components in BUTLER ............ 23
Figure 5: IoT Domain Model .................................................................................. 26
Figure 6: User to Physical Entity relationship ....................................................... 27
Figure 7: User to Physical Entity relationship (with ownership) ......................... 27
Figure 8: Physical to Virtual Entity relationship ................................................... 28
Figure 9: SmartObject to Physical Entity relationship .......................................... 29
Figure 10: Introduction of the Service and Resource concepts and its relationship to SmartObjects and Physical Entities ........................................... 30
Figure 11: Users to Application/Resource relationship ........................................... 31
Figure 12: Modeling of Users, SmartMobiles and SmartObjects as Physical Entities 32
Figure 13: FI-WARE Data Element structure .......................................................... 32
Figure 14: FI-WARE Context Element structure .................................................... 33
Figure 15: Context to Virtual Entity relationship ................................................... 33
Figure 16: Context to Resource relationship ............................................................ 34
Figure 17: Application to Context relationship ....................................................... 35
Figure 18: Examples of BUTLER Deployment models ............................................ 36
Figure 19: BUTLER Overall Architecture ............................................................ 38
Figure 20: Relationship between the BUTLER Information Model and the BUTLER Service and Resource Model ................................................................. 39
Figure 21: Service and Resource Model ................................................................ 40
Figure 22: Service and Resource .......................................................................... 40
Figure 23: Service and Resource – Player Service .................................................. 41
Figure 24: Service and Resource – Motion Detection Service ............................... 41
Figure 25: Service and Resource – Location Service ............................................. 41
Figure 26: BUTLER Resource .............................................................................. 42
Figure 27: Attribute and Metadata ......................................................................... 42
Figure 28: Attribute and Metadata – Player Service .............................................. 43
Figure 29: Attribute and Metadata – Motion Detection Service ............................ 43
Figure 30: Attribute and Metadata – Temperature Service ..................................... 43
Figure 31: Access Method ...................................................................................... 44
Figure 32: Access Method – Player Service ............................................................. 45
Figure 33: Access Method – Motion Detection Service .......................................... 45
Figure 34: Resource Discovery .............................................................................. 45
Figure 35: Service/Resource local discovery and use ............................................. 46
Figure 36: Service/Resource remote discovery and use ......................................... 47
Figure 37: Service implementations ...................................................................... 48
Figure 38: Resource Listeners .............................................................................. 48
Figure 39: High-level Interactions between Security Roles .................................... 49
Figure 40: Retrieval of Access-token sequence diagram ...................................... 51
Figure 41: Registration of key material (factory-originated) ................................... 53
Figure 42: Registration of key material (set by the owner) ..................................... 54
Figure 43: End to end security sequence diagram ................................................ 55
Figure 44: Example of BUTLER deployment ....................................................... 57
Figure 45: Localization Manager Interaction .......................................................... 60
Figure 46: Basic concepts associated to the Data Discovery and Marketplace ....... 63
Figure 47: Example of the BUTLER interface for the discovery of Entities and Data Sources based on a map tool (the Entity is here depicted as a polygonal shape; Data Sources as map pins) .................................................. 64
Figure 48: Data Discovery and Marketplace High-level Architecture ...................................................... 66
Figure 49: Virtual Entity Management architecture .................................................................................. 67
Figure 50: Virtual Entity Management sequence diagram ....................................................................... 68
Figure 51: Data Offering Creation architecture ....................................................................................... 69
Figure 52: Data Offering Creation sequence diagram .............................................................................. 70
Figure 53: Search and Purchase architecture ......................................................................................... 71
Figure 54: Search and Purchase sequence diagram .................................................................................. 72
Figure 55: Definition of Business Events architecture ........................................................................... 74
Figure 56: Definition of Business Events sequence diagrams ................................................................... 75
Figure 57: Reception of SmartObject Data architecture ........................................................................... 77
Figure 58: Reception of SmartObject Data sequence diagram ................................................................... 78
Figure 59: Access to cached SmartObject data architecture ..................................................................... 79
Figure 60: Access to cached SmartObject data sequence diagram ........................................................... 80
Figure 61: BUTLER Context Model within the BUTLER Information Model ........................................... 82
Figure 62: FI-WARE Data/Context Architecture Overview ....................................................................... 83
Figure 63: High-level Context Management Architecture ....................................................................... 86
Figure 64: Context Declaration ................................................................................................................ 87
Figure 65: Context Declaration sequence diagram .................................................................................. 88
Figure 66: Context Information Distribution ............................................................................................. 90
Figure 67: Context Consumption sequence diagram ................................................................................ 91
Figure 68: User Profile Manager high level architecture .......................................................................... 92
Figure 69: Register Account message sequence chart .............................................................................. 93
Figure 70: Get Account message sequence chart .................................................................................... 94
Figure 71: Update Account message sequence chart ............................................................................... 95
Figure 72: BUTLER Overall Architecture – Communications Layer .......................................................... 96
Figure 73: Active Monitoring - ICMP ping command .............................................................................. 98
Figure 74: SNMP vision ............................................................................................................................. 99
Figure 75: Device Discovery ..................................................................................................................... 103
Figure 76: Device Directory requests and results ...................................................................................... 104
Figure 77: IoT Protocol Adapter interfaces and internal .......................................................................... 107
Figure 78: BUTLER Overall Architecture – Data/Context Management Layer ......................................... 114
Figure 79: Interaction between the user behavior capturing module and other functional component in the context management layer ........................................................................................................ 120
Figure 80: A distributed architecture for behavior awareness ................................................................... 120
Figure 81: Step counting with complex event processing ...................................................................... 121
Figure 82: Sequence diagram of event-based motion activity recognition .............................................. 122
Figure 83: HARD-BN: A Heterarchical, Autonomic, Recursive and Distributed Bayesian Network framework for flexible and scalable activity recognition ................................................................. 124
Figure 84: Localization Manager interactions ......................................................................................... 128
Figure 85: Device data collection and actuation protocol ........................................................................ 131
Figure 86: BUTLER Overall Architecture – Services Layer .................................................................. 133
Figure 87: Butler Multimedia Services ................................................................................................... 134
Figure 88: Service Registration ............................................................................................................... 140
Figure 89: Service Discovery .................................................................................................................... 141
Figure 90: Service composition principle .................................................................................................. 142
Figure 91: BUTLER Overall Architecture – System/ Device Management Layer .................................... 152
Figure 92: SmartObject Management Portal ........................................................................................... 153
Figure 93: SmartObject Management Portal – Service View ................................................................... 154
Figure 94: SmartObject Management Portal – Service Bundles View ................................................... 154
Figure 95: Software Update model ......................................................................................................... 156
Figure 96: Protocol for transferring firmware images .............................................................................157
Figure 97: PoC Cluster SmartHome ......................................................................................................162
Figure 98: SmartHome Multimedia everywhere ....................................................................................163
Figure 99: SmartHealth personalized diabetes self-care .........................................................................165
Figure 100: SmartParking space management .......................................................................................166
Figure 101: Users and Providers of data in Smart Shopping Proof of Concept .........................................167
Figure 102: SmartHome Storyline .........................................................................................................176
Figure 103: Detailed view energy awareness and forecasting in SmartHome Scenario ..............................177
Figure 104: Use case example: Multimedia Follow me application .........................................................178
Figure 105: Diagram of components and services ..................................................................................179
Figure 106: Traditional Parking vs. SmartParking Scenarios ..................................................................180
Figure 107: SmartParking High-Level Architecture ..............................................................................181
Figure 108: SmartParking Message Sequence Chart .............................................................................182
Figure 109: Diabetes Companion: a HTML5 enabled personalized diabetes self-care application ............183
Figure 110: Activity recognition in the SmartHealth scenario ................................................................184
Figure 111: SmartTransport updated architecture ................................................................................185
Figure 112: SmartShopping application interface ..................................................................................186
Figure 113: Relationship among entities in the SmartShopping field trial ...............................................187
List of tables

Table 1: BUTLER Resource Types ................................................................. 42
Table 2: Resource Access Method Types ...................................................... 44
Table 3: BUTLER Security Roles ................................................................. 49
Table 4: Application Credentials ................................................................. 51
Table 5: User Credentials ........................................................................ 52
Table 6: Permission Table example ............................................................ 52
Table 7: Example of Addressing and Resources tables in the SmartObject Gateway ........................................................................ 58
Table 8: Example of Addressing and Resource Description tables in the Local SmartServer .............................................................. 58
Table 9: Example of addressing table in the SmartServer ............................. 59
Table 10: Network Monitoring supported operations .................................... 101
Table 11: Device Management operations ................................................... 101
Table 12: Device Discovery operations ....................................................... 103
Table 13: Device Directory operations ....................................................... 104
Table 14: Device Authentication supported operations ............................... 105
Table 15: Authorization Server supported operations .................................. 106
Table 16: Methods Description ................................................................ 108
Table 17: User Device Directory supported operations .................................. 110
Table 18: User Authentication supported operations .................................... 111
Table 19: User Connectivity Manager supported operations ....................... 111
Table 20: Server Directory supported operations ........................................ 112
Table 21: Server Authentication supported operations ............................... 113
Table 22: Server Connectivity Manager supported operations ..................... 114
Table 23: Complex Event Processing engine supported operations .............. 118
Table 24: Persistent Storage supported operations ...................................... 119
Table 25: User Behavior Capture supported operations ............................... 125
Table 26: Generic Notification/Actuation supported operations ................. 125
Table 27 Resource Access supported operations ........................................ 126
Table 28: Devices Example ..................................................................... 126
Table 29: Simple Event Processing supported operations ......................... 127
Table 30: Localization Manager supported operations ............................... 129
Table 31: User Profile Manager supported operations ............................... 130
Table 32: Device Data Collection and Actuation supported operations .......... 131
Table 33: Context / Behavior Information Provider supported operations ...... 132
Table 34: Multimedia Messaging Service supported operations .................. 135
Table 35: Multimedia DVB Services supported operations ......................... 136
Table 36: Multimedia Player Service supported operations ....................... 137
Table 37: Energy Service supported operations ........................................... 139
Table 38: Application Repository supported operations .............................. 140
Table 39: Service Registration supported operations .................................... 141
Table 40: Service Discovery supported operations ....................................... 141
Table 41: Service Composition supported operations .................................... 143
Table 42: Service Re-binding supported operations ..................................... 144
Table 43: Data Discovery supported operations .......................................... 145
Table 44: Resource Exposition supported operations ................................... 147
Table 45: Context Exposition supported operations ..................................... 148
Table 46: Behavior Exposition supported operations .................................... 149
Table 47: User Profile Data Model ............................................................... 150
Table 48: Person data object attributes ...................................................... 150
Table 49: User Profile Exposition supported operations .............................. 151
Table 50: Localization Exposition supported operations ..............................................................151
Table 51: SmartObject Management Portal supported operations.............................................154
Table 52: Service Monitoring supported operations.................................................................155
Table 53: Deployment supported operations.............................................................................156
Table 54: Software Update supported operations.....................................................................158
Table 55: Data Management and Marketplace Portal supported operations .........................160
Table 56: Context Management Portal supported operations....................................................161
# 1. ACRONYMS AND DEFINITIONS

<table>
<thead>
<tr>
<th>Acronyms</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IERC</td>
<td>Internet of Things European Research Cluster</td>
</tr>
<tr>
<td>FI-PPP</td>
<td>Future Internet - Private Public Partnership (EU project)</td>
</tr>
<tr>
<td>FI-WARE</td>
<td>Future Internet - Core platform</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indication</td>
</tr>
<tr>
<td>CEP</td>
<td>Complex Event Management</td>
</tr>
<tr>
<td>ARM</td>
<td>Architecture Reference Model</td>
</tr>
<tr>
<td>NGSI</td>
<td>Next Generation Services Interfaces</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>PE</td>
<td>Physical Entity</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technologies</td>
</tr>
<tr>
<td>ADSL</td>
<td>Asymmetric Digital Subscriber Line</td>
</tr>
<tr>
<td>KNX</td>
<td>KNX (EN 50090, ISO/IEC 14543), OSI-based network communications protocol for intelligent buildings.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BSOS</td>
<td>BUTLER Smart Object Services</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>REST</td>
<td>REpresentational State Transfer</td>
</tr>
<tr>
<td>JSON-RPC</td>
<td>Java Script Object Notation - Remote Protocol Call</td>
</tr>
<tr>
<td>CoAP</td>
<td>Constrained Application Protocol</td>
</tr>
<tr>
<td>PIR</td>
<td>Passive Infrared Sensor</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>URL</td>
<td>uniform resource locator</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>AS</td>
<td>Authorization Server (BUTLER)</td>
</tr>
<tr>
<td>HSM</td>
<td>Hardware Security Module</td>
</tr>
<tr>
<td>Oauth 2.0</td>
<td>open standard for authorization (protocol version 2.0)</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identifier</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>NFC</td>
<td>Near Field Communication</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positionning System</td>
</tr>
<tr>
<td>GE</td>
<td>Generic Enabler (from FI-WARE)</td>
</tr>
<tr>
<td>NM</td>
<td>Network Monitoring</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time (network latency)</td>
</tr>
<tr>
<td>PA FG</td>
<td>Protocol Adapter Functional Group</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
<tr>
<td>GW</td>
<td>Gateway</td>
</tr>
<tr>
<td>------</td>
<td>--------------</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>SMA</td>
<td>Signal Magnitude Area</td>
</tr>
<tr>
<td>HARD-BN</td>
<td>Heterarchichal Autonomic Recursive and Distributed Bayesian Network</td>
</tr>
<tr>
<td>MMS</td>
<td>Multimedia Service (BUTLER)</td>
</tr>
<tr>
<td>DVB</td>
<td>Digital Video Broadcasting</td>
</tr>
<tr>
<td>GPG</td>
<td>Gnu Privacy Guard</td>
</tr>
<tr>
<td>OTAU</td>
<td>Overt The Air Update</td>
</tr>
<tr>
<td>UAP</td>
<td>Upgrade Access Point</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphic User Interface</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>ADL</td>
<td>Activities of Daily Living</td>
</tr>
</tbody>
</table>
2. INTRODUCTION

2.1. PURPOSE OF THE DOCUMENT

The purpose of this document is to present the consolidated architecture of the BUTLER system as the result of the project work on its second year. It also provides a high-level description of the initial pervasive BUTLER proof-of-concept. This deliverable extends and updates the information provided in Deliverable D3.1 – Architectures of BUTLER Platforms and Initial Proofs of Concept, providing a much more elaborated architecture proposal.

2.2. INTENDED AUDIENCE

The report is firstly a working document for the project partners as it reflects the expertise of the project members, their assets and their wills and defines therefore path for future project activities. The document also targets members of other IoT related projects and more especially the one from the IERC and the FI-PPP clusters. The purpose is to inform these projects about BUTLER intentions so to encourage other projects to react to provide advices based on past work (this apply in particular for projects such as IoT-A or FI-WARE) or identify possible synergies (this apply for recent projects such as iCore).

Finally, the report and more especially its comprehensive state of the art section should provide a sound reference to any technical stakeholder wishing to increase it IoT related knowledge.

2.3. DOCUMENT STRUCTURE

According to its title and purpose, this report comprises two main sections: Section 4 is devoted to the architecture the BUTLER project has devised in order to meet its requirements and the scenarios supported by its proofs of concept; Section 5 focuses on the description of the BUTLER initial pervasive proof-of-concepts, describing each of the scenarios supported in said proof-of-concept.

The structure of Section 4 (BUTLER Architecture) is as follows:

- Section 4.1. refreshes BUTLER architectural layout, describing its four layers: Communications, Data/Context Management, Services and System/Device Management.
- Section 4.2. provides a high-level overview of BUTLER Architecture, introducing its more relevant functional components and the way they are mapped to the BUTLER architectural layering.
- Section 4.3. introduces the BUTLER Information Model. It describes the main conceptual entities BUTLER handles. It is heavily influenced by the IoT-A Reference Model and contains extensions, inspired by FI-WARE, to explicitly handle context.
- Section 4.4. shows the most relevant interactions and concepts provided by BUTLER. The purpose of this section is to teach any reader how the BUTLER architecture works in practice and how different functional components co-operate with each other to provide the described functionality. The following key concepts have been identified:
  - BUTLER Deployment Model: It describes how the different functional components described in the BUTLER Architecture can be deployed on entities playing the different roles BUTLER supports (SmartObject, SmartServer and SmartMobile)
  - BUTLER Service and Resource Model: This model offers a way to represent the capabilities provided by a SmartObject in terms of Resources and Services. SmartObject Services are exposed by SmartObjects and group a number of Resources.
  - BUTLER Security Services: This section described how security is implemented within BUTLER, by introducing an Authorization Server that handles the secure dialogue between Resource Consumers and Resource Providers.
  - BUTLER SmartObject Addressing: A comprehensive description on the mechanisms to address SmartObjects is provided.
• BUTLER Localization Management: This section shows how the location of SmartObjects is handled.
• BUTLER Data Discovery and Data Marketplace: This section introduces the mechanisms for Device Owners and Service Developers to define and discover data sources and virtual entities. It also describes how Device Owner creates Data Offerings (from SmartObjects) that can be purchased by Service Developers for its integration in BUTLER applications.
• BUTLER Context Management: This section describes the general-purpose context management BUTLER has implemented. It relies on Complex Event Processing.
• BUTLER User Profile Management: A description of how the BUTLER user’s profile is managed.

Section 4.4.8 specifies each of the functional components the BUTLER architecture is made of. It is structured according to the BUTLER architectural layering and follows a common schema (description, interaction with other functional components, and main operations provided).

Finally, Sections 4.6. and 4.7. show the dependencies between BUTLER requirements and scenarios and the functional components of the BUTLER architecture.

2.4. SUGGESTED PREVIOUS READINGS

This document can be read without any prior knowledge of BUTLER. However, to make the user get the most of it, the following BUTLER deliverables and external information should be read first:

• Deliverable D1.1 – Requirements and Exploitation Strategy [1]. It provides the information about the BUTLER use cases, applicability domains and requirements.
• Deliverable D1.2 – Refined Proof of Concept and Field Trial Specification [2]. It will provide a better understanding of BUTLER planned field trials and proofs of concept.
• Deliverable D3.1 – Architectures of BUTLER Platforms and Initial Proofs of Concept. [4]. It provides an overview of the initial proposals for the architecture of BUTLER.
• Deliverable D2.4 – Selected technologies for BUTLER Platform [3]. It provides information about the technological choices BUTLER has made in order to meet its requirements.
• From IoT-A, Deliverable D1.4 – Converged architectural reference model for the IoT v2.0 [11]. It introduces the IoT Architecture Reference Model (ARM). This ARM has been the primary input for the modeling of the BUTLER Information Model.
• FI-WARE Architecture [15] and specially the FI-WARE Data/Context Management Architecture [16]. BUTLER has taken advantage of the data management approach provided by FI-WARE and its introduction of Complex Event Processing as key enabling technology.
3. EXECUTIVE SUMMARY

The main objective of BUTLER is to support the construction of pervasive applications that make use of heterogeneous devices (SmartObjects in BUTLER parlance), based on different protocols and standards. Said pervasive applications aim to improve daily user activities in different domains taking into account contextual information (user needs, preferences and location, status of the physical entities the user interacts with and so on).

The interworking between the different elements of the BUTLER architecture is, at first sight, really simple: users invoke applications that utilize user data, contextual information, location information and predicted behavior to achieve their purposes. As part of said purposes, applications also request actuators to execute an action. To enable applications implement their functionality, BUTLER takes the raw information generated by users and devices and shuffle it to create rich contextual information, to calculate precise location information, or to predict user behavior. Access to devices in order to actuate on them is also provided. This apparently simple approach poses many challenges on BUTLER. To name a few, BUTLER has to be able to gather information from the physical environment through devices supporting disparate communication technologies, application protocols and data models (nothing new for any IoT project). BUTLER has to be also able to provide efficient ways to declare and compute contextual and location information, as well as predict user behavior. BUTLER has to achieve all of this in a secure way so that only authorized parties access the devices that gather information from the physical environment or actuate on it. Other key security services, namely integrity and confidentiality, have also to be guaranteed when communicating with devices. Finally, some business models that BUTLER can support involve the supply of data mediated or inferred by BUTLER to third-parties wishing to use it to create new applications. Additional challenges have to be confronted with when taking into account the different deployment scenarios BUTLER supports. Indeed, the different functional components the BUTLER architecture comprises can be distributed across entities playing the different roles defined in BUTLER: SmartObjects (with the SmartObject Gateway companion to integrate different types of devices), SmartServers and SmartMobiles. It is obvious, for instance, that if a robust and comprehensive context generation functionality is wished, the involvement of one, or several, SmartServers is needed. However, basic or limited context generation processes can be implemented in a SmartObject Gateway without requiring the involvement of a SmartServer.

BUTLER has not intended to reinvent the wheel. On the contrary, it has taken advantage of existing efforts, either within the framework of the research-related EU initiatives, such as IoT-A or FI-WARE; or supported by other industry standards body, such as OMA, OAuth, SAML 2.0 or OSGi. When they provide a valid solution for fulfilling BUTLER requirements, said efforts have been taken as input of the BUTLER architectural choices and even taken “as is”.

The BUTLER Architecture structures according to four layers. The Communications Layer deals with the communication infrastructure and the means that enable the communication between the entities playing the different roles (SmartObject, SmartMobile and SmartServer) defined in BUTLER. The Data / Context Management Layer is somehow the cornerstone of BUTLER as it has to cope with the task of gathering and processing information from data sources (mainly in SmartObjects) to make sense of it and thus computing contextual and precise location information or predict user behavior. The Services Layer is responsible for providing all the necessary means for the construction of pervasive BUTLER applications. Finally, the System / Device Management Layer provides the means for the management and monitoring of SmartObjects and of all the functional components the rest of layers are made of.

IoT-A has had a major influence on the BUTLER Information Model. One of the main findings of the IoT-A Domain Model is the explicit differentiation between the actual resources a device provides and the virtual entities that represent entities from the real world. Such a separation has been inherited by BUTLER, as it
provides a dear and straightforward approach to the management of context. It seems obvious that contextual information is tied to an “entity” and therefore, as the IoT-A Domain Model introduces clearly such an abstraction, BUTLER has decided to follow it. However, the IoT-A Domain Model does not introduce an explicit context abstraction. Therefore, we have considered FI-WARE, as it does provide such an abstraction (inherited from OMA Next Generation Services Interface). That way, we have associated context to virtual entities and provided a set of functional components that enable a general-purpose context management functionality hosted by a SmartServer. Thus, BUTLER enables users to pick up the virtual entities they are interested in and to declare what the context associated to those entities means for them. Next, BUTLER is able to collect data from different data sources, usually associated to SmartObjects, mix data sources in real-time, generate contextual information and deliver to the BUTLER context-aware applications. Thus, BUTLER has focused not only in the delivery of contextual information. On the opposite, we have tried to analyze all the steps of the process, enabling the users of the contextual information to declare what context means for them. Finally, we have not used the IoT-A Functional Model as it is too abstract for the concrete realization of BUTLER (as it is not our purpose to deploy a fully-compliant IoT-A system). We have taken advantage of the discovery concepts it introduces, but have merged all the data discovery capabilities in a single functional component, including also an explicit feature dealing with the management and discovery of virtual entities (something missed in the IoT-A Functional Model and that makes much sense when supporting scenarios where the end-user query the system, find out which available entities exist and declare how the context associated to said entities looks like).

The key enabler of the integration of heterogeneous SmartObjects is the BUTLER SmartObject Gateway. It implements a Service and Resource Model integrated into a Service-Oriented Architecture (SOA) that allows representing different devices in a homogeneous way, allowing accessing their provided data and functionalities and building then BUTLER applications based on resources provided by various devices. Here Services are meaning a group of related resources a SmartObject exposes (not to be mistaken by an end-user application). A number of associated functional entities support this Service and Resource Model by enabling registration and discovery of SmartObjects and the services and resources they host. In this context, OSGi provides an implementation base to take advantage of SOA, particularly, at the gateway side which needs to manage multiple devices and different protocols.

Security Services are architected around an Authorization Server. Its main task is to authorize a consumer to access a resource (what is exposed by a SmartObject). To do so, the Authorization Server registers resources and delivers necessary access tokens and security material to secure the exchange of messages. OAuth 2.0 is used for this dialogue, but specific formats for the access tokens and procedures for the derivation of security material are introduced. The Authorization Server implements a basic authentication protocol based on login/password paradigm. In case stronger authentication is desired BUTLER architecture allows the introduction of an Authentication Server that supports SAML v2.0.

Localization Management is a key component of any generic IoT-based infrastructure as it enables various types of context-aware applications. The BUTLER solution lies on a Localization Manager that continuously receives raw ranging data (e.g. RSSI, ToA, AoA, etc.) from SmartObjects and in real-time processes these data to estimate the position of unknown SmartObjects by using a specific localization algorithm implemented in a centralized manner in the SmartServer. The Localization Manager uses the common functional components to access information from the SmartObjects and additional functional components to expose the precise localization information it generates.

It has been mentioned previously that the FI-WARE abstractions has been taken as inputs for modeling the context. However, it has not been only helpful with regard to modeling. Another decisive influence of FI-WARE and their sections devoted to data management is Complex Event Management (CEP). FI-WARE introduces a CEP Generic Enabler but although it describes the enabler interfaces, it does not provide too detailed information about its usage in real scenarios. BUTLER introduces a Complex Event Processing engine as a functional component supporting scenarios where data coming from SmartObjects is
distributed towards BUTLER applications enabling a rich set of data offerings that is made available to application developers. It is also used to build other functional components, notably those focused on generic generation of contextual information, user behavior prediction, and management of simple events.

To sum up, BUTLER provides a rich set of functional components supporting BUTLER requirements and use cases. Although the set of functionalities is really large, a number of key functionalities have been identified and described in this document. Apart from the aforementioned functionalities (BUTLER Information Model, BUTLER Service and Resource Model, Deployment Models, Security Services, Localization Management, and Context Management), the following ones will be also analyzed: User Profile Management, SmartObject Addressing, and Data Discovery and Data Marketplace).
4. BUTLER ARCHITECTURE

4.1. BUTLER ARCHITECTURAL LAYERING

BUTLER was not conceived with a pre-established architecture in mind. Instead, it defined four architectural layers trying to cover all the functionalities needed to fulfill the requirements on a context-aware IoT architecture targeting several vertical domains. At the same time, BUTLER does not aim to reinvent the wheel and, once identified the architectural components it will require to implement the functionalities described in use cases and user stories by means of already defined architectures. Thus, those of FIWARE or IoT-A will be taken as basis for the definition of the BUTLER functional components. Other frameworks or standardization efforts will be used whenever necessary.

At the same time, it is needed to acknowledge that BUTLER systems are deployed into three different entities: BUTLER SmartObject, BUTLER SmartServer and BUTLER SmartMobile (modeling the devices and gateways, the server and the clients, respectively; see section 4.4.1).

The four architectural layers considered in BUTLER are the following ones:

- **Communications Layer**: It handles the end-to-end holistic communication infrastructure –based on standards as much as possible– that will enable connecting SmartObjects (e.g., sensors, actuators), client devices –SmartMobiles– (e.g., smart phones, smart terminals) and service execution platforms (e.g., gateways –SmartObject Gateways–, application servers –SmartServers–).
- **Data/Context Management Layer**: It will specify data models, interfaces and procedures for data collection from context data sources (e.g., SmartObjects, information systems) and for processing that data in order to provide context information.
- **System/Device Management Layer**: It comprises the necessary components for management and maintenance of SmartObjects, Services and any other system-wide entity, such as configuration, software management, performance management, or diagnostics.
- **Services Layer**: It will specify necessary models, components and interfaces for description, discovery, binding, deployment and provisioning of context-aware services.

![Figure 1: BUTLER Architectural Layering](image)
The interworking between the different layers can be seen in the figure below: users invoke applications that use the Services Layer to access user data, contextual and location information, and user predicted behavior to achieve their purposes. As part of said purposes, applications also request actuators to execute actions. The Data / Context Management Layer is responsible for providing said information to the Services Layer, by taking the raw information generated by users and devices and processing. Access to devices in order to actuate on them is also provided. All this raw information is raised through the Communications Layer. Finally, the System / Device Management Layer interworks with the rest of layers to handle the management and monitoring of SmartObjects and of all the functional components the rest of layers are made of.

4.2. HIGH-LEVEL ARCHITECTURE

BUTLER has not intended to develop a totally new architecture. Instead it has tried to build on existing standards and industry initiatives and therefore to focus on some close EU initiatives, such as IoT-A or FI-WARE.

IoT-A has had a major influence on BUTLER through its Architectural Reference Model (ARM). It is made of several submodels. The foundation of them is the Domain Model, which introduces the main abstractions of the ARM. One of the main findings of the IoT-A Domain Model is the explicit differentiation between the actual resources a device provides and the entities in the real world. Said entities are represented by means of so-called virtual entities. Such a separation has been inherited by BUTLER, as it provides a clear and straightforward approach to the management of context (see section 4.3. for a discussion on how the IoT-A Domain Model is adapted into the BUTLER Information Model). From an architectural point of view the
ARM includes a Functional Model that does not constitute an architectural proposal in itself, but a framework for the modeling of IoT architectures. The IoT-A Functional Model introduces two main entities: the IoT Service Resolution and the Virtual Entity Resolution. The former is a discovery service registering IoT Services (what is exposed by devices). The latter is another discovery service that registers the relationships between the entity attributes and the services that feed such attributes. No explicit entity discovery service is introduced, as the discovery and lookup operations provided by the Virtual Entity Resolution always need to include the type of attribute the user is looking for.

![IoT-A Functional Model Diagram](image_url)

**Figure 3: IoT-A Functional Model**

BUTLER has introduced a simplified discovery infrastructure, the Data Discovery functional component that allows users to look for data sources (from SmartObjects) and also for virtual entities, by means of a geolocalized interface. The main rationale behind this decision is the need to explicitly handle context. The IoT-A Domain Model does not introduce an explicit context abstraction (although it is possible to argue that the attributes associated to a virtual entity are actually the context attributes). In this point, FI-WARE is introduced as it does provide a context abstraction. Following what OMA Next Generation Services Interface (NGSI) standardized, we have associated context to virtual entities and provided a set of functional components that enable general-purpose context management functionalities. Thus, BUTLER enables users to pick up the virtual entities they are interested in and to declare what the context associated to those entities means for them. Next, BUTLER is able to collect data from different data sources, usually associated to SmartObjects, mix data sources in real-time, generate contextual information and deliver to the BUTLER context-aware applications.

BUTLER implements also a Service and Resource Model that allows representing different devices in a homogeneous way, allowing accessing their provided data and functionalities and building then BUTLER Applications based on resources provided by various devices. In BUTLER, Services are meaning a group of related resources a given SmartObject exposes (not to be mistaken by an end-user application). A number of associated functional entities support this Service and Resource Model by enabling registration and discovery of SmartObjects and the services and resources they host.
The Communications Layer deals with the communication infrastructure and the means that enable the communication between the entities playing the different roles defined in BUTLER (SmartObject, SmartMobile and SmartServer). It also implements BUTLER Security Services. The Data / Context Management Layer supports the task of gathering and processing information from data sources (mainly in SmartObjects) to make sense of it and thus computing contextual and precise location information or predict user behavior. The Services Layer is responsible for providing all the necessary means for the construction of pervasive BUTLER applications. Finally, the System / Device Management Layer supports the management and monitoring of SmartObjects and of all the functional components the rest of layers are made of. End-user applications do not belong to the BUTLER Architecture, but play the role of clients of the BUTLER functionality. They can be deployed in SmartMobiles or in application servers providing a classic web front-end to end-users.

The Communications Layer is responsible for enabling the secure connection and the interoperability of the SmartObjects, SmartMobiles and SmartServers (based on IP connectivity). Physical objects use heterogeneous standards, protocols and. These objects shall be able to communicate between each other and with other systems. To this purpose, an IoT Protocol Adapter functional group ensures the interoperability across different communication technologies, hiding the different protocols and standards used by individual objects. The Communications Layer also provides the capabilities for discovering BUTLER SmartObjects. A Device Directory functional component is responsible for storing data and information

---

1 Regular functional components are shown in orange. Functional components belonging to both the layer where they have been placed and to the System / Device Management Layer are shown in purple.
about the SmartObjects. Since the communication of SmartObjects is usually not reliable, a Network Monitoring functional component is used to continuously check the current status of the network and evaluate the related performances. Moreover, the Communications Layer provides a functional component to manage the connectivity of users’ SmartObjects (which can use different technologies such as 3GPP, Wi-Fi, Ethernet) to the remote or local BUTLER servers. Similar to BUTLER SmartObjects, some functionalities for the maintenance of directories of BUTLER user SmartObjects and BUTLER SmartServers are provided. For security purposes, services for authentication and authorization are also. The authentication and authorization functionalities ensure that only identified and authorized SmartObjects (and SmartObjects) can join the BUTLER network in order to ensure a secure access control and that access to the resources provided by SmartObjects is secured as well.

The Data / Context Management Layer is responsible for all data- or context-related functionalities, including collection and capturing of different types of data, persistent storage, processing of data as events, either simple or complex, and the definition of context associated to entities. This layer specification includes several functional components and the required data models used by them: context, which represents highly dynamic data, or more static that can be profile/user or device related. A particular type of user data is about the behavior of the user that can be inferred by sensor and contextual information. Both Simple and Complex Event Processing functionalities are provided. Location and User Profile data are also part of this layer and two specific functional components are defined here in order to manage these data. By considering the complexity of user events, context and profile data (like location and movements, preferences, actions, etc.) a specific functional model for Behavior Capturing can produce higher level behavior understanding of the user, in terms of current or expected activities and situations. A generic functionality is represented by Persistent Storage which gives the possibility to applications to store data. Contextual information related to entities is provided by a Context / Behavior Information Provider, which relies on Complex Event Processing to achieve its purposes. Specific functional entities are also modeled to enable the management and acquisition of data sources by users: Data Management and Marketplace Portal, which enables services to discover virtual entities, and Context Management Portal, to manage the context associated to each of said entities.

The Services Layer is responsible for the description, discovery, binding, deployment, and provisioning of BUTLER services. It is also responsible for making said services available to the applications created by service developers. The Services Layer provides also the generic enabler for Data Discovery, which supports the functionalities associated to the discovery and purchase of data sources and to the discovery of entities representing physical entities and the association of context declarations to each of said entities. Finally, an Application Repository and a number of built-in services and applications are also provided as part of the Services Layer.

Finally, the System / Device Management Layer provides the means for the monitoring and management of SmartObjects, Services, Data Sources, Context definitions and Entities. The functional components it is made of play also a role in the rest of architectural layers, as seen in Figure 4.

4.3. BUTLER INFORMATION MODEL

4.3.1. Introduction

The distinct feature of BUTLER is being context awareness. That characteristic poses a concrete challenge when addressing the creation of context-aware applications: there does not seem to be a widely agreed definition of what context awareness really is. In BUTLER, context awareness builds around location awareness, but goes further as BUTLER aims to provide context awareness within an IoT environment. Therefore, context awareness deals with decision-making based on information gathered by sensors. Location of any kind of entities can be inferred from the information feed generated by sensors, but context can comprise or be inferred from additional information also taken from sensors. Such information
can change over the time and, as a response to such changes, the system will make different decisions. Therefore, to achieve said awareness, it is necessary to provide an information model that seamlessly integrates context within the concepts handled by BUTLER as an IoT-based project.

Two main inputs have been taken into account in order to create the BUTLER Information Model. The first one is IoT-A.\(^2\) Its IoT Domain Model \([11]\), the foundation of the IoT Reference Model, provides a set of basic definitions that can be applied to the modeling of the BUTLER concepts and entities. However, IoT-A do not handles the concept of “context” in itself, at least in an explicit way. Therefore, a second and complementary input will be considered: FI-WARE,\(^3\) as it explicitly provides a Data/Context Management section \([16]\) dealing with context management, which models context in a very basic way.

The approach taken for the creation of the BUTLER Information Model is as follows: the IoT-A Domain Model has been considered as the foundation of the model, however simplified when necessary.\(^4\) It has been complemented with the FI-WARE Data/Context Management information model, so that context is associated to the entities proposed by the IoT-A Domain Model. Throughout this section, UML has been used to graphically illustrate said BUTLER Information Model\(^5\). For the sake of completeness, the IoT-A Domain Model is included in the figure below:

---

\(^2\) IoT-A (\[http://www.iot-a.eu/\]) is a European FP7 Research Project addressing the Internet-of-Things Architecture. IoT-A proposes the creation of an architectural reference model together with the definition of an initial set of key building blocks.

\(^3\) FI-WARE (\[http://www.fi-ware.eu/\]) is a European FP7 Research Project aiming to foster the emerging Future Internet by creating an open architecture and a reference implementation of a novel service infrastructure, building upon generic and reusable building blocks developed in earlier research projects.

\(^4\) For instance, the Augmented Entity is not considered; Digital Artifacts –and therefore a non-Human User– have been removed as well; Network-Resources have been also removed and therefore the BUTLER model only contemplates On-Device Resources.

\(^5\) Generalization is used to depict an “is-a” relationship. Generalization / specialisation are modelled using a solid line with a large hollow triangle. The lines between the classes denote the generalization relationship. A hollow diamond shape used to connect lines between two classes denotes the aggregation relationship. The diamond shape is on the containing class end of the tree with a single line that connects the contained class to the containing class. In other words, the containing class “has-a” contained class. The use of the aggregation relationship does not preclude the actual implementation of the model from actually being a composition (that is, a “stronger” case of aggregation). Finally, an “open arrow” is used to denote a “one-way” association. Said notation indicates that every object in Class A is associated with zero or more objects in Class B, and that every object in Class B is associated with exactly one object in Class A.

Concepts depicting hardware are shown in blue, software in green, and concepts that fit into either multiple or no categories in maroon.
4.3.2. The foundation: The IoT Domain Model from IoT-A

According to the IoT Domain Model, the basic IoT scenario is that of a generic user that wishes to interact with a so-called Physical Entity (PE) of the physical world. Physical entities can be almost any object or environment. However, unlike the physical world, where interactions happen directly (a person can drive a car, raise a parcel from the ground or take his/her body temperature by touching one’s forehead…), in the digital world interactions with the Physical Entities have to be done in a mediated way, by invoking a service or application that will either provide information about the Physical Entity or actuate on it. Physical Entities in turn can aggregate other physical entities. We can model the relationship between users and physical entities in the way described in the figure below.

---

6 The IoT Domain model uses a similar colour code: hardware elements are shown in blue, software in green, animated beings in yellow, and concepts that fit into either multiple or no categories in brown.
Although the IoT Domain Model does not include them explicitly, there can be additional relationships between Users and Physical Entities. The most obvious is the “ownership” or at least entitlement for the management of the physical entity. The relevance of said relationship relies on the access permissions it creates (that is, the owner of a house will have the “right” to get information about his/her home and to adjust the desired temperature, while a stranger will not, at least until the owner does not give him/her right to).

As the Physical Entities have no actual existence in the digital world, a representation of them is needed. Such a representation is achieved by means of so-called Virtual Entities. Said Virtual Entities have two main features:

- **Represent Physical Entities.** Each Virtual Entity is associated to a single Physical Entity (a Virtual Entity cannot represent more than one Physical Entity; however, it is possible that the same Physical Entity is represented by several Virtual Entities). Each Virtual Entity must have one and only one ID that identifies it univocally. As with Physical Entities, Virtual Entities in turn can aggregate other virtual entities.

- **Enables the monitoring, management and actuation of Physical Entities.** Ideally, Virtual Entities are a synchronized representations of a given set of aspects (or properties) of the Physical Entity. This means that relevant digital parameters representing the characteristics of the Physical Entity are updated upon any change of the former. In the same way, changes that affect the Virtual Entity could manifest themselves in the Physical Entity.
The relation between Virtual and Physical Entity is usually achieved by embedding into, by attaching to, or by simply placing in close vicinity of the Physical Entity (intentionally or by chance) one or more ICT Devices that provide the technological interface for interacting with or gaining information about the Physical Entity. Devices are known, in BUTLER parlance, as SmartObjects. A SmartObject thus enables the interaction between Physical Entities and Virtual Entities.

From the IoT point of view, there are three basic types of SmartObjects:

- **Sensors** provide information about the Physical Entity they monitor.
- **Tags** are used to identify Physical Entities to which they are usually attached to.
- **Actuators** can modify the physical state of a Physical Entity, like changing the state (translate, rotate, stir, inflate, switch on/off...) of Physical Entities or even activating/deactivating functionalities.

Additionally, SmartObjects can be an aggregation of several devices of different types.
To bridge the gap between the Virtual Entities and the SmartObjects, enabling the monitoring and manipulation of Physical Entities from the digital world, Services and Resources are introduced. Services represent SmartObject or other digital entities that provide Resources. Services represent a concrete SmartObject or a logical entity which is not necessarily bound to one SmartObject or to any SmartObject at all. Each BUTLER Service provide/exposes Resources and requires/uses Resources provided by other Services. Resources are the software components that actually provide information (sensor data, property, state variables) about, or enable the actuation on Physical Entities.
However there is no intrinsic relationship between Physical Entities and their digital counterparts, the Virtual Entities on one hand, and the SmartObjects, and the Resources they host. Therefore, the Users are enabled to act on or to know about Physical Entities by associations between Virtual Entities and Resources. For each Virtual Entity there can be associations with different Resources that may provide different functionalities like retrieving information or enabling the execution of actuation tasks on said virtual entities. Resources are grouped in Services that are provided by SmartObjects. The relationship between Services and Resources, the different levels of Services and the BUTLER Service and Resource Model are described in detail in section 4.4.2.

When a User wishes to acquire information about or actuate on a given Physical Entity, it must carry out a discovery process determining which Resources associated to the Virtual Entities representing said Physical Entities enable actuation or data access. Next, the User picks up the Resources that match his/her requirements and invokes them. However, the user does not usually directly invoke Resources but through an Application, which accesses Resources to perform its business logic. Applications are end-user oriented applications that aim at increasing the quality of life of the end-users. Said applications use several Resources in order to add context awareness to the application logic.
Finally, it is necessary to acknowledge that both SmartObjects and Users can be modeled as a Physical Entity. For instance, in certain applications, such as device management, whose main concern is the devices themselves and not the entities or environments that these devices monitor, the SmartObject can be modeled as a Physical Entity, which is represented as a Virtual Entity. The same may happen with SmartMobiles (the client device used by users in the BUTLER parlance).
4.3.3. The context complement: FI-WARE Data/Context Management

The Data/Context Management chapter of the FI-WARE specifications introduces the concepts of Data and Context Elements. A Data Element refers to data whose value is defined as consisting of a sequence of one or more <name, type, value> triplets referred as data element attributes, where the type and value of each attribute is either mapped to a basic data type and a basic data value or mapped to the data type and value of another data element.

Context in FI-WARE is represented through Context Elements. A Context Element extends the concept of Data Element by associating an EntityId and EntityType to it, uniquely identifying the entity to which the Context Element information refers. There may be many different context elements referring to the same entity in a system, each containing the values of a different set of attributes. This allows that different applications handle different context elements for the same entity, each containing only those attributes of that entity relevant to the corresponding application. For the sake of completeness, the modeling of Data and Context Elements is shown in the figures below:

Figure 12: Modeling of Users, SmartMobiles and SmartObjects as Physical Entities

Figure 13: FI-WARE Data Element structure
The association of the Context Element to the entities the system handles fits very well with the information model described up to here (which deeply relies on the IoT Domain Model proposed by IoT-A). BUTLER will propose to associate Contexts (equivalent to FI-WARE Context Elements) to Virtual Entities. Therefore, it will possible to handle the context of the Physical Entities represented by the Virtual Entities the Context is associated to. On the other hand, several Contexts can be associated to a given Virtual Entity (just in the same way as many Context Elements can be associated to a same entity).

On the other hand, the Context Element attributes the FI-WARE specifications describe will be made up from the data read through the resources associated to the Virtual Entities. Thus, the attributes of the Context in BUTLER will be mostly created from data read from Resources. Each attribute will be the result of...
an operation executed over data elements from one or several Resources. Such Resources may be associated or not to the Virtual Entity the Context is associated to.\footnote{For instance, the context associated to a house can include the outdoors temperature. Said temperature value can be exposed through a resource associated to the Weather Service, which in turn has been modeled also as a virtual entity. Although the resource exposing the temperature is not associated to the virtual entity representing the given house, an element of its context relies on said “external” resource.}

Finally, Applications will be entitled to use not only Resources when they need to know about the status of Physical Entities, but also by means of richer Contexts.

Figure 16: Context to Resource relationship
4.4. KEY CONCEPTS AND INTERACTIONS

4.4.1. BUTLER Deployment Models

Functional components defined in the BUTLER architecture can be deployed in different ways. In the BUTLER overall architecture the following roles are identified: SmartObjects and Gateways, SmartMobile and SmartServer. Each of these roles can host one or many different functional components that are defined in the architecture and be deployed in different hardware/software platforms: for example roles can be deployed locally on a device, a home gateway or a local server, or on a remote server.

SmartObjects and SmartObject Gateway can be deployed on the same device hardware: an example for this could be an intelligent security camera (IoT device) which provides the SmartObject role (the IP security camera itself) but also the SmartObject Gateway role by providing access to camera functionality through high level APIs, connectivity to servers and mobile applications through an IP connection, and additional features like security, link to user data and context, etc. SmartObject Gateways can also be deployed for example in existing Internet connectivity gateways (ADSL modem/router) that can provide, beside internet connectivity, also access to home SmartObjects by providing multiple wireless network integration (for example ZigBee and Wi-Fi). A description of how a SmartObject Gateway integrates SmartObjects can be seen in section 4.4.2.2.

Another example to describe different deployment models in BUTLER are SmartServers. A BUTLER SmartServer, for example the Localization Manager, can be deployed locally to an environment, for
example a building in a campus, or a single carriage on a train. In this case applications will need to discover the local server and will be able to communicate directly with the server to perform their operation, by means of a local wireless connectivity, for example. Several environments can have deployed different SmartServers. Another deployment model provides SmartServer deployed remotely: applications should discover the server by means of the SmartServer Directory functionality or by other usual off-band procedures and then communicate to the server using Internet connectivity. The same SmartServer functionality can therefore be deployed both locally and remotely.

SmartServers that are deployed locally can also be connected to the Internet and be able to communicate with other SmartServers local to an environment (if also connected) or with a remote SmartServer that coordinates the functionality of the local server or replicates some data. Another example is the different deployment models to perform Context Management. The Context Management functionality can be offered locally by a SmartServer so that a SmartObject Gateway or other servers can update contextual information about local entities. In addition the local Context Management SmartServer can also be connected to other similar SmartServer that provide the same context aggregation functionality so that contextual information is both available locally and also aggregated on a centralized context repository.

Another deployment option could consist of having a completely distributed system where SmartServers hosting the Context Management functionality are only deployed locally, but are able to communicate to each other, and to discover other BUTLER SmartServers to find context information related to entities/users not in the current administration domain.

SmartMobile is deployed usually on user-carried mobile devices, like a tablet or a smartphone. Integration with other BUTLER entities (SmartObject Gateways and SmartServer) is provided by the SmartMobile BUTLER Framework that provides common functionalities like user authentication and authorization, SmartServer/Gateway discovery or notifications. Deliverable D4.2 [6] describes the implementation details of the BUTLER SmartMobile Framework.

Figure 18: Examples of BUTLER Deployment models
Figure 18 illustrates some of the described deployment examples: an IP camera provides both SmartObject and SmartObject Gateway functionality on same hardware device. The camera can be connected to a local SmartServer that provides user location data which is also duplicated on a remote Localization Manager. A utility meter SmartObject is locally connected to a SmartObject Gateway that delivers data to a remote SmartServer. Other local SmartServers deployed locally in the home, are connected together and can access functionalities provided by other local SmartServers, as well as cooperate with remote SmartServers. Finally, on some BUTLER devices like tablets and smartphones the SmartMobile framework is installed so that they can communicate with local/remote SmartServer or local SmartObject Gateways.

4.4.2. BUTLER Service and Resource Model

4.4.2.1. Introduction

The main objective of BUTLER is to support the construction of pervasive applications that make use of heterogeneous devices, i.e., SmartObjects, based on different protocols and standards. Pervasive applications aim to improve daily user activities in different domains, such as home, transport, healthcare, city, shopping, etc., considering contextual information (user needs and preferences, available services, etc.). For this, pervasive applications gather information from the physical environment and, if needed, adapt the physical and/or logical spaces. The key enabler of the integration of heterogeneous SmartObjects is the BUTLER SmartObject Gateway. It supports a Service and Resource Model integrated into a Service-Oriented Architecture (SOA) [37] that allows representing different devices in a homogeneous way, allowing accessing their provided data and functionalities and building then pervasive applications based on resources provided by various devices and services.

The Service-Oriented Architecture has become a mature and effective architecture for software development requiring high modularity, much interaction and distribution. SOA enables the construction of applications in a decentralized manner based on the notion of services in software sense. In BUTLER, SOA is adopted to mediate interactions between services and to provide opportunities for discovery, selection and composition of services. In this context, OSGi [38] provides a base implementation to take advantage of SOA, particularly, at the gateway side which needs to manage multiple devices and different protocols. In addition, iPOJO [39] is used to simplify OSGi services development by considering the notion of service components. A service component represents functional and management capabilities of services, while separating access to their interfaces.

The next sections describe the global architecture of the BUTLER SmartObject Gateway, as it provides an extensive description of the way the SmartObjects functionalities are “exposed”; and the BUTLER Service and Resource Model, which rules the way said exposition is done. Specific sections for Resource Discovery and the use of the Service and Resource Model are also provided.

4.4.2.2. BUTLER SmartObject Gateway Architecture

The BUTLER SmartObject Gateway integrates different devices and communication technologies in order to provide a homogeneous access to the underlying heterogeneous networks. The figure below illustrates the overall architecture of the BUTLER SmartObject Gateway.
At the Device level, heterogeneous devices form individual networks that can be accessed by either standard-based (CoAP, ZigBee, MQTT, UPnP, KNX, etc.) or rather proprietary protocols (SunSPOT, TST, ZIGPOS nodes, etc.). At the BUTLER SmartObject Gateway level, the BUTLER SmartObject Gateway provides an abstract and homogeneous way of representing and accessing those devices: the so-called BUTLER Device Access API. On the one hand, this API allows integrating different devices into the platform by representing them as BUTLER SmartObjects. On the other hand, it allows building easily integrated horizontal applications from various domains using the existing devices in a homogeneous way. The framework does not enforce devices to implement this API (even if this could be possible) but rather it proposes having device proxies implementing the API at the BUTLER SmartObject Gateway level.

At the BUTLER SmartObject Gateway level, southbound bridges understand and use the protocol implemented on devices. Bridges are responsible for discovering devices and creating their corresponding proxy objects: BUTLER SmartObject Services (BSOS). BSOS implement the BUTLER Device Access API, allowing thus a homogeneous access to the resources provided by devices. Thus, each device joining the BUTLER network will be then detected by the corresponding bridge, which will create the appropriated BSOS object. The BUTLER SmartObject Gateway is based on the OSGi platform [38] that provides a service execution and deployment framework. Thus, BSOS are then represented as OSGi services. The BUTLER SmartObject Gateway provides an abstraction of physical devices allowing higher level applications accessing them without being aware of their technical details.

BUTLER SmartObject Services can be used by applications willing to access data provided by devices or to perform actions on them. However, in general, applications will make use of higher level services that process and aggregate raw device data and provide richer contextual information. These higher level services are called BUTLER Smart Services.

BUTLER Smart Services expose higher level context information in terms of resources. For instance, a localization service can use low level SmartObject Services to access information provided by motion detection devices in order to provide more accurate localization information. Similarly, a parking information service can reuse information from individual parking sensors (exposed through SmartObject Services).
services) in order to provide a city-scale parking service. Smart Services can run on local or remote BUTLER SmartServers. Thus, communications with Smart Services can be local or remote. BUTLER Smart Services are used by BUTLER Applications.

**BUTLER Applications** are end-user oriented applications that aim at increasing the user’s quality of life. Such applications use BUTLER SmartServices in order to add context awareness to the application logic. BUTLER Applications can run on local SmartServers, remote SmartServers or SmartMobile platforms. Thus, communications between applications and services can be local or remote.

In order to access remotely the services available on the SmartObject Gateway, different protocols can be used, such as HTTP REST, JSON-RPC, Web services, MQTT, CoAP, etc. Northbound bridges, specific to these communication protocols, allow accessing remotely the devices available on the SmartObject Gateway.

### 4.4.2.3. Service and Resource Model Description

The BUTLER Service and Resource Model (see Figure 21) derives from the BUTLER Information Model. See its relationship in Figure 20.

![Diagram of BUTLER Resource and Service Model](Image)

**Figure 20:** Relationship between the BUTLER Information Model and the BUTLER Service and Resource Model

The BUTLER Service and Resource Model allows defining SmartObjects and services that expose resources and building then BUTLER Applications by using resources provided by different providers.
In this model, services are classified as SmartObjectServices, which represent services exposed by SmartObjects (i.e., devices), such as a player service exposed by a TV device; and SmartServices which represents logical services (not directly exposed by SmartObjects) such as a weather service or a localization service. BUTLER services provide resources that can be then used by BUTLER Applications and also by other SmartServices (see Figure 22).

The Figure 23 shows an example of SmartObjectService. It is a player service, exposed by a SmartObject (a TV device for example), that provides different types of resources allowing accessing the SmartObject data (e.g., volume, media, state, etc.) and invoking actions on the associated SmartObject (e.g., play, stop, pause, etc.).

---

One of the built-in components provided by BUTLER is a multimedia service that uses player services in order to provide its higher level resources. While a player service provides data and actions of a given device, a multimedia service manages a set of devices via their provided player services (see section 4.5.3.2 for further details).
Another example of SmartObjectService is a motion detection service, exposed for instance by a PIR sensor (Passive Infrared Sensor), that provides a resource representing a sensed motion detection (see Figure 24).

A SmartService providing higher level localization information (e.g., occupancy at home) can use the motion detection service and expose occupancy-related resources as shown in Figure 25.

Resources represent information (properties, state variables, sensed data) or functionalities (actions) exposed by Services (provided by SmartObjects or not). From a purely functional point of view, a resource is characterized by the following elements:

- An identifier (ResourceID).
- A resource type (see Table 1 below).
- A set of attributes.
• A set of access methods (see Table 2 below).

Moreover, resources can include other elements (see section 4.4.3 on security).

Resources are classified in the following types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SensorData</td>
<td>Sensory data provided by a service. This is real-time information provided, for example, by the SmartObject that measures physical quantities.</td>
</tr>
<tr>
<td>Action</td>
<td>Functionality provided by a service. This is mostly an actuation on the physical environment via an actuator SmartObject supporting this functionality (turn on light, open door, etc.) but can also be a request to do a virtual action (play a multimedia spot on a TV, make a parking space reservation, etc.)</td>
</tr>
<tr>
<td>StateVariable</td>
<td>Information representing a SmartObject state variable of the service. This variable is most likely to be modified by an action (turn on light modifies the light state, opening door changes the door state, etc.) but also to intrinsic conditions associated to the working procedure of the service</td>
</tr>
<tr>
<td>Property</td>
<td>Property exposed by a service. This is information which is likely to be static (owner, model, vendor, static location, etc.). In some cases, this property can be allowed to be modified.</td>
</tr>
</tbody>
</table>

Table 1: BUTLER Resource Types

Resources have attributes that represent intrinsic information about resources. An attribute is characterized by a name, a type, a value and a set of metadata.

Figure 27: Attribute and Metadata

The Figure 28 shows attributes and metadata examples for some resources exposed by the player service.
Notice that all data resources (i.e., of type Property, StateVariable and SensorData) have a particular attribute that corresponds to the resource data. In the previous example, the data of the state resource is of type enumeration and its current value is STOPPED. The data of the volume resources is of type Integer and has associated metadata the volume description and unit.

Figure 29 and Figure 30 show respectively attributes and metadata of resources exposed by motion detection and temperature services.
Different actions can be performed over the resources exposed by a service via the following access methods: GET, SET, ACT, SUBSCRIBE, UNSUBSCRIBE. Access methods are characterized by a type and a set of input parameters (see Figure 31).

![Diagram of Access Method](image)

**Figure 31: Access Method**

Access methods are classified in the following types:

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

**Table 2: Resource Access Method Types**

The access methods that can be associated to a resource depend on the resource type, for example, a GET method can only be associated to resources of type Property, StateVariable and SensorData. A SET method can only be associated to StateVariable and modifiable Property resources. An ACT method can only be associated to Action resources. SUBSCRIBE and UNSUBSCRIBE methods can be associated to any resource type. Figure 32 and Figure 33 show respectively access methods and parameters for some of the resources exposed by the player and motion detection services.
4.4.2.3.1. Resource Discovery and Exposition (CEA)

Discovering and using resources exposed by BUTLER Services is a favored approach for avoiding using static service interfaces and then increase interoperability. Therefore, BUTLER Services and their exposed resources are registered into Service/Resource repositories. The BUTLER SmartObject Gateway uses the OSGi service registry as Service/Resource repository, where resources are registered as service properties (see Figure 34). Clients ask the Service/Resource repository for resources fulfilling a set of specified properties (defined by LDAP filters). In response, the Service/Resource repository sends clients the list of service references that expose the requested resources and for which clients have authorizations (processed by the authorization server). Clients can then access/manipulate the resources exposed by their selected service objects.
Registered resources and services into the BUTLER Gateway Repository are available to local clients and accessed using Java method invocations.

Figure 35 shows a declaration of service dependency (using iPOJO annotations [39]) to a BUTLER SmartObjectService that exposes volume, play and stop resources and which providing SmartObject is located at the kitchen. Using the OSGi registry, iPOJO will discover services that satisfy the specified requirements, and inject a corresponding object into the player field when accessed. Resources can then be manipulated using the corresponding access methods as shown in Figure 35.

```java
@Requires(filter = "(&(device.location=kitchen) +
  "(&(&(resources=play)(resources=stop))(resources=volume)))")
private Service player;

// get the volume resource
Data volumeData = player.get("volume");

// change (set) the data value of the volume resource
volumeData = player.set("volume", new Integer(45));

// invoke the play resource
Response playResponse = player.act("play", mediaID);

// subscribe to the invocation of the stop resource
Response stopSubscriptionResponse = player.subscribe("stop", this);
```

Figure 35: Service/Resource local discovery and use

Moreover, resources and services can be exposed for remote discovery and access using different communication protocols, such as HTTP REST, JSON-RPC, etc., and advanced features may also be supported (as semantic-based lookup). Figure 36 shows examples of HTTP Rest requests and responses for discovering and manipulating resources.

<table>
<thead>
<tr>
<th>Getting the available services exposing the given resources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request Address</strong></td>
</tr>
</tbody>
</table>

---

9 iPOJO is an Apache Felix subproject delivering a service component runtime that aims to simplify the development of OSGi applications.
## Getting the data value of the volume resource

**Request Address**
http://butler-gw01.eu/services/PlayerService_TV-ABCDE12345/resources/volume/get

**Request Method**
GET

**Response**
200 OK
```
{
  "volume": "45"
}
```

## Invoking the play resource

**Request Address**
http://butler-gw01.eu/services/PlayerService_TV-ABCDE12345/resources/play/act

**Request Body**
```
{
  "media": "http://localhost:8080/123.mov"
}
```

**Request Method**
POST

**Response**
200 OK
```
{
  "Success": "Playing 123.mov on TV-ABCDE12345"
}
```

---

4.4.2.3.2. **Using the BUTLER Service and Resource Model**

The BUTLER Service and Resource Model is used to build SmartObject representations and services and also to integrate them into the BUTLER SmartObject Gateway. SmartObjects and services must implement the **BUTLER Service** interface by extending respectively the **SmartObjectService** and **SmartService** abstract classes (see Figure 37). The BUTLER Service interface defines methods allowing accessing and manipulating
the service exposed resources. The SmartObjectService and SmartService abstract classes define respectively SmartObject and service specific methods, and ease the development of SmartObjects and services by implementing the common SmartObject and service methods.

Using the listener pattern, resources can notify their subscribed clients about attributes modifications, changes in data values (for data resources), and action invocations (for action resources). Resources send events to their listeners using callback methods: attributeNameModified(), dataModified(), actionInvoked(). Listeners must implement a ResourceListener interface.

4.4.3. BUTLER Security Services

BUTLER Security Services are implemented everywhere in each component.

4.4.3.1. Definitions

<table>
<thead>
<tr>
<th>Role</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>User entity granting access to an abstract resource. Generally, the user refers to a person, but can also be an application. The user shall be authorized to access the resource by the owner of the resource.</td>
</tr>
<tr>
<td>Resource Provider</td>
<td>Entity providing (and optionally updating) a resource. The Resource Provider shall check an access-token to provide/update the resource. Resource Metadata shall be registered in Authorization Server (AS)</td>
</tr>
<tr>
<td>Resource Consumer</td>
<td>Client application getting and consuming or acting on a resource on behalf of a user. Such user must be authorized to access the resource.</td>
</tr>
<tr>
<td>Authorization Server</td>
<td>The Authorization Server is the responsible for user authentication and for authorizing Resource Consumers getting or acting on a resource by issuing a resource-related access-token. Optionally, it may delegate the user authentication to an external Authentication Server.</td>
</tr>
</tbody>
</table>
The Authorization Server is then responsible for implementing access-control management with regard to resources. It also plays the role of resource directory as access to resources is only possible when the Authorization Server issues an access-token.

**Table 3: BUTLER Security Roles**

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authorization Server</strong></td>
<td>This optional role can be used by Authorization Server to rely on an authentication protocol not natively implemented in the Authorization Server. It means that the Authorization Server and Authentication Server shall be able to federate user identities.</td>
</tr>
</tbody>
</table>

### 4.4.3.2. High Level Interactions

The figure below presents the main interactions between the different Security Roles defined in BUTLER.

![Figure 39: High-level Interactions between Security Roles](image)

An overall use case can be represented as follow:

1. A User authenticates towards the Authorization Server
2. The User registers resource metadata to the Authorization Server.
3. Later, an application accesses the Authorization Server to retrieve an access-token for a specific resource always on behalf of an authenticated user – if the user is not already authenticated to the Authorization Server, it shall get authenticated and grant the application to retrieve the access-token.
4. The application securely retrieves the resource by providing the resource access-token. Next, it will provide the related service to end users.

The BUTLER Security Services include:

- Secure Transport of messages between any SmartObject and Authorization Server.
- Retrieval of the Access-token.
- User Authentication to Authorization Server.
- Application Authentication to Authorization Server.
- Resource Registration.
- Resource Authentication to Authorization Server.
- Key Management.
4.4.3.3. Relevant Use Cases

In order to explain how security services work in BUTLER, four use cases will be described:

- **Resource Registration**: This use case describes how a resource is registered by a user (its owner) in the Authorization Server. See section 4.4.3.3.1.
- **Retrieval of (Resource) Access-Token**: This use case describes how a Resource Consumer (an application) retrieves, on behalf of an authenticated user, an access-token for subsequent consumption of said resource from a Resource Provider. See section 4.4.3.3.2.
- **Resource Key Management**: This use case describes different alternatives for the sharing of an encryption key between the Resource Provider and the Authorization Service. See section 4.4.3.3.3.
- **End-to-End Security**: This use case describes the secure consumption of Resources by a Resource Consumer. See section 4.4.3.3.4.

4.4.3.3.1. Resource Registration

Generally speaking, a Resource is an Entity that can be addressed using a URL schema. Within BUTLER Resources follow the Resource Model described in section 4.4.2. With regard to the authorization process and, specifically towards the Authorization Server, the Resource address is described using a URL schema. Each URL will be used by the Application (Resource Consumer) to reference the Resource and perform allowed actions on it.

The resource is described using the following elements:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Resource name.</td>
</tr>
<tr>
<td><strong>Owner</strong></td>
<td>Owner of the Resource.</td>
</tr>
<tr>
<td><strong>Semantic</strong></td>
<td>Semantic description: for instance: temperature,</td>
</tr>
<tr>
<td><strong>Available actions:</strong></td>
<td>The AccessMethods available for a Resource (GET/SET/ACT/SUBSCRIBE/UNSUBSCRIBE). These are the actions a Resource Consumer can perform on the Resource.</td>
</tr>
<tr>
<td><strong>Key Material</strong></td>
<td>The key material required to build the access-token for end-to-end security</td>
</tr>
<tr>
<td><strong>Authentication Credentials</strong></td>
<td>Credential used by the Resource to access security material associated to an access-token.</td>
</tr>
</tbody>
</table>

To register a Resource on the Authorization Server, the user shall first authenticate to the server. Next, s/he shall provide the rest of elements that describe a Resource:

- Setting the Name: User selects a name which does not need to be unique.
- Setting the Owner: The owner is automatically the authenticated user.
- Setting the Semantic: The user selects a semantic in a server predefined list. A predefined list is useful to help the application interacting with the SmartObject hosting the Resource.
- Setting Resource URL: The user shall define an URL accessible from the network s/he wants to access the Resource.
- Setting of the Resource supported actions (GET/SET/ACT/SUBSCRIBE/UNSUBSCRIBE).
- Setting Resource Key Material and Authentication Credentials –see below Resource Key Management.
4.4.3.3.2. (Resource) Access-Token Retrieval

The following schema presents the overall message exchange to retrieve an access-token on behalf of a user:

![Image of sequence diagram]

The application shall first provide its identification; next, user shall authenticate and selects required Resources. Once authenticated, the application is provided with an authorization code to get the access-token. The different operations involved in the use case are described below.

4.4.3.3.2.1. Secure Transport of message between any SmartObject and the Authorization Server

All the messages are transported over TLS session. The TLS session shall perform Server Authentication based on an X509 server certificate. Therefore, all SmartObjects shall be configured with the corresponding Certificate Authority (CA) to check the validity of the server certificate.

4.4.3.3.2.2. Application (Resource Consumer)

Identification/Authentication

The application (Resource Consumer) must be registered at the Authorization Server beforehand. The application is provided out-of-band with client credential:

<table>
<thead>
<tr>
<th>Client application</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Client_identifier</td>
<td>Base64 string.</td>
</tr>
<tr>
<td>Client_secret</td>
<td>Base64 string</td>
</tr>
</tbody>
</table>

Table 4: Application Credentials

In order to interact with the Authorization Server, the application needs to get authenticated using the OAuth mechanism [24]. For the first interaction, the application provides a client-identifier. The
Authorization Server checks the application is registered as client application. To retrieve an access-token for a Resource (second interaction), the application shall authenticate to the Authorization Server by providing the client-secret. The client-identifier and the client-secret are shared between the application and the Authorization Server.\textsuperscript{10}

4.4.3.3.2.3. **User Authentication**

In order to interact with the Authorization Server (AS), the user on behalf of whom the request is done shall first authenticate to the server. The Authorization Server contains a database of user including username, password, and long-unique-identifier. The user is registered by the Authorization Server administrator with a username, a random password and a long-unique-identifier (generated by the server). The user authenticate to the AS by entering a user/password.

The user authentication method may be enhanced by using other authentication schema. Anyway, from BUTLER perspective, the focus for the security point-of-view is not the user authentication but more the authorization process and the security of the machine-to-machine exchange of data. From a purely functional point of view, regardless of the authentication procedure, the AS shall be able to link the authenticated subject is a unique user identifier. User name and user password are just the credentials used with this authentication method.

<table>
<thead>
<tr>
<th>User authentication parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User_name</td>
<td>String</td>
</tr>
<tr>
<td>User_password</td>
<td>String –generated by the server and can be modified by the user</td>
</tr>
<tr>
<td>User_uuid</td>
<td>Long unique string identifier (generated by the server at user registration process).</td>
</tr>
<tr>
<td>User_acquaintances</td>
<td>List of user_uuid of user acquaintance</td>
</tr>
</tbody>
</table>

Table 5: User Credentials

4.4.3.3.2.4. **Management of User Acquaintance and Permissions**

A user acquaintance is another user who is known by the user. For a User-1 to add an acquaintance in his/her (authenticated) user acquaintance list, User-1 shall be provided out-of-band with a user_uuid of the new acquaintance, User-2. Once User-2 is added in User-1 acquaintance list, User-1 can give permission for this new acquaintance to use some User-1 Resources.

Each Resource is owned by a user. The Authorization Server manages a Permission Table:

<table>
<thead>
<tr>
<th>UserID</th>
<th>ResourceID</th>
<th>Allowed Action (scope)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>GET/SET</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>GET</td>
</tr>
</tbody>
</table>

Table 6: Permission Table example

The owner of the resource can perform any available actions on his/her Resources. The owner can allow another user (in its acquaintance list) to perform some actions on its resources. In the above example, User-1 owns Resource-1 and has allowed User-2 to get an Access-Token authorizing to perform the GET action on the Resource-1.

\textsuperscript{10} According to the OAuth 2.0 perspective, the messages shall be transported over TLS session
### 4.4.3.3.3. Resource Key Management

#### 4.4.3.3.3.1. Factory Key schema

The Resource Provider shall embed the secret shared with the Authorization Server. The manufacturing process includes the manufacturing of the SmartObject hardware, the installation of the software and the setting of the secrets. Usually, the setting of the secret is performed at a secure center—so called Personalization Center—which securely embeds a so-called MASTER-KEY—generally located in a Hardware Security Module (HSM).

The Authorization Server and the Personalization Center may share the single MASTER-KEY and a predefined derivation function shared between the Authorization Server and the Resource Provider. In some cases, the shared secret is generated at the Personalization Center and securely provided to the Authorization Server, so that the Authorization Server registers the secret associated to the Resource identifier in a secure database.

At manufacturing time, the SmartObject is initialized with a **Key-encryption-key**:

<table>
<thead>
<tr>
<th>Key-encryption-key</th>
<th>Factory-key-encryption-key = derivation-function1(object-identifier, FACTORY-KEY-ENC-MASTER-KEY)</th>
</tr>
</thead>
</table>

At personalization time, using the Key-encryption-key, the SmartObject is personalized with an **Access-token-encryption-key**:

<table>
<thead>
<tr>
<th>Access-token-encryption-key</th>
<th>Access-token-encryption-key = derivation-function2(object-identifier, AUTHORIZATION-SERVER/PERSO-CENTER-ACCESS-TOKEN-ENC-MASTER-KEY)</th>
</tr>
</thead>
</table>

In case the Authorization Server knows the derivation-function2 and the PERSO-CENTER-ACCESS-TOKEN-ENC-MASTER-KEY, it is able to recompute the SmartObject key. In case the Authorization Server does not have the MASTER-KEY, it shall retrieve the shared key from the abovementioned secure database.

In the factory key management schema, the resource key-material is simply the object identifier. The user shall be able to retrieve the object identifier. In a simple case, the object identifier is written on the SmartObject casing (for instance, it can be the serial-number of the SmartObject).

![Diagram](image)

**Figure 41: Registration of key material (factory-originated)**

The Access-token-encryption-key will be used subsequently for the implementation of the End-to-end Security mechanisms BUTLER provides. When issuing an access-token referring to a given Resource Provider, its payload is encrypted with the Access-token-encryption-key. That way, when a Resource Consumer gets an access-token, and uses it to request a Resource from a Resource Provider, it uses the
Access-token-encryption-key to decrypt the access-token and access the access-token payload information (see section 4.4.3.3.4 for further details).

### 4.4.3.3.2. User-defined shared key schema

The factory key mechanism is not always well suited in all M2M cases as it involves sharing of a secret between the SmartObject (Resource Provider) factory and the Authorization Server. A good choice is to let the user setting the shared key of its SmartObjects. In this case, the Resource Provider shall propose a local interface to setup the Resource Provider key.

![Diagram](image)

**Figure 42: Registration of key material (set by the owner)**

### 4.4.3.3.4. End-to-end Security

The End-to-end Security involves both the Resource Consumer and Resource Provider. It is implemented at application level.

The security of the protocol shall be studied carefully and validated. In particular, all the cryptographic functions and protocol details shall be further defined and reviewed. Here we give only the overall idea of the end-to-end security implemented in BUTLER; the effective specification and implementation will be provided in deliverable D5.1 (Platform Pervasive Functionalities). Please note that some parts of this deliverable may be provided in a –RESTRICTED diffusion– annex in case Gemalto needs to preserve the business interests and exploitation opportunities of some of its industrial partners.

#### 4.4.3.3.4.1. Generation of Access-Token and security material

The access-token and security material are generated by the Authorization Server. The OAuth 2.0 protocol does not specify how to build and checks the access-token. For BUTLER, the access-token shall implement the following requirements.

1. The access-token can used several time until an expiry-date.
2. The access-token refers a specific resource.
3. Receiving the access-token, the Resource Provider shall be able to check that the access-token comes from the authorized application.

**NOTE:** Because the Resource Provider does not encompass any user information, the access-token does not refer user related information.

The Authorization Server randomly generates security material and returns it to the Resource Consumer:

---

11 Available by month 30.
Security material

Authentication-key
Request-encryption-key
Response-encryption-key.

The request-encryption-key and the response-encryption-key will be used to securely transport data between Resource Consumer and Resource Provider. The authentication-key will be used to authenticate the application to the Resource Provider.

The Authorization Server computes the access-token-data as follow:

<table>
<thead>
<tr>
<th>Access-token-data</th>
<th>Access-token-data :=&lt;resource-identifier&gt;</th>
<th>&lt;hash(application-id)&gt;</th>
<th>&lt;authentication-key&gt;</th>
<th>&lt;request-response-key-identifier&gt;</th>
<th>&lt;expiry-date&gt;</th>
<th>&lt;actions&gt;</th>
</tr>
</thead>
</table>

Finally:

<table>
<thead>
<tr>
<th>Access-token</th>
<th>&lt;encrypt(access-token-data, access-token-encryption-key)&gt;</th>
</tr>
</thead>
</table>

4.4.3.3.4.2. End-to-end security – Using Access-Token and security material

The Resource shall be retrieved in a secure way. The protocol implements the following requirements:

- Application data shall be transported securely.
- The Resource Provider shall verify that the access-token is a valid one.
- In case the access-token has been retrieved by a fraudulent application, the fraudulent application shall NOT be able to use it.

![End to end security sequence diagram](image)

Figure 43: End to end security sequence diagram
The message flow is shown below:

<table>
<thead>
<tr>
<th>Message</th>
<th>Implementation</th>
</tr>
</thead>
</table>
| Resource Consumer to Resource Provider request  
<Access-token> | Resource Provider processing  
• Resource Provider decrypts the access-token using the access-token-encryption-key (shared between the Resource Provider and the Authorization Server) to have the access-token-data.  
• Resource Provider retrieves the authentication-key from the access-token-data.  
• Resource Provider retrieves the request-response-key-identifier from the access-token-data.  
• Resource Provider decrypts the request to have the application-data.  
• Resource Provider checks that hash(application-data.application-identifier) == access-token-data.hash(application-id)  
NOTE: The Resource Provider has authenticated the application because only the application knows the application-private-identifier.  
|  
| Resource Provider to Authorization Service request  
:request-response-key-identifier | Resource Provider processing  
On success of the verification process, the Resource Provider authenticates to the Authorization Server to retrieve the request-encryption-key and the response-encryption-key identified by request-response-key-identifier by providing its credentials to get authenticated.  
On reception of the request-encryption-key, the Resource Provider decrypts the request-payload, encrypts the response using the response-encryption-key and returns the encrypted response.  
|  
| Resource Provider to Resource Consumer response  
:request-id | Resource Consumer Processing  
The Resource Consumer decrypts the response using the response-encryption-key. |

### 4.4.4. BUTLER SmartObject Addressing

The main goal of naming and addressing procedures is to uniquely identify SmartObjects in order to send data to the intended recipient. The identifier allocated to a SmartObject is known as its “address” [30]. The address is normally quoted in a numeric format and easily encoded into a binary form for transmission across the network. The identification address needs to be globally unique so that it can be accessed remotely from anywhere.

The IPv6 protocol is able to support a huge number of connected devices expected to be part of the Internet of Things (IoT) thanks to its (theoretical) address space of $2^{128}$ addresses.

Compatibility with legacy Internet is the most relevant constraint for IoT addressing as this feature is not supported (natively) in the current legacy Internet protocols.
Taking into consideration the State-of-the-Art (SoTA) analyzed by the IoT-A project concerning IoT naming and addressing schemes [10][12], the BUTLER project proposes to use gateways for integrating non-IP nodes (e.g. ZigBee devices, RFID tags, etc.) with a BUTLER identifier provided by the responsible SmartObject Gateway. To better understand the choice of an addressing scheme, it is suggested to refer to Figure 44, which shows an example of BUTLER deployment in two different buildings. In particular, each building has its own LocalSmartServer connected with several gateways within the same Local Area Network (LAN) of the building. As it can be observed, each gateway is able to manage different heterogeneous underlying networks. For instance, the smartGW₁ of the first building manages two different SmartObject networks, one based on ZigBee and the other one based on 6LoWPAN. Thus, given this heterogeneity, a common identification scheme, managed by the gateway, is necessary. Note that the user might be within the LAN or outside and accessing the BUTLER network remotely.

![Figure 44: Example of BUTLER deployment](image)

In order to access a specific SmartObject in BUTLER, the SmartObject Gateway has to store information related to the addressing of the underlying technologies. Thus, a specific SmartObject is addressed by using the IP address of the LocalSmartServer, the identifier of the SmartObject Gateway along with the identifier of the specific SmartObject, SmartObjectID.

The tables below show a possible way of storing the addressing information in the SmartObject Gateway, LocalSmartServer and SmartServer, respectively.

Table 7 shows a set of addressing and resources tables that could be implemented in the SmartObject Gateway. In particular, the first table lists all SmartObjects connected to the SmartObject Gateway itself along with corresponding SmartObjectID of the specific underlying communication technology (e.g. ZigBee, NFC, 6LoWPAN, etc.). In the Table 7, this identifier is denoted as SpecificSmartObjId.
Table 7: Example of Addressing and Resources tables in the SmartObject Gateway

<table>
<thead>
<tr>
<th>IdSmartObj</th>
<th>IdComTech</th>
<th>SpecificSmartObjId</th>
<th>Mobile (Y/N)</th>
<th>Coordinates (GPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0xA2C3</td>
<td>1</td>
<td>45.065019,7.658929,210</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0xBA02</td>
<td>0</td>
<td>41.90171,12.478701,205</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2001::CEAB:BA0E:00A1</td>
<td>1</td>
<td>40.858433,2.294424,212</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>049285B9212580</td>
<td>0</td>
<td>48.853287,2.348379,208</td>
</tr>
</tbody>
</table>

Object Available Resources

<table>
<thead>
<tr>
<th>IdSmartObj</th>
<th>IdResource</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Resource Description

<table>
<thead>
<tr>
<th>IdResource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Temperature</td>
</tr>
<tr>
<td>2</td>
<td>Humidity</td>
</tr>
<tr>
<td>3</td>
<td>Pressure</td>
</tr>
<tr>
<td>4</td>
<td>Light</td>
</tr>
<tr>
<td>5</td>
<td>[...]</td>
</tr>
</tbody>
</table>

Communication Technologies

<table>
<thead>
<tr>
<th>IdComTech</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ZigBee</td>
</tr>
<tr>
<td>2</td>
<td>NFC</td>
</tr>
<tr>
<td>3</td>
<td>6LoWPAN</td>
</tr>
<tr>
<td>4</td>
<td>[...]</td>
</tr>
</tbody>
</table>

Table 8 shows a possible implementation of Addressing and Resources tables in the Local SmartServer. The structure is pretty similar to the SmartObject Gateway one, with an additional column indicating the identifier of SmartObject Gateway. The table named “Local Smart GWs” lists the match between the SmartObject Gateway id and the IP network address.

GW Local Addressing

<table>
<thead>
<tr>
<th>IdSmartObj</th>
<th>IdComTech</th>
<th>SmartGW</th>
<th>SpecificSmartObjId</th>
<th>Mobile (Y/N)</th>
<th>Coordinates (GPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0xA2C3</td>
<td>1</td>
<td>45.065019,7.658929,210</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0xBA02</td>
<td>0</td>
<td>41.90171,12.478701,205</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2001::CEAB:BA0E:00A1</td>
<td>1</td>
<td>40.858433,2.294424,212</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>049285B9212580</td>
<td>0</td>
<td>48.853287,2.348379,208</td>
</tr>
</tbody>
</table>

Object Available Resources

<table>
<thead>
<tr>
<th>IdSmartObj</th>
<th>IdResource</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Local Smart GWs

<table>
<thead>
<tr>
<th>SmarthGW</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>192.168.0.1</td>
</tr>
<tr>
<td>2</td>
<td>BABE::CODE</td>
</tr>
<tr>
<td>3</td>
<td>FABE::2251</td>
</tr>
<tr>
<td>4</td>
<td>FE80::CC0S</td>
</tr>
<tr>
<td>5</td>
<td>2001::SAGE</td>
</tr>
</tbody>
</table>

Resource Description

<table>
<thead>
<tr>
<th>IdResource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Temperature</td>
</tr>
<tr>
<td>2</td>
<td>Humidity</td>
</tr>
<tr>
<td>3</td>
<td>Pressure</td>
</tr>
<tr>
<td>4</td>
<td>Light</td>
</tr>
<tr>
<td>5</td>
<td>[...]</td>
</tr>
</tbody>
</table>
At the SmartServer level (see Table 9), the addressing table could be implemented by providing the list of all the BUTLER networks along with network name, corresponding LocalSmartServer IP address, URL of the LocalSmartServer and GPS coordinates.

<table>
<thead>
<tr>
<th>IdButlerNWK</th>
<th>Name</th>
<th>AddressLocalSmartServer</th>
<th>URL LocalSmartServer</th>
<th>CoordinatesPoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>«Butler1NWK»</td>
<td>AAAA::0FCE</td>
<td><a href="http://butler.com/butler1NWK">http://butler.com/butler1NWK</a></td>
<td>45.065019,7.658929</td>
</tr>
<tr>
<td>2</td>
<td>«ISMB»</td>
<td>1568::1568</td>
<td><a href="http://ismb.it/butler">http://ismb.it/butler</a></td>
<td>41.90171,12.478701</td>
</tr>
<tr>
<td>3</td>
<td>«TELECOM»</td>
<td>7313::C006</td>
<td><a href="http://butlertelecom.it">http://butlertelecom.it</a></td>
<td>48.858433,2.294424</td>
</tr>
<tr>
<td>4</td>
<td>«Gemalto»</td>
<td>6E6A::1700</td>
<td><a href="http://gemalto.com/butler">http://gemalto.com/butler</a></td>
<td>48.853287,2.348379</td>
</tr>
</tbody>
</table>

Table 9: Example of addressing table in the SmartServer

4.4.5. BUTLER Localization Management

Localization Management is a key component of a generic IoT-based infrastructure as it enables various context-aware applications. In particular, within BUTLER Localization Management relies on a so-called Localization Manager that supports the following use cases: video follow me at home (SmartHome), getting help outside (SmartHealth), assisted to find a parking space (SmartCity), reactive discounts on popular goods (SmartShopping) etc., see [1] for more details.

A preliminary high-level architecture of the Localization Manager has been introduced in [3], where the main functionalities have been presented. This deliverable goes into more details of the architecture and shows the main interactions of the Localization Manager with respect to the other components of the BUTLER architecture. As it can be observed from Figure 45, mainly, the Localization Manager interacts with the Communications Layer of the SmartObjects (here represented by the SmartObject Gateway) and with the Context Exposition (Pub/Sub Broker) functional component. In particular, the Localization Manager continuously receives raw ranging data (e.g. RSSI, ToA, AoA, etc.) from the SmartObject Gateway and in real-time processes these data to estimate the position of unknown SmartObjects by using a specific localization algorithm implemented in a centralized manner in the SmartServer.

In order to meet the BUTLER horizontal requirements coming from different domains (e.g. home/office, health, city/transport and shopping), the Localization Manager may implement either an adaptive localization algorithm, which is able to tune some key parameters, or a set of different algorithms, each of which performs well in specific scenarios and type of ranging measures. Thus, on the basis of scenario/environment and type of ranging measurements, the Localization Manager runs the most suitable localization algorithm among the implemented ones such that the positioning performance in the current scenario is optimized and application requirements are met.
When an estimate for a position of a specific SmartObject is available, the Localization Manager forwards this information to the Context Exposition (Pub/Sub Broker). More details about the interactions of the Localization Manager with both the SmartObject Gateway and the Context Exposition (Pub/Sub Broker) along with exchanged messages are presented in section 4.5.2.8.

### 4.4.6. BUTLER Data Discovery and Data Marketplace

#### 4.4.6.1. Introduction

One of the functionalities provided by BUTLER is the possibility of accessing exposed data generated by BUTLER SmartObjects. This exposed data can be roughly described as a continuous flow of values of a given Resource. Such functionality enables third-party developers to create new applications on top of BUTLER. BUTLER addresses the exposition and purchase of BUTLER SmartObjects data by said third-parties through a horizontal BUTLER SmartServer architecture that mediates between SmartObjects and applications. This functionality aims to enable Service Developers to discover and use data from SmartObjects through a single discovery point. It also addresses the management of simple events that Service Developers can define taking SmartObject data as input. That way, Service Developers can define a set of conditions that take SmartObject data as input and trigger actions whenever the conditions are met. Such simple events are named Business Events.

Two main roles are thus defined: the aforementioned Service Developer and the so-called Device Owner:

- **Device Owners** are the actual owners (or administrators) of SmartObjects. They are the ones that make a decision on sharing with third-parties the information coming from their SmartObjects.
- Said third parties, **Service Developers**, are in turn those wishing to use data coming from SmartObjects to create new services and applications.

Although the Data Discovery functionality (with the subsequent purchase and use of data sources by third parties) seems the main feature to address by BUTLER, that is just half the equation. The way the Device Owner manages data sources and how s/he decides to sell the information generated by SmartObjects is also relevant in order to implement the whole picture.

#### 4.4.6.2. Concepts

Discovery and purchase of SmartObject data follow the principles introduced in the BUTLER Information Model (see section 4.3.) and extends them in order to manage dynamic data streams. The inspiration to
said extension is the Xively\textsuperscript{12} data hierarchy. The offering of Xively is structured around a hierarchy of data types:

- **Feed.** It is a collection of measured data, often at a particular geolocation, defined by the creator of the feed and measured by sensors and devices. A feed can represent measures coming from both physical (such as a room, a mobile device, a building or a forest); and virtual entities such as a Second Life model, server bandwidth monitoring, etc.

- **DataStream.** It represents an individual sensor or measuring device within a feed. Every datastream must have a unique (within the feed) alphanumeric identifier. It can also specify 'units' (e.g. 'watts') as well as user-defined 'tags' (e.g. 'fridge_energy').

- **Datapoint.** It represents a single value of a datastream at a specific point in time. It is simply a key-value pair of a timestamp and the value at that time.\textsuperscript{13}

Regardless of the actual data hierarchy, the most powerful feature of the Xively approach is the introduction of the datastreams, which model the continuous generation of measures carried out by devices. Taken as starting point both the BUTLER Information Model and the BUTLER Service and Resource Model, a similar approach is followed: Resources of the type SensorData are considered, and their updated values (measures) are taken as the components of data streams (datapoints in the Xively model). Depending on the functionalities of SmartObjects, said measures can be obtained by using the SUBSCRIBE method (so that the SmartObjects are continuously “pushed” the measures they are taken) or by means of the GET method (executing a polling operation to create said Data Sources).

Thus, the following abstractions are used in BUTLER to manage discovery, exposition and purchase of SmartObject data. The rationale is to use concepts that are thought to be meaningful for the users of the system:

- **Sensors (Data Sources):** These are the basic elements that can be managed by the Device Owner with regard to the Data Exposition functionality. As mentioned previously, a Data Source is roughly equivalent to the continuous stream of measures (each of them named data items) provided by a Resource hosted by a BUTLER SmartObject. It can be seen as a Xively Datastream. A sensor (Data Source) is described by means of a Data Source Descriptor, that can comprise the following elements:

<table>
<thead>
<tr>
<th>Data Source ID</th>
<th>The unique identifier of the Data Source (mandatory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source Name</td>
<td>A meaningful name assigned by the owner of the Data Source</td>
</tr>
<tr>
<td>Data Source Type</td>
<td>A type of Data Source (from a predefined set of available types)</td>
</tr>
<tr>
<td>Data Source Unit</td>
<td>The unit of the measures (data items) the Data Source comprises</td>
</tr>
<tr>
<td>Data Source End-Point</td>
<td>A description of the protocols and semantics used to access the Data Source. It contains at least a URL.</td>
</tr>
<tr>
<td>Data Source Conditions</td>
<td>Constraints of the Data Source (such as the frequency of generation of data items or how fresh/fresh the data items are)</td>
</tr>
<tr>
<td>Associated Entity ID(s)</td>
<td>The identifiers of the Entities the Data Sources are associated to</td>
</tr>
</tbody>
</table>

\textsuperscript{12} Xively (http://xively.com/), initially known as Pachube and next as Cosm, is an on-line service provider allowing owners of sensors and devices to connect sensor data to the web and developers to build their own applications using such data. It was created in 2007 by Usman Haque, a British architect and artist, and released to the public in 2010. It is currently owned by LogMeIn.

\textsuperscript{13} Example: You wire up a hallway (‘feed’) with temperature, humidity and CO2 sensors (‘datapoints’). You create a feed named ‘ACME Hallway’, with three datastreams where the IDs could be: ‘temperature’, ‘humidity’ and ‘CO2’; which might be tagged ‘thermal,non-contact’, ‘capacitive, SHT21’ and ‘MG811’ respectively; and have units ‘Celsius’, ‘%RH’ and ‘ppm’. Individual datapoints at a point in time might be ‘23.2’, ‘34’ and ‘3820’ respectively.
Data Sources are comprised by **Data items** (equivalent to Xively Datapoints) are extremely simple in nature and are described using the following elements:

<table>
<thead>
<tr>
<th>DataSourceID</th>
<th>The identifier of the Data Source the data item belongs to (mandatory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Item Value</td>
<td>The value of the measure (mandatory)</td>
</tr>
<tr>
<td>Data Item Timestamp</td>
<td>When the measure was taken</td>
</tr>
</tbody>
</table>

- **Devices**: A BUTLER SmartObject. From a Data Discovery and Marketplace point of view, a SmartObject groups some sensors (Data Sources) with a common location and communication technology. It is equivalent to a Xively Feed.

- **Entities**: Virtual Entities were introduced by the IoT-A Domain Model and taken as one of the main concepts handled in the BUTLER Information Model. They represent real-world entities that are meaningful for the Device Owner. An entity is described by means of an Entity Descriptor, that can comprise the following elements:

<table>
<thead>
<tr>
<th>EntityID</th>
<th>The unique identifier of the Entity (mandatory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Name</td>
<td>A meaningful name assigned by the owner of the entity</td>
</tr>
<tr>
<td>Entity Type</td>
<td>A type of entity defining the class of entities a specific instance belongs to (from a predefined set of available types)</td>
</tr>
<tr>
<td>Entity Description</td>
<td>Natural language description of the entity</td>
</tr>
<tr>
<td>Entity Owner</td>
<td>A UserID defining the user that “owns” or “manages” the entity (mandatory)</td>
</tr>
<tr>
<td>Entity Coordinates</td>
<td>They describe the location and/or area the physical entity occupies</td>
</tr>
<tr>
<td>Entity Tags</td>
<td>Tags assigned by the entity owner</td>
</tr>
</tbody>
</table>

- **Data Offerings**: Said offerings group a set of sensors so that Device Owners can offer them as a selling unit. A Data Offering is described using the following elements:

<table>
<thead>
<tr>
<th>DataOfferingID</th>
<th>The unique identifier of the Data Offering (mandatory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Offering Name</td>
<td>A meaningful name assigned by the owner of the Data Offering</td>
</tr>
<tr>
<td>Data Offering Description</td>
<td>Natural language description of the Data Offering</td>
</tr>
<tr>
<td>Data Offering Owner</td>
<td>A UserID defining the user that created the Data Offering (mandatory)</td>
</tr>
<tr>
<td>Data Offering Coordinates</td>
<td>They describe the location and/or area the Data Offering is associated to</td>
</tr>
<tr>
<td>Data Offering Tags</td>
<td>Tags assigned by the Data Offering owner</td>
</tr>
<tr>
<td>Data Offering Price</td>
<td>Price assigned to the Data Offering by its owner</td>
</tr>
<tr>
<td>DataSourceID(s)</td>
<td>The identifiers of the Data Sources the Data Offering is made of</td>
</tr>
<tr>
<td>Data Offering Service Conditions</td>
<td>Constraints of the Data Offering if they apply to all Data Sources</td>
</tr>
</tbody>
</table>
Finally, BUTLER also provides the possibility of defining so-called Business Events (a set of conditions on Data Sources; when the conditions are evaluated to true the user receives a notification). Business Event is the term used to describe the simple event processing functionalities (see section 4.5.2.7) provided by BUTLER to Service Developers. Generally speaking, an event is a set of conditions that, if satisfied or evaluated to true, determine an action must be carried out. In BUTLER, the conditions an event is made of take data items that come from SmartObjects as inputs. Therefore, when the conditions that take as input SmartObject data are evaluated to true, an action is triggered. Such an action is usually the delivery of another data item to the event consumer. Said data item can be the one that made the set of rules evaluate to true. A Business Event is described using the following elements:

<table>
<thead>
<tr>
<th>BusinessEventID</th>
<th>The unique identifier of the Business Event (mandatory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Event Owner</td>
<td>A UserID defining the user that created the Business Event (mandatory)</td>
</tr>
<tr>
<td>Business Event Rules</td>
<td>The rules describing the processing of the events and the actions to be taken if the rules are evaluated to true</td>
</tr>
</tbody>
</table>

### 4.4.6.2.1. Device Owners functionalities

Device Owners are enabled to manage the controlled exposition and purchase of data sources from their SmartObjects. Device Owners use a management portal belonging to the BUTLER System/Management layer and hosted by a SmartServer in order to handle the management of the different options offered to them. Following the aforementioned concepts, the main options offered to the Device Owner through the BUTLER Data Management and Marketplace Portal are the following ones (see section 4.5.4.2 for further description):

- **Entity Management**: This option enables Device Owners to manage Virtual Entities. The Device Owner is expected to freely define entities using a map-based web interface and assign it a name, an entity type, a geolocalized shape and tags for subsequent discovery. Hierarchies of Virtual Entities are also supported. Provided that the Device Owner has been granted appropriate permissions over BUTLER SmartObjects, s/he is able also to associate the data they provide to Virtual Entities.

- **Data Source Management**: This option enables a Device Owner to manage SmartObjects and Data Sources (known as sensor for a more intuitive management). Provided that the Device Owner has been granted appropriate permissions over BUTLER SmartObjects, they are enabled to tag their sensors or to access the current sensor features. A batch loading of sensor descriptors is also available.

- **Data Offerings**: This option allows the definition of Data Offerings. Other available options are the definition of the service levels of the data delivered, the definition of access control levels (Device Owners can apply several privacy levels to packets, i.e. public, private, etc.), and offering price setting.
- **Storage Management**: It enables the definition of custom rules determining storage criteria (data sent by sensors could be saved always, never, or following a set of custom storage rules). Different levels of granularity are also supported (from single sensors to all the Data Sources associated to an Entity).

4.4.6.2.2. **Service Developer functionalities**

Service Developers are offered a Data Marketplace with embedded Data Discovery functionalities. They are able to discover Entities and Data Offerings exposed through the BUTLER SmartServer platform according to a given set of criteria (owner, price, tags, popularity, location...). They are also offered the possibility of defining so-called Business Events (a set of conditions on Data Sources; when the conditions are evaluated to true the user receives a notification). See below the main options offered to the Service Developer through the BUTLER Data Management and Marketplace Portal:

- **Search and Purchase**: this option embeds the Data Discovery functionality and, through a web map-based interface, enables Service Developers to discover appropriate Data Offerings that match their requirements. As stated previously, the owner or administrator of SmartObjects can characterize the Data Offerings created by grouping Data Sources. Beyond the location, Data Offerings are characterized by tags, ownership, service level and price. The Service Developer can navigate over a map to discover available Data Offerings and determine their features or, on the other hand, use a web form to introduce the search criteria s/he wishes.

![Figure 47: Example of the BUTLER interface for the discovery of Entities and Data Sources based on a map tool (the Entity is here depicted as a polygonal shape; Data Sources as map pins)](image)

- **Business Event Definition**: This option enables Service Developers to define Business Events. In BUTLER, the conditions a Business Event is made of take data items that come from SmartObjects as inputs. Therefore, when the conditions that take as input SmartObject data are evaluated to true, an action is triggered. Such an action is usually the delivery of another data item to the event consumer. Said data item can be the one that made the set of rules evaluate to true. Business Events are always defined among the Data Sources a Data Offering comprises.

4.4.6.3. **Involved architectural entities**

The high-level architecture of BUTLER Data Discovery and Data Marketplace is described in Figure 48. It has been abstracted in order to make it generic. The main elements it comprises are the following:
• **Data Source(s)**: The entry of the Data Discovery and Marketplace functionality is modeled as generic Data Sources. It is assumed that data sources are created from SmartObjects Resources of type SensorData and/or StateVariable. They are periodically polled in order to create a dynamic data stream.

• **Device Data Collector/Actuator**. This is the functional component through which data from SmartObjects arrives. It implements the necessary operations (GET, SUBSCRIBE) to receive or retrieve data items from SmartObjects (thus, it is also responsible for configuring SmartObject to send the information they generate). Said data items are the value of a Resource of type SensorData and/or State Variable that are received as the result of a SUBSCRIBE operation. For SmartObjects not supporting the delivery of information (that is, only providing information upon request) this functional component implements polling mechanisms to get information (using the GET operation).

• **Data Management and Marketplace Portal**: This functional component enables users to handle the functionalities described in section 4.4.6.2. It comprises two main subcomponent:
  - **Data Management and Marketplace Orchestrator**: It provides two different front-ends: to Device Owners, and to Service Developers;
  - **Data Marketplace Registry**: It stores the information about the Data Offerings. Said information is managed through the Data Management and Marketplace Orchestrator.

• **Data Discovery**. This functional component keeps a record of available Data Sources and Entities. In the former case, it stores all the parameters needed to know the type of information the Data Source generates and how to access it (Data Source Descriptor). In the latter, the main parameters recorded are those related to the description and location of the Entity (Entity Descriptor).

• **Complex Event Processing Engine**. The Complex Event Processing engine takes the responsibility of transforming the incoming data streams (from Data Sources) into new outgoing data streams each of them being one of the Data Offerings the Device Owner has defined. It is also responsible for managing persistent storage.

• **Simple Event Processing**. This functional component takes the responsibility of managing Business Events and its role is usually played also by the Complex Event Processing engine.

• **Resource Exposition**. This functional component takes the responsibility of exposing data coming from Data Sources. There are two approaches for this component:
  - **A Publication/Subscription Broker**. It is a regular publication/subscription broker receiving the outgoing data streams provided by the Complex Event Processing engine and delivering them to the entities on behalf of which the Service Developer has subscribed to. This broker can be any publication/subscription broker (for instance, a MQTT broker [19], such as Mosquitto [20], or a PubSubHubbub hub [22]).
  - **Pull Interface**. It provides an interface for authorized entities to access SmartObject data stored in the SmartServer.

• **Persistent Storage**. This functional component is a NoSQL database that stores the information coming from SmartObjects for subsequent access.

• **Data Consumer(s)**. The entities that receive information from Data Sources or from the execution of Business Events. They are usually BUTLER Applications created by Service Developers.
4.4.6.4. Relevant use cases

There are several use cases associated to the Data Discovery and Data Marketplace functionality. In the following sections, only the most relevant ones are addressed:

- **Virtual Entity Management**: It deals with the creation of Virtual Entities and the association of Data Sources to them. Thus, it describes the way a Device Owner creates Virtual Entities and associates some of his/her Data Sources (provided by his/her SmartObjects) to said Entities. See section 4.4.6.4.1 for further description.

- **Creation of Data Offerings for subsequent selling**: This use case describes how the Device Owner reviews his/her Data Sources and Data Offerings them for subsequent selling. See section 4.4.6.4.2 for further description.

- **Search and purchase of Data Offerings**: This use case describes how the Service Developer discovers relevant Data Offerings, purchases them and obtains necessary credentials for subsequent use of the data exposed through a Data Offering. See section 4.4.6.4.3 for further description.

- **Definition of Business Events**: This use case describes how the Service Developer defines so-called Business Events. These are a set of conditions on Data Sources; when the conditions are evaluated to true the user receives a notification. See section 4.4.6.4.4 for further description.

- **Reception of SmartObject data**: This use case describes how data exposed through Data Offerings is received by the applications developed by the Service Developer. Said applications are subscribed to data streams that pass on the data from SmartObjects. See section 4.4.6.4.5 for further description.

- **Access to cached SmartObject data**: This use case describes how data mediated and cached by the BUTLER SmartServer can be subsequently accessed by applications developed by the Service Developer. See section 4.4.6.4.6 for further description.
Next sections will follow the typical use case structure, including a number of pre-requisites and a regular sequence diagram. For all of them, the following pre-requisites are established:

- Users (either Device Owners or Service Developers) have been registered in the Data Management and Marketplace Portal and have obtained necessary credentials to perform authentication.
- Relevant Resources tied to SmartObjects have been already registered (by running the use cases associated to the Security Services; proper synchronization has been achieved with the Data Discovery functional component). Although Data Sources are usually bound to SmartObjects (as Resources), it is possible to use other Data Sources where data items do not provide from a SmartObject.
- Data Sources have been got associated to the appropriate Device Owners. Said Device Owners are the actual owners of the SmartObjects hosting Data Sources or have been entitled to define the way their data is shared.
- Data Sources have been configured to send the information they gather to a SmartServer. This server implements the Device Data Collector/Actuator functional component coping with protocol and data model adaptation. This Device Data Collector/Actuator can be configured to forward the incoming data streams to other functional components such as the Complex Event Processing engine.
- The Data Management and Marketplace Portal provides an interface (usually graphical) for Device Owners (to manage Entities and Data Offerings) and for Service Developers (to discover Data Offerings and define Business Events).

4.4.6.4.1. Virtual Entity Management

This use case describes how a Device Owner manages Virtual Entities and associate Data Sources to them. The involved architectural entities and the sequence diagram are shown in the figures below:

![Figure 49: Virtual Entity Management architecture](image-url)
A detailed explanation is provided here:

1. The user, a Device Owner, accesses the Data Management and Marketplace Orchestrator, a subcomponent of the Data Management and Marketplace Portal. The initial access usually involves the authentication of the user so that the Orchestrator gets aware of the user identity and therefore obtains a UserID. This authentication step is performed in cooperation with the User Authentication functional component (not shown for simplicity).

2. Next, the Device Owner selects the option ‘Create New Entity’. The Device Owner is enabled to draw, over a map, the shape of the Virtual Entity s/he wishes to create. S/he provides at least a name, a type, and a shape (with implicit coordinates). Tags can be also assigned to the Entity. The Device Owner also defines whether the Entity and its associated Data Sources are visible to other users or not.

3. The Virtual Entity Descriptor is stored into the Data Discovery. The descriptor includes the information provided by the Device Owner in the previous step. An EntityID is automatically assigned by the Data Discovery and returned.
4. Next, the Device Owner selects the option of associating Data Sources to Virtual Entities.
5. Thus the Data Management and Marketplace Orchestrator accesses the Data Discovery and retrieves available Data Source Descriptors (availability means, in this stage, the Data Sources registered in the platform the Device Owner is responsible for). This process can be executed as many times as needed. For the sake of simplicity just one of said processes is shown. The query to the Data Discovery includes the UserID (to enable access control) and optionally base geographical coordinates to ease the location of Data Sources.
6. The Data Discovery sends back a set of Data Source Descriptors. Depending on the presence of coordinates in the initial query, a different set of Data Source Descriptors is returned.
7. The Data Management and Marketplace Orchestrator graphically shows the available Data Sources to the user. A map-based interface is provided.
8. The Device Owner picks as many as wished Data Sources and assigns them to an Entity.
9. The Entity Descriptor is updated with the identifiers of the Data Sources associated to the Entity by the user.
10. Next, the user selects the option of creating a hierarchy of Entities.
11. Thus the Data Management and Marketplace Orchestrator accesses the Data Discovery and retrieves available Entity Descriptors (availability means, in this stage, the Entities created previously by the Device Owner). The query to the Data Discovery includes the UserID (to enable access control) and optionally base geographical coordinates to ease the location of Entities.
12. The Data Discovery sends back a set of Entity Descriptors.
13. The Data Management and Marketplace Orchestrator graphically shows the available Entities to the user. A map-based interface is provided.
14. The Device Owner picks some of them and creates an Entity hierarchy.
15. The Entity Descriptor is updated with the identifiers of the Entities involved in the hierarchy.

4.4.6.4.2. Data Offering management

The involved architectural entities and the sequence diagram are shown in the figures below:

![Data Offering Creation architecture](image-url)
A detailed explanation is provided here:

1. The user, a Device Owner, accesses the Data Management and Marketplace Orchestrator, a subcomponent of the Data Management and Marketplace Portal. The initial access usually involves the authentication of the user so that the Orchestrator gets aware of the user identity and therefore obtains a UserID. This authentication step is performed in cooperation with the User Authentication functional component (not shown for simplicity).

2. Next, the Device Owner selects the option ‘Create New Data Offering’. The user creates a new Data Offering by assigning a name, a type, a set of tags, a price and some service levels conditions.

3. The Data Offering Descriptor is stored within the Data Marketplace Registry. A DataOfferingID is automatically assigned and returned.

4. Next, in order to allow the association of Data Sources to Data Offerings, the Data Management and Marketplace Orchestrator accesses the Data Discovery to retrieve available Data Source Descriptors (availability means, in this stage, the Data Sources the Device Owner is responsible for). The query to the Data Discovery includes the UserID (to enable access control) and optionally base geographical coordinates to ease the location of Data Sources.

5. The Data Discovery sends back a set of Data Source Descriptors.

6. The Data Management and Marketplace Orchestrator graphically shows the available Data Sources to the user. A map-based interface is provided.

7. The Device Owner picks as many as desired Data Sources and assigns them to a Data Offering.

8. The Data Offering Descriptor is updated within the Data Marketplace Registry. The update operation includes the DataOfferingID and the identifiers of the Data Sources the offering is made of.

Figure 52: Data Offering Creation sequence diagram
4.4.6.4.3. **Search and Purchase of Data Offerings**

This additional pre-requisite is established:

- A number of Data Offerings have been previously created by Device Owners so that they are ready for purchase.

The involved architectural entities and the sequence diagram are shown in the figures below:

![Figure 53: Search and Purchase architecture](image-url)
A detailed explanation is provided here:

1. The user, a Service Developer, accesses the Data Management and Marketplace Orchestrator, a subcomponent of the Data Management and Marketplace Portal. The initial access usually involves the authentication of the user so that the Orchestrator gets aware of the user identity and therefore obtains a UserID. This authentication step is performed in cooperation with the User Authentication functional component (not shown for simplicity). Next, the Service Developer selects the ‘Data Offering Search’ option. The Data Management and Marketplace Orchestrator provides an interface to perform the search operation.

2. The Service Developer introduces the criteria s/he wishes. Criteria available are price, tag, type (optional), name (optional), and service conditions.

3. The Data Management and Marketplace Orchestrator accesses the Data Marketplace Registry and queries for the Data Offerings that matches search criteria.

4. The Data Marketplace Registry returns the descriptors of the Data Offerings that matches the Service Developer’s search criteria.

5. The Data Management and Marketplace Orchestrator shows the Data Offerings that matches the Service Developer’s criteria.
6. The Service Developer agrees to purchase one or several the Data Offerings (the actual payment procedure is not shown as it depends on the deployment scenario).

7. The Data Management and Marketplace Orchestrator translates the Data Offering Descriptor (comprising the identifiers of the Data Sources a Data Offering comprises and the service level conditions associated to the Data Offering) into CEP applications to be deployed into the Complex Event Processing Engine. Before deploying them, the Data Management and Marketplace Orchestrator orders the Device Data Collector/Actuator to forward all the incoming data items from Data Sources belonging to the Data Offerings to the CEP engine. It includes the identifiers of the Data Sources in the request.

8. Data Sources are instructed to forward data streams (the mechanism to do so is not described in detail here as it is not the purpose of the description).

9. Next, the Data Management and Marketplace Orchestrator deploys the CEP applications into the CEP engine, so that it is able to process incoming data streams from Data Sources, generate new data streams representing the streams the Data Offering comprises and forward them to the Resource Exposition Pub/Sub Broker. To do so, certain information must be embedded into the CEP applications. Namely the identifier of the Data Offering, a token identifying the subsequent user subscription and the identifiers of the involved Data Sources. These elements are included for being subsequently embedded into the outgoing data streams and make said data streams able to be identified by the Resource Exposition Pub/Sub Broker.

10. The Resource Exposition Pub/Sub Broker receives the configuration information it needs to handle subsequent user subscriptions requests. Such configuration information contains the credentials associated to the user.

11. The user receives the necessary credentials to be used to subsequently request the subscription to the Data Offering data items. Said credentials can be provided in any format but must contain at least the credentials required to access the Resource Exposition Pub/Sub Broker and the Subscription Token identifying the Data Offering s/he wishes to be subscribed to.

12. The Data Management and Marketplace Orchestrator accesses the Data Marketplace Registry in order to store the information about which Data Offerings have been subscribed to by which Service Developers.

4.4.6.4.4. Definition of Business Events

This additional pre-requisite is established:

- Service Developers have purchased the Data Offerings they are interested in (that is, the use case Search and Purchase of Data Offerings has been executed). Business Events are defined as a set of rules that take as input data items belonging to the Data Sources an offering is made of.

The involved architectural entities and the sequence diagram are shown in the figures below:
Figure 55: Definition of Business Events architecture
A detailed explanation is provided here:

1. The user, a Service Developer, accesses the Data Management and Marketplace Orchestrator, a subcomponent of the Data Management and Marketplace Portal. The initial access usually involves the authentication of the user so that the Orchestrator gets aware of the user identity and therefore obtains a UserID. This authentication step is performed in cooperation with the User Authentication functional component (not shown for simplicity).

2. Next, the Service Developer selects the ‘Create Business Event’ option.

3. The Data Management and Marketplace Orchestrator accesses the Data Marketplace Registry and retrieves all the descriptors of all the Data Offerings the Service Developer has previously purchased.

4. The Data Marketplace Registry sends back a set of Data Offering Descriptors.

5. The Data Management and Marketplace Orchestrator shows the Data Offerings to the Service Developer.

6. The Service Developer selects one of said Data Offerings.
7. The Data Management and Marketplace Orchestrator accesses the Data Discovery in order to retrieve the information about the Data Sources the Data Offering is made of. The query to the Data Discovery includes the all the Data Source identifiers retrieved in step 4 as part of the Data Offering Descriptors.

8. The Data Discovery sends back a set of Data Source Descriptors.

9. The Data Management and Marketplace Orchestrator shows the available Data Sources to the Service Developer.

10. The Service Developer defines the conditions the Business Event comprises by means of a graphical interface that encapsulates a rule manager. Conditions are composed by means of a set of rules that are applied to available Data Sources. At least the following operators are available:
    - Simple mathematical operations to be applied to Data Sources: addition, subtraction, multiplication, division...
    - Simple statistical operations to be applied to a set of Data Sources: average, minimum, maximum.

The following data values are considered:
    - Current (or last) value within a Data Source (that is, the last data item generated by the Data Source)
    - Average (also maximum or minimum) value within a Data Source (that is, taking the last data items over a period of time; or the last data items). These operations include as attribute a period of time or a number of values (“average value over the last 10 values” or “average value over the last 10 seconds”).

The Service Developer defines also the action to be executed. Said action would be typically the delivery of a value that could be computed using the same type of rules already mentioned.

11. Once the Service Developer has created the Business Event declaration, the Data Management and Marketplace Orchestrator transforms the conditions the Business Event declaration is made of into CEP applications to be deployed into the Complex Event Processing engine that plays the role of Simple Event Processing functional component, so that it is able to detect whether the conditions are evaluated to true. If so, the CEP engine would compute the result of the action defined in the Business Event declaration and forward the result to the Resource Exposition Pub/Sub Broker. Next, the Management and Marketplace Orchestrator deploys the CEP applications into the CEP engine, so that it is able to process incoming data streams from Data Sources and evaluate whether the Business Event conditions are met. To do so, certain information must be embedded into the CEP applications. Namely a BusinessEventID, a token identifying the subsequent user subscription, and the identifiers of the involved Data Sources. The first two elements (BusinessEventID and Subscription Token) are included for being embedded into the outgoing data streams and make said data streams able to be identified by the Resource Exposition Pub/Sub Broker.

12. The Resource Exposition Pub/Sub Broker receives the configuration information it needs to handle subsequent user subscriptions requests. Such configuration information contains the credentials associated to the user.

13. The Service Developer receives the necessary credentials to be used to request the delivery of Business Events results. Said credentials can be provided in any format but must contain at least the credentials required to access the Resource Exposition Pub/Sub Broker and the Subscription Token identifying the Business Events s/he wishes to be subscribed to.

14. The Data Marketplace Registry is updated. The relationship of the Data Offering to the BusinessEventID and to the user that has created it is stored.

4.4.6.4.5. Reception of SmartObject data

This additional pre-requisite is established:
    - Service Developers have purchased the Data Offerings they are interested in (that is, the use case Search and Purchase of Data Offerings has been executed).
- Data Consumers (the entities, usually BUTLER applications, on behalf of which the Service Developer has requested data) have previously accessed the Resource Exposition Pub/Sub Broker to subscribe to the delivery of SmartObject data using the credentials obtained by the Service Developer in step 13 in use case Search and Purchase of Data Offerings (in fact, any entity can execute this pre-requisite on behalf of the Data Consumers; the only relevant issue is a Data Consumer end-point to have been provided for subsequently receiving the data items).

The involved architectural entities and the sequence diagram are shown in the figures below:
A detailed explanation is provided here:

1. Initially, the Service Developer makes a subscription request towards the Resource Exposition Pub/Sub Broker. Such a request includes the credentials received in step 13 in use case Search and Purchase of Data Offerings.

2. Data Sources send information to the Device Data Collector / Actuator functional component. It makes basis tasks of protocol and data model adaptation and forwards the transformed data streams to the Complex Event Processing engine. The Device Data Collector / Actuator is able to insert the DataSourceId into the data items forwarded to the Complex Event Processing engine.

3. According to the configuration information received in step 8 in previous use case (Search and Purchase of Data Offerings), relevant data streams are forwarded to the Complex Event Processing engine (mind that this step does not preclude the Device Data Collector / Actuator from forwarding the same data streams to additional data sinks with other purposes).

4. The Complex Event Processing engine applies the CEP applications received in step 9 in the previous use case (Search and Purchase of Data Offerings) to incoming data streams so that it is able to determine which of them must be forwarded to the Resource Exposition Pub/Sub Broker. The outgoing data streams will include the DataOfferingId the new data streams refers to, and the Subscription Token needed to process subsequent user subscriptions. The data streams filtered by the Complex Event Processing engine are forwarded to the Resource Exposition Pub/Sub Broker.

5. Data items are delivered to Data Consumers according to the available subscription (to those matching the Subscription Tokens included in the data items received by the Resource Exposition Pub/Sub Broker).

4.4.6.4.6. Access to cached SmartObject data

The following pre-requisites are established:

- Service Developers have previously purchased the Data Offerings they are interested in (that is, the use case Search and Purchase of Data Offerings has been executed).
- Additionally, s/he has stated that s/he wishes the data items belonging to a given Data Offering must be available for subsequent access (as part of the Search and Purchase of Data Offerings the step 9 deploys a CEP application that instructs data items belonging to a given Data Offering to be stored in the Persistent Storage functional component).
- A similar use case to the Reception of SmartObject data is run. However, step 4 redirects data items to the Persistent Storage functional component instead (or besides) forwarding them to the Resource Exposition Pub/Sub Broker.
- The Data Consumer accesses the Pull Interface implemented by the Resource Exposition functional component. The initial access usually involves the authentication and authorization of the Data Consumer. These steps are performed in cooperation with the Authorization Service functional component (not shown for simplicity).

The involved architectural entities and the sequence diagram are shown in the figures below:
A detailed explanation is provided here:

1. The Data Consumer provides the identifier of the Data Offering it wishes to access (filtering conditions can be also provided: usually time constraints). The Resource Exposition (Pull Interface) functional component gets also aware about the UserID.
2. The Resource Exposition (Pull Interface) functional component verifies whether the user is allowed to access the requested Data Offering data items by querying the Data Marketplace Registry.
3. The Data Marketplace Registry sends back the information.
4. If allowed, the Resource Exposition (Pull Interface) functional component queries the Persistent Storage functional component providing the Data Offering Id and any other filtering condition provided by the Data Consumer.
5. The data items that match the query are sent back to the Resource Exposition (Pull Interface) functional component.
6. The data items that match the query are sent back to the Data Consumer.

4.4.7. BUTLER Context Management

4.4.7.1. Introduction

In its seminal article “Understanding and Using Context” ([8]), Anind K. Dey provided a canonical definition of context:
Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

Not only had he provided a context definition, but a definition of what context awareness means:

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task.

A straightforward conclusion is that although nothing prevents context from comprising only “static” attributes (that is, intrinsic features of the entity, as they can be relevant for the situation of said entity) context is mainly focused on dynamic information that changes over time.

Standards and efforts such as IoT-A [9] or inherited from OMA Next Generation Services Interface (NGSI) Context Management [13] mainly focus on how context information can be exposed to clients interested in said information (in fact IoT-A does not even talk about ‘context’; it can be implicitly inferred that context is the set of attributes a given virtual entity owns). However, the way the users of such context are enabled to define it or how it is computed is not explicitly described. It is a key aspect in a dynamic environment with different stakeholders in which a myriad of data sources hosted by a large amount of SmartObjects can be used to compute the value of a specific context element of a virtual entity (using the IoT-A Domain Model parlance).

As described previously, IoT-A does not actually mention the term ‘context’. Although the IoT ARM states that Virtual Entities14 have attributes, it does not specify how the value of said attributes is computed. It only states that Services have the responsibility of carrying out such a task (that is, offering the values of said attributes associated to Virtual Entities) and that they will do so by using Resources hosted by Devices (Resources are defined as the software components that actually provide information about, or enable the actuation on Physical Entities; they are usually tightly coupled to the actual Devices and have native interfaces). Resources are also associated to Virtual Entities. However, the way Services carry out the computation of the attribute value is left for implementation.

Similar approaches can be found in OMA NGSI Context Management as it does not cover at all how context is generated (that is, it leaves the Context Generation to so-called Context Producers without stating how context is declared and how the value of the context attributes are generated).

BUTLER has taken two main inputs when defining its context management architecture. First one is the abovementioned IoT-A, as it provides the conceptual framework that BUTLER has taken as basis. See the relationship between the BUTLER Information Model and the BUTLER Context Model in Figure 61.

---

14 The terms in italics in this paragraph are taken from the IoT-A Domain Model.
Secondly, and considering the gaps mentioned in the previous section, BUTLER has surveyed a more practical example: FI-WARE [11]. It aims to provide a Core Platform comprising a number of so-called Generic Enablers. The FI-WARE vision is structured into six Technical Chapters, one of them being devoted to Data/Context Management, which is defined as “the facilities for effective accessing, processing, and analyzing massive volume of data, transforming them into valuable knowledge available to applications” [14]. The most relevant Generic Enablers defined in said Technical Chapter [16] are a Publish/Subscribe Broker GE (that enables applications to interchange heterogeneous events following a standard publish–subscribe paradigm); and a Complex Event Processing GE. Additional enablers are also available. The relationship between them can be seen in the figure below (from [17]). In said figure, the relationship with data sources from the IoT domain is modeled by means of the IoT GEs (in the upper left corner of the picture).
Thus FI-WARE introduces Complex Event Processing (CEP) as one of the technologies to manage context. CEP deals with the analysis of event data in real-time to generate immediate insight and enable instant response to changing conditions. The technology and implementations of CEP provide means to expressively and flexibly define and maintain the event processing logic of the application. The functionality of a CEP engine (ruled by the abovementioned event processing logic) is usually implemented by means of so-called CEP applications. Said applications determine the way the CEP engine works in order to detect patterns or conditions, execute actions as a result of said conditions, filter incoming data streams or generate new data streams as the result of the application of conditions.

However, although FI-WARE introduces a high-level architecture for Data/Context Management, nothing is said about how the cooperation between generic enablers is done in order to compute the context attributes and how it would work in a specific IoT environment as the one defined by IoT-A. In fact, although interfaces to IoT generic enablers, applications and a Pub/Sub Context broker are mentioned, nothing is said about how to do it in a real deployment.

As with Data Discovery and Data Marketplace, BUTLER uses Complex Event Processing as one of the main enablers of Context Management. Although there are other approaches for specific components of the context (such as with Localization Management, see section 4.5.2.4; or with User Behavior extraction, see section 4.5.2.8) BUTLER provides a general purpose architecture for creating basic context information associated to an Entity. As with Data Discover and Data Marketplace, a so-called Context Management Portal is introduced. It aims to orchestrate the cooperation between entities such as a Context Event Processing engine or a Pub/Sub Broker in order to compute and expose context information. This functional component belongs to the System/Device Management Layer as part of the functional group Management Portal.

\[15\] Image taken from [17]
Context Management is targeted to Service Developers (see users categories in section 4.4.6.1) that wish to pick up relevant Entities and define the context associated to them so that their applications can use the context information to customize the way the application works. The Context Management Portal provides the means for said user, to be able to retrieve the information about available Entities and Data Sources and to declare which the components of the context (for a given Entity) are and how their values are computed. Therefore, it works in close cooperation with the Data Discovery functional component so that it is able to provide the necessary information to the user. Next, it implements a method that comprises several steps. The most important of them is the translation of the context declaration rules introduced by the user (usually taking a Business Rule Engine as component within the Context Management Portal) into CEP applications that can be deployed to the CEP engine and understood by it so that it is able to create new data sources representing context attributes. These new data sources are the result of executing real-time mathematical/statistical operations on the data items generated from the different incoming Data Sources.

4.4.7.2. Concepts
As described in the previous section, two main concepts are used by the Context Management functionality: Virtual (Entities) and Data Sources. Definition of both concepts and their associated attributes are provided in section 4.4.6.2.

4.4.7.3. Involved architectural entities
The high-level Context Management architecture is shown in Figure 63. It has been abstracted in order to make it generic. The main elements it comprises are the following (notice its similarity to the Data Discover and Data Marketplace architecture):

- **Data Source(s):** The entry of the Context Management functionality is modeled as generic Data Sources. They are based on the Resource information provided by SmartObjects related or placed in the vicinity of physical entities. It is expected that the context of a given Entity can be composed from the information sensed by said SmartObjects. Therefore, Data Sources will be used by the user to declare the contents and value of context attributes associated to a given Entity.

- **Device Data Collector / Actuator.** This is the functional component through which data from SmartObjects arrives. It implements the necessary operations (GET, SUBSCRIBE) to receive or retrieve data items from SmartObjects (thus, it is also responsible for configuring SmartObject to send the information they generate). Said data items are the value of a Resource of type SensorData and/or State Variable that are received as the result of a SUBSCRIBE operation. For SmartObjects not supporting the delivery of information (that is, only providing information upon request) this functional component implements polling mechanisms to get information (using the GET operation).

- **Context / Behavior Information Provider.** This functional component is responsible for providing context information. It relies on a Complex Event Processing Engine to provide its functionality. With regard to Context Management, the Complex Event Processing engine takes the responsibility of transforming the incoming data streams (from Data Sources) into new outgoing data streams each of them being one of the attributes the context is made of. The transformation process is ruled by the CEP applications created by from the Context Management Portal.

- **Context Management Portal:** This functional component, a subcomponent of the Management Portal, enables users to handle the context associated to Virtual Entities. It is responsible for:
  a) Providing an interface towards the user.

---

16 See section 4.5.2.2 for a discussion on the explicit and implicit use of the Complex Event Processing engine as functional component: it is implicit when it is used as basis for another functional component; it is explicit when it is used "as is".
b) Enabling the user, through that interface, to pick up relevant Entities and Data Sources and, by means of a set of predefined rules, declare which attributes the context of a given Entity is made of, and how the value of said attributes is computed from the available Data Sources.

c) Translating the user declarations into the native code handled by the Complex Event Processing engine the Context / Behavior Information Provider is based on.

d) Feeding such native code into the Complex Event Processing engine.

e) Handling the subscription to notifications on context values, usually on behalf of a service or application managed by the user.

To sum up, this functional component handles the following process: Data Sources selection → context rules declaration → deployment to the Context / Behavior Information Provider (a CEP engine).

- **Data Discovery.** This functional component keeps a record of available Data Sources and Entities. In the former case, it stores all the parameters needed to know the type of information the Data Source generates and how to access it (Data Source Descriptor). In the latter, the main parameters registered are those related to the description and location of the Entity (Entity Descriptor).

- **Context Exposition.** This functional component takes the responsibility of exposing context information. It is a regular publication/subscription broker receiving the outgoing data streams provided by the Context / Behavior Information Provider and delivering them to the entities on behalf of which the Service Developer has subscribed. This broker can be the Context Broker defined by OMA NGSI Context Management or any other publication/subscription broker (for instance, a MQTT broker [19], such as Mosquitto [20], or a PubSubHubbub hub [22]). From a purely functional point of view, it will offer data streams (representing each the evolving values of a given context attribute) and therefore it is totally equivalent to the Resource Exposition functional component described in the Data Discovery and Data Marketplace functionality.

- **Data Consumer(s).** The entities that receive the context information created from Data Sources. They are usually BUTLER applications created by service developers.
4.4.7.4. Relevant use cases

There are several use cases associated to the Context Management functionality. In the following sections, only the most relevant ones are addressed:

- **Declaration of Context.** Here, it will be described how the context elements are defined by the user and how this definition is fed into the Context / Behavior Information Provider. See section 4.4.7.4.1.
- **Generation of Context.** This use case describes how the context attributes are computed and delivered to context consumers. See section 4.4.7.4.2.

4.4.7.4.1. Declaration of context

This section follows a typical use case structure. It aims to describe a typical procedure involving the declaration of context. Thus, the following pre-requisites are established:

- Users (Service Developers) have been registered in the Context Management Portal and have obtained necessary credentials to perform authentication.
- Relevant (Virtual) Entities have been already defined and registered in the Data Discovery functional component. As Entities are created by Device Owners, they have made them visible.
- The “owner” of the Entity has provided appropriate permissions so that the service developer has been entitled to use said Entities.
- Relevant Resources tied to SmartObjects have been already registered (by running the use cases associated to the Security Services; proper synchronization has been achieved with the Data Discovery functional component). Although Data Sources are usually bound to SmartObjects (as Resources), it is possible to use other Data Sources where data items do not provide from a SmartObject.
- Data Sources have been configured to send the information they gather to a SmartServer. This server implements the Device Data Collector / Actuator functional component coping with protocol and data model adaptation. This Device Data Collector / Actuator can be configured to forward the incoming data streams to other functional components such as the Context / Behavior Information Provider.
- The Context Management Portal provides an interface (usually graphical) for Service Developers to discover relevant Entities and Data Sources (usually by means of tags, location, description, owner...).

The involved architectural entities and the sequence diagram are shown in the figures below:

---

17 Solid lines: actual flow of data from data sources (SmartObjects) to applications. Dashed lines: management operations (including the deployment of event processing logic (CEP applications) onto the Complex Event Processing engine.
Figure 64: Context Declaration
A detailed explanation is provided here:

1. The user, a Service Developer, accesses the Context Management Portal. The initial access process usually involves the authentication of the user so that the Context Management Portal gets aware of the user identity and therefore obtains a UserID. This authentication step is performed in cooperation with the User Authentication functional component (not shown for simplicity). Next, the user selects the option ‘Entities and Data Sources Search’.

2. The Context Management Portal accesses the Data Discovery functional component to retrieve available Entity and Data Source Descriptor. This process can be executed as many times as needed. For the sake of simplicity just one instance is used. The query to the Data Discovery includes the UserID (to enable access control) and optionally base geographical coordinates to ease the location of Entities and Data Sources.

3. The Data Discovery sends back a set of Entity and Data Source descriptors that match the search criteria (UserID and base coordinates if available).

4. The Context Management Portal graphically shows the available Entities and Data Sources to the user. A map-based interface is usually used.

Figure 65: Context Declaration sequence diagram
5. The user declares the elements of the context(s) s/he is interested in by means of a graphical interface that encapsulates a rule manager. The procedure is as follows: Once the user has determined the Entity the context must be associated to, s/he declares the attributes the context associated to said Entity has to comprise in terms of: name and type of value. Next, s/he describes how the value of each attribute has to be computed, by using a set of rules that have to be applied to Data Sources (Data Source can be already associated to the Entity or not)\(^ {18}\).

The following operators are available:
- Simple mathematical operations applied to Data Sources: addition, subtraction, multiplication, division...
- Simple statistical operations applied to a set of Data Sources: average, minimum, maximum.

The following timing segmentation options are considered:
- Current (or last) value within a Data Source (that is, the last data item generated by the Data Source)
- Average (also maximum or minimum) value within a Data Source (that is, taking the last data items over a period of time; or the last data items). These operations include as attribute a period of time or a number of values (“average value over the last 10 values” or “average value over the last 10 seconds”).

Examples (in pseudocode) can be as follows (‘ds’ stands for “data source”):
- Value (entity1.attr1) := 10 * CurrentValue (ds1) + 5 * CurrentValue (ds2)
- Value (entity1.attr2) := AverageValue (ds1, 10)
- Value (entity1.attr3) := Maximum (AverageValue (ds3, 5), AverageValue (ds4, 5))

Additionally, the Context Management Portal registers the context declarations it has generated in order not to create redundant CEP applications (that is, preventing users from creating context attributes that have been already created).

6. Once the user has declared the context attributes, the Context Management Portal transforms the declarations into CEP applications to be deployed into the Context / Behavior Information Provider (as it is an instance of a Complex Event Processing engine with a specific CEP behavior). Before deploying them, the Context Management Portal orders the Device Data Collector / Actuator to forward all the incoming data items from Data Sources involved in the context declarations to the Context / Behavior Information Provider. It includes the identifiers of the Data Sources in the request.

7. Data Sources are instructed to forward data streams (the mechanism to do so is not described in detail here as it is not the purpose of the description)

8. Next, the Context Management Portal deploys the CEP applications into the Context / Behavior Information Provider, so that it is able to process incoming data streams from Data Sources, generate new data streams representing the context attributes and forward them to the Context Exposition Pub/Sub Broker. To do so, certain information must be embedded into the CEP applications. Namely the ID of the Entity the attribute refers to, the ID of the handled context attribute, a token identifying the subsequent user subscription, and the identifiers of the involved Data Sources. The first three elements (EntityID, ContextAttributeID and Subscription Token) are included for being embedded into the outgoing data streams and make said data streams able to be identified by the Context Exposition Pub/Sub Broker.

9. The Context Exposition Pub/Sub Broker receives the configuration information it needs to handle subsequent user subscriptions requests. Such configuration information contains the credentials associated to the user.

10. The user receives the necessary credentials to subsequently request the subscription to the context data items. Said credentials can be provided in any format but must contain at least the credentials

\(^ {18}\) An example of context value associated to an entity of type ‘building’ can be the maximum current temperature value in the building. If several temperature sensors are deployed, the maximum current temperature value would be the maximum value being sensed by temperature devices.
required to access the Context Exposition Pub/Sub Broker and the Subscription Token identifying
the context attributes s/he wishes to be subscribed to.

4.4.7.4.2. Generation of context

The following pre-requisites are established:

- Data Sources are configured to send the information they generate to the Device Data Collector /
  Actuator functional component. It takes care of protocol and data model adaptation.
- Previous use case (Declaration of context) has been run so that the Data Sources the context
  attributes relies on will be forwarded to the Context / Behavior Information Provider.
- Data Consumers (the entities, usually applications, on behalf of which the Service Developer has made
  a context declaration) have previously accessed the Context Exposition Pub/Sub Broker to subscribe to
  the delivery of context information using the credentials received by the user in step 10 in the previous
  use case (in fact, any entity can execute this pre-requisite on behalf of the Data Consumers; the only
  relevant issue is a Data Consumer end point to have been provided for subsequently receiving the data
  items).

The involved architectural entities and the sequence diagram are shown in the figures below:

Figure 66: Context Information Distribution
Figure 67: Context Consumption sequence diagram

A detailed explanation is provided here:

1. Initially, the Service Developer makes a subscription request towards the Context Exposition Pub/Sub Broker. Such a request includes the credentials received in step 10 above.

2. Data Sources send information to the Device Data Collector / Actuator functional component. It makes basic tasks of protocol and data model adaptation and forwards the transformed data streams to the Context / Behavior Information Provider. The Device Data Collector / Actuator is able to insert the DataSourceID into the data items forwarded to the Context / Behavior Information Provider.

3. According to the configuration information received in step 6 in previous use case (Declaration of context) relevant data streams are forwarded to the Context / Behavior Information Provider (mind that this step does not preclude the Device Data Collector / Actuator from forwarding the same data streams to additional data sinks with other purposes).

4. The Context / Behavior Information Provider applies to incoming data streams the CEP applications received in step 8 in previous use case (Declaration of context) so that it is able to create new data streams each of them being one of the context attributes associated to an Entity. The outgoing data streams will include the EntityID, the ContextAttributeID the new data streams refers to, and the Subscription Token needed to process subsequent user subscriptions. The new Data Sources generated by the Context / Behavior Information Provider are forwarded to the Context Exposition Pub/Sub Broker.

5. Data is delivered to Data Consumers according to the available subscription (to those matching the Subscription Tokens included in the data streams received by the Context Exposition Pub/Sub Broker).
4.4.8. BUTLER User Profile Management

In the BUTLER architecture, the directory of users (User Directory) and their profile data is maintained on a centralized system: the **User Profile Manager**. It is a functional component that offers standard interfaces and provides users’ management functionalities to both users and applications. BUTLER Applications can use the User Profile Manager to read/modify generic user profile data, but also application-specific attributes that can be defined as attributes-value pairs within a user profile. In addition, the User Profile Manager provides mechanisms to connect a BUTLER user profile to external social networks.

Main system architecture for the User Profile Manager follows the typical three-tier pattern of enterprise application and User Profile Manager serves business and data logic (users’ profiles).

![User Profile Manager](image)

**Figure 68: User Profile Manager high level architecture.**

The diagram in the figure above shows the functional sub-components the User Profile Manager comprises and the connected external systems.

- **User Profile Manager RESTful APIs** interfaces can be accessed regardless the platform selected by applications developers (Android, iOS, JavaScript, etc.). They can be invoked by a SmartMobile running an application but it is also possible to build a custom view with a dedicated web-app like a web-portal. Presentation (graphical) logic is delegated to application developers.

- **External Systems Connectors** implement the interactions with other external systems like social networks and OpenID providers. The diagram shows connections to OpenID providers [28] which allow users to reuse an already existing OpenID identity to register a new profile in User Profile Manager. **External Social Networks** and **OpenID providers** can be used as external identity providers and exploited to connect a new user profile to an existing user social network identity. We refer to this capability as “social login”. User Profile Manager connects to External Social Networks also to create connections between user’s profile and his existing Social Network accounts to enable specific “social” features. We refer to this capability as “social connect” feature.

- **The Data model** component refers to user profile objects that map the user profile data model.

- **User profiles** are the data engines used to store data about users: profile data, user capabilities, and permissions related to connected social networks.

As detailed in section 4.5.3.9.2, the User Profile data model supports generic profile information data and also an extensible set of application-specific attributes that each application can read or modify. Moreover the adopted user profile model is compliant with de-facto OpenSocial standard [27], to ease interoperability with other applications external to the BUTLER platform.

The following interaction diagrams and descriptions show the interactions between client and the User Profile Manager component.
4.4.8.1.1. Register Account

The diagram shows the interaction and messages involved in the Register Account operation.

Figure 69: Register Account message sequence chart

1. Client issues a request to User Profile Manager. Parameters are passed in the request
2. User Profile Manager checks input parameters, in particular if username provided is already in the system
3. If user already exists, an error comes back to the client with a dedicated message
4. If user doesn’t exist, user is correctly created through data layer functions and a unique user identifier (UserID) is generated
5. User object is created
6. Success response is returned with the just created user data, in particular the unique UserID.

4.4.8.1.2. Get Account

This function allows retrieving information about a user profile. The functionality is protected by providing authorization credentials.
1. Client issues a request to User Profile Manager. Parameters are passed in the request, in particular the requested UserID.

2. User Profile Manager uses data layer functions to retrieve user profile.

3. If the user is not found, User Profile Manager notifies with an appropriate error message.

4. If the user is found, User Profile Manager creates data object from data layer.

5. Successful response is returned containing all available user profile information.

### 4.4.8.1.3. Update Account

This feature allows updating information about a user profile. The functionality is protected by providing authorization credentials.
Figure 71: Update Account message sequence chart

1. The Client issues a request. Parameters are passed in the request, in particular the update user profile data.
2. The User Profile Manager uses data layer functions to retrieve user profile.
3. If user is not found, the User Profile Manager notifies with an appropriate error message.
4. If user is found, the User Profile Manager retrieves data object representing the user profile from the data layer and modifies updated the profile with the new data.
5. The User Profile Manager retrieves the user updated profile with the support of data layer.
6. The User Profile Manager generates data model objects.
7. Success response is returned with the just updated user data.

4.4.8.1.4. Delete Account

This feature allows the removal of a user account from the system. The functionality is protected by providing authorization credentials (sequence diagram not shown).
4.5. ARCHITECTURE SPECIFICATION

4.5.1. Communications Layer

4.5.1.1. General

Figure 72 below shows the Communications architectural layer included within the overall BUTLER architecture. This layer is responsible for the end-to-end holistic communication, enabling the connection and the interoperability of the SmartObjects, SmartMobiles and SmartServers (based on IP connectivity) and for ensuring the management of the SmartObject. It also provides Security Services.

Physical objects are characterized by the use of heterogeneous standards, protocols and technologies (e.g. ZigBee, KNX, NFC, CoAP, etc.). These objects shall be able to communicate between each other and with other systems; to this purpose, an IoT Protocol Adapter functional group ensures the interoperability across different communication technologies, hiding the different protocols and standards used by individual objects. Among the others, the Communications Layer provides the capabilities for discovering BUTLER SmartObjects. A new SmartObject joining the BUTLER network needs to be discovered, enabling its remote identification; a repository for BUTLER SmartObjects should be also maintained, in fact, a Device Directory functional component is responsible for storing data and information about the SmartObjects. Since the communication of SmartObjects is often subject to RF interference and/or attenuation from obstacles that lead to temporary disconnections, a Network Monitoring functional component is used to continuously check the current status of the network and evaluate the related performances. Such performances are measured through quantifiable parameters, such as PLR (Packet Loss Rate) and latency. Moreover, the Communications Layer provides a functional component to manage the connectivity of users’ SmartObjects (which can use different technologies such as 3GPP, Wi-Fi, Ethernet) to the remote or local BUTLER servers. Similar to BUTLER SmartObjects, several functionalities are provided for maintaining directories of BUTLER user SmartObjects and BUTLER SmartServers. A Device Monitoring functional component provides a generic set of management operations which can be handled at physical device node.

For security purposes, the platform provides services for authentication and authorization. The authentication and authorization functionalities ensure that only identified and authorized SmartObjects (and SmartObjects) can join the BUTLER network in order to ensure a secure access control and that access to the resources provided by SmartObjects is secured as well.

The following functional components have been identified within the Communications Layer:

- **Network Monitoring**: This functional component monitors the performance and the network parameters of BUTLER SmartObjects. It is also a part of the System / Device Management Layer.
• **Device Monitoring**: It provides a basic set of device management operations to set several administrations and monitoring parameters used by device platforms. It is also a part of the System / Device Management Layer.

• **Device Discovery**: This functional component is responsible for discovering BUTLER SmartObjects.

• **Device Directory**: This functional component is responsible for maintaining repository for BUTLER SmartObjects.

• **Device Authentication**: This functional component ensures that only authenticated BUTLER SmartObjects can join the BUTLER network.

• **IoT Protocol Adapters**: An adaptation layer that hides the underlying technology heterogeneity and ensures the interoperability between different communication technologies.

• **Authorization Server**: This functional component is responsible for resource access control management and users authentication.

• **User Device Directory**: This functional component maintains a directory of BUTLER users and their associated devices.

• **User Authentication**: This functional component is responsible for the authentication of the users.

• **User Connectivity Manager**: This functional component manages the connectivity of the user, selecting the right connectivity according to the service needs.

• **Server Directory**: This functional component maintains a directory of BUTLER servers.

• **Server Authentication**: This functional component ensures that only authenticated servers can join the BUTLER network.

• **Server Connectivity Manager**: This functional component manages the connectivity of the servers in the BUTLER network.

4.5.1.2. Network Monitoring

4.5.1.2.1. Description

Network Monitoring (NM) is a functional component of the BUTLER Communications Layer that provides availability, utilization and overall performance of M2M local area networks (MANs). More specifically, the NM functional component will be implemented on the SmartObject Gateway to monitor the network parameters of the corresponding underlying network made of heterogeneous SmartObjects. In particular, the NM collects and analyses network traffic and then provides aggregated network statistics per SmartObject. Since the NM provides real time statistics about the network, it will help network administrators in meeting the agreed network service level and having better planning for the future.

The performance of a M2M network is measured through quantifiable parameters, or performance metrics that vary depending on the purpose and nature of the network itself. In particular, the following metrics are often considered important in the field of network monitoring:

• **Bandwidth**, which represents the rate of data transfer, measured in bit/s;

• **Network Capacity**, which defines the maximum amount of aggregated data that can circulate in the network per time unit, e.g. the ability of the network to collect as much data as possible;

• **Throughput**, the average rate of successful message delivery over a communication channel;

• **Latency**, i.e. the time from the source sending a packet to the destination receiving it.

• **Delay**, which measures the time necessary for a network to determine where to route the information and to transmit the packet completely, including the propagation delay to the recipient of the information;

• **Jitter**, variation in the time of arrival at the receiver of the information;

• **Packet delivery rate**, which is the percentage of the total number of transmissions that are successfully received by the recipient. In other works, this metrics aims to measure the network reliability by keeping track of the number of packets sent by the transmitter and those actually received by the recipient.
Two main monitoring methods can be adopted in order to make measurements on network traffic: active monitoring and passive monitoring.

**Active monitoring** consists of sending specific ad-hoc packets within the network on which to perform network related measurements. The active monitoring (enabled by commands such as ping and traceroute, see Table 10) is a kind of probe to be introduced into the network, reporting how many packets are lost, how long the network takes to deliver the packets, how the delay varies, the path taken by packets and so forth. The active method allows performing measures (for example latency between two nodes, the percentage of lost packets, bandwidth measurements, etc.) that would be more complicated to perform with a passive one. However, the active approach implies an unavoidable network overhead due to the injected probe packets which compete with user traffic.

**Passive monitoring** provides the analysis of traffic without sending additional packets into the network. This approach is a bit less accurate with respect to the active one but it does not interfere with the flow of network traffic. By using a passive approach, the measurements analysis can be performed only off-line and not during the data collection. This could make the post-processing time quite high.

A well-known monitoring method is based on the SNMP (Simple Network Management Protocol [23]). The SNMP is a network protocol that allows controlling and managing the functionalities of the elements belonging to a TCP/IP network. The SNMP architecture is based on the concepts of **Manager** and **Agent**. The main elements of SNMP are reported as follows:

- Network Management System (NMS or Manager);
- Agent (software that runs on managed nodes);
- Management Information Base (MIB);
- Network management protocol.
The **NMS** or Manager is typically a device (or more than one) that works as an interface between the network administrator and the system to be handled. In most cases, they are calculators that run management software (in the SNMP model, all the intelligence is maintained in the NMS).

The managed nodes are network devices, workstation, servers, software applications or any entity or node on which runs the **Agent** software. The agent is a process running on the various devices of the network that responds to request for information and executes operation imposed by the NMS. It can also communicate with the NMS asynchronously, generating urgent events. Each agent maintains a local database (**MIB**) of variables, which describes the state of the devices and that affects its operations.

According to the SNMP vision, each device maintains one or more variables that describe its state; these variables are called **objects**. The **MIB** contains the collection of all possible information (objects) and functionalities managed by the device. The manager contains the entire MIB database; the agent contains the part of the MIB that is of interest for that particular device.

The NMS interacts with the agent using the **SNMP protocol**. This protocol allows the NMS to request the status of the local objects of an Agent, and to modify them if necessary.

A plethora of network devices (all the TCP/IP devices) actually support the SNMP protocol, which is a simple protocol requiring a minimal set of resources for its efficient implementation. However, while SNMP proves to be extremely useful for network ‘management’ issues, active and passive approaches (introduced above) are more suitable for pure ‘monitoring’ purposes.

Since the active and passive monitoring methods are complementary, the proposal for BUTLER is to have a NM functional component based on both approaches. In fact, the foreseen NM functionalities aim to provide real-time network related statistics for each SmartObject.

In particular, the proposed NM functional component is composed of three main stages:

- **Traffic sniffing**: this is the passive part, performed by capturing BUTLER commands while traversing the internal data bus within the SmartObject Gateway;
- **Process and analysis**: this is the active part, performed using BUTLER *ping* and *traceroute* commands;
• **Statistic production**: this stage is responsible for providing aggregated network status statistics in order to expose them through BUTLER external APIs.

More specifically, a certain number of specific statistical parameters and measures will be monitored. In particular, the link utilization will be expressed by the following metrics:
- **Number of transmitted commands** \( (n_{Tx}) \): defined as the total number of BUTLER commands sent per SmartObject;
- **Number of application bytes** \( (n_{Bytes}) \): defined as the total number of BUTLER command bytes sent per SmartObject;

The link quality will be expressed by the metrics:
- **Latency**: expressed as a measure of the Round Trip Time (RTT);
- **Command loss ratio**: calculated as the number of commands lost in a specific interval of time;

The SmartObject Gateway commands will be:
- **BUTLER ping**, used to test the reachability of SmartObjects as well as to measure the corresponding latency. It is calculated as half of the RTT related to messages sent from the SmartObject Gateway to the SmartObjects;
- **BUTLER traceroute**, used for discovering the multi-hop path from the SmartObject Gateway to a specific SmartObject.

More details concerning the proposed Network Monitoring functional component are reported in [7].

### 4.5.1.2.2. Interaction with other functional components

This functional component interacts with the IoT Protocol Adapters functional component in order to allow the communication with underlying networks made of heterogeneous SmartObjects.

### 4.5.1.2.3. Operations

The Network Monitoring functional component provides the following operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ping</td>
<td>It is used to test the reachability of a SmartObject and to measure the round-trip time for messages sent from the originating SmartObject Gateway to a destination SmartObject.</td>
<td>SmartObjectID</td>
<td>echo response</td>
</tr>
<tr>
<td>Latency Statistic</td>
<td>It is used to evaluate the link quality which is between a SmartObject and the SmartObject Gateway.</td>
<td>SmartObjectID</td>
<td>Round Trip Time in milliseconds</td>
</tr>
<tr>
<td>Command Loss Ratio Statistic</td>
<td>It is used to evaluate the link quality which is between a SmartObject and the SmartObject Gateway.</td>
<td>SmartObjectID</td>
<td>percentage of commands loss in an interval of time.</td>
</tr>
<tr>
<td>Trace Route</td>
<td>It is used to calculate the route (path) from the SmartObject Gateway to a specific SmartObject.</td>
<td>SmartObjectID</td>
<td>list nodes that compose the path.</td>
</tr>
<tr>
<td>Number of transmitted commands</td>
<td>It is used to evaluate link utilization in a given interval of</td>
<td>SmartObjectID</td>
<td></td>
</tr>
</tbody>
</table>
Table 10: Network Monitoring supported operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| SET_PERIOD     | All the sensors connected to a node are scanned with a period of time that can be changed remotely. This period define also the frequency the sensor events will be notified to the user if he subscribed to periodic notifications. | • SmartObjectID (mandatory)  
• Period (mandatory)  |
| SET_LOCATION   | The localization of the node can be changed remotely at the physical device. | • SmartObjectID (mandatory)  
• Location (mandatory)  |
| SLEEP          | Sleep operation asks the devices to enter its Sleep Mode to save some battery. It takes as input the sleep duration of the device. While sleeping, devices stop polling its sensors and can’t receive or send data. | • SmartObjectID (mandatory)  
• Duration (mandatory)  
• Unit (mandatory)  |

Table 11: Device Management operations

To understand more the generic aspect of these operations, some examples showing the mapping to specific devices protocols are given.

In Zolertia implementation:

a) To modify the scanning period on a Zolertia node, send a COAP POST [41] request with the following URI:
   
   coap://[node_address]/admin/deviceManagement with period=new_period_value in the payload, where new_period_value is in seconds

b) To change remotely the location of a Zolertia node, send a COAP POST request with the following URI:
   
   coap://[node_address]/admin/deviceManagement with loc=new_location in the payload.

c) To ask a Zolertia node to enter its Sleep Mode, send a COAP GET request with the following URI:
   
   coap://[node_address]/admin/sleep?dur=sleep_duration&unit=sleep_unit, where sleep_duration
is an integer and `sleep_unit` can be `s` (for second), `m` (for minute) or `h` (for hour). For example, the request:
```
coop://[node_address]/admin/sleep?dur=180&unit=s
```
will ask the device to sleep for 180 seconds.

In Waspmote implementation: To ask a Waspmote node to enter its Sleep Mode, send a Transmit Request with 0x50 as ApplicationID, and `sleep#DD:HH:MM:SS` in the payload, where `DD:HH:MM:SS` is the value of the sleep duration. For example, the request `sleep#00:00:05:30` will ask the device to sleep for 5 minutes and 30 seconds.

4.5.1.4. Device Discovery

4.5.1.4.1. Description

This functional component is responsible for detecting the appearance of BUTLER-compliant SmartObjects into an underlying network. BUTLER SmartObjects are discovered in dynamic and transparent ways. Thus, SmartObjects removals are also detected by the discovery mechanisms.

Device Discovery mechanisms can be based on an infrastructure-less approach, where SmartObjects advertise their existence to their local network, or an infrastructure-based approach where a centralized entity handles the device discovery operations.

In the BUTLER SmartObject Gateway, Device Discovery is managed by various protocol bridges relying on different communication protocols (see section 4.5.1.8). For each discovered device, its associated bridge creates a BUTLER SmartObject, its BUTLER SmartObjectServices and Resources, that interact with the physical device using the underlying communication protocol stack. Figure 75 shows the Device Discovery process and the creation of configured BUTLER SmartObjectService instances using factories.
In addition, it is possible to provide a Device Discovery interface for foreign clients via the SmartObject Gateway, see [7] for more details. This lookup interface will use Device Discovery mechanism and it will return a SmartObject description (i.e., SmartObjectID and related factories) in alternative formats (e.g., JSON). It may also support advanced interfaces used for discovering which SmartObject that hosts a particular resource.

### 4.5.1.4.2. Interaction with other functional components

Discovery components are part of the southbound bridges (see sections 4.4.2.2 and 4.5.1.8). They communicate with physical devices using the underlying protocol stack. Moreover, they use known factories in order to create the appropriated SmartObject instances.

### 4.5.1.4.3. Operations

The Device Discovery component provides the following operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look for devices</td>
<td>Discover apparition of devices</td>
<td></td>
</tr>
<tr>
<td>Creation of BUTLER</td>
<td>Configure and instantiate BUTLER</td>
<td>Configuration (SmartObject properties)</td>
</tr>
<tr>
<td>SmartObjects</td>
<td>SmartObjects representing the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>discovered devices</td>
<td></td>
</tr>
<tr>
<td>Dispose of BUTLER</td>
<td>Disposal of BUTLER SmartObjects</td>
<td>SmartObjectID</td>
</tr>
<tr>
<td></td>
<td>from which devices have been</td>
<td></td>
</tr>
<tr>
<td></td>
<td>removed</td>
<td></td>
</tr>
</tbody>
</table>

Table 12: Device Discovery operations

### 4.5.1.5. Device Directory

#### 4.5.1.5.1. Description

This functional component is responsible for maintaining a registry of available BUTLER SmartObjects. In the BUTLER SmartObject Gateway, a BUTLER SmartObject instance created as the result of the Device Discovery process is automatically registered into the Device Directory. In a similar way, a BUTLER SmartObject instance disposed following the removal of its associated device is automatically unregistered of the Device Directory.

The Device Directory supports looking for device descriptions, as well as publishing, updating and removing device descriptions. Figure 76 shows some result examples given by the Device Directory to device lookups.

<table>
<thead>
<tr>
<th>Request</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>butler:devices</td>
<td>Butler Device Instances:</td>
</tr>
<tr>
<td></td>
<td>-- ID: TV-ABCD12345</td>
</tr>
<tr>
<td></td>
<td>-- ID: TV-ABCD67890</td>
</tr>
</tbody>
</table>

19 Currently, the OSGi registry is used as the BUTLER SmartObject Gateway Device Directory.
4.5.1.5.1. Operations

The device directory provides the following operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| Register BUTLER SmartObjects | Register a created SmartObject                     | • SmartObjectID (mandatory)  
|                         |                                                  | • SmartObject description (properties)        |
| Update BUTLER SmartObjects | Maintain an up-to-date register of SmartObjects | • SmartObjectID (mandatory)  
|                         |                                                  | • SmartObject description                    |
| Remove BUTLER SmartObjects | Remove a disposed SmartObject                     | • SmartObjectID (mandatory)                    |
| Discover BUTLER SmartObjects | Look for SmartObjects                             | -                                              |
| Get a BUTLER SmartObject description | Obtain the description of a given BUTLER SmartObject | • SmartObjectID (mandatory)                    |

Table 13: Device Directory operations

4.5.1.6. Device Authentication

4.5.1.6.1. Description

This functional component ensures that only authenticated SmartObjects can join to the BUTLER network. Device Authentication is usually a built-in functionality within the Authorization Server, according to section 4.4.3. However, it is possible to model it as a stand-alone functional component according to its specific functionalities.

4.5.1.6.2. Interaction with other functional components

The Device Authentication functional component will be used by end-users. The run-time authentication procedure is usually embedded within the overall authorization procedure.

4.5.1.6.3. Operations

The Device Authentication component provides the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Key</td>
<td>Register the key that will be used by an authorized resource hosted by a device (a SmartObject) in further authentication / authorization procedures. This operation is part of the overall Resource Registration operation within the ones supported by the Authorization Server</td>
<td>• Resource Key Material (it may be the object identifier, when the Factory Key schema is used, or a user-defined key)</td>
</tr>
</tbody>
</table>
Table 14: Device Authentication supported operations

4.5.1.7. Authorization Server

4.5.1.7.1. Description
The Authorization Server implements resource access controls management in direct resource consumption scenarios. It authenticates users and authorizes Resource Consumers to get a resource by issuing a resource-related access-token. That way it becomes a resource directory. Optionally, it may delegate the user authentication to the Authentication Server.

4.5.1.7.2. Interaction with other functional components
The Authorization Server functional component is used the entities and functional components that require to access resources in a direct way (i.e. not mediated) implementing end-to-end security. Therefore, users and applications are the usual clients of the Authorization Server functionalities.

4.5.1.7.3. Operations
The following operations are provided by the Authorization Server functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| Access-Token Request       | The application (Resource Consumer) requests an access-token for subsequently accessing a Resource. The operation is usually carried out through a user agent. | • Client identifier (OAuth)  
• Resource identifier (optional) |
|                            | The Authorization Server verifies that the application is registered in the Authorization Server (that is, that client credentials have been previously delivered to the application). An access-token and security material (authentication key, for authenticating the application, request) is sent back by the Authorization Server for the application to subsequently request a resource |
| User Authentication        | The user authenticates towards the Authorization Server, implicitly authorizing an application (Request Consumer) to ask for a resource access-token | • User credentials (username/password)  
• Client identifier (implicit) |
| Authorization Granting     | The user grants an application (Resource Consumer) to access a resource. An authorization code is sent back by the Authorization Server for the user to pass it on to the application | • Resource URL (identifier)  
• Client identifier (implicit) |
| Application Authentication | An application (Resource Consumer) requests an access-token for subsequently accessing a |
|                            | • Client-secret (OAuth)  
• Authorization code (obtained as result of the Authorization Granting message) |
Resource. Unlike the first type of access-token request message (Access-Token Request), the application authenticates instead of only identifying to the Authorization Server, by using the authorization code obtained previously.

| Resource Registration | Registration of a Resource by a User. | • Resource Name  
• Resource Semantic  
• Resource URL (identifier)  
• Resource Available Actions (GET/SET/..)  
• Resource Key Material (it may be the object identifier, when the Factory Key schema is used, or a user-defined key) |
| Permissions Management | The owner of a resource (a user) can grant or revoke access to one of his/her resources | • Username  
• Resource URL (identifier)  
• Operation (ADD/REMOVE) |
| Get End-to-End Security Encryption Keys | Upon the reception of a resource request issued by an application (Resource Consumer) the Resource Provider uses the information included in the request to request encryption keys to protect subsequent delivery of resource information to the Resource Consumer | • Request-response-encryption-key-identifier (previously delivered by the Authorization Server to the Resource Consumer when providing the Access-Token) |

**Table 15: Authorization Server supported operations**

### 4.5.1.8. IoT Protocol Adapters

#### 4.5.1.8.1. Description

#### 4.5.1.8.1.1. Overview

A plethora of physical objects (devices, sensors, actuators) are used within the BUTLER environment, each of them characterized by the use of heterogeneous standards, protocols and technologies (e.g. ZigBee, KNX, NFC, CoAP, etc.). In order to ensure the interoperability between different communication technologies, the IoT Protocol Adapter Functional Group\(^\text{20}\) (PA FG) provides a unified way to access the SmartObjects residing on a wireless sensor networks (WSNs), regardless of the connectivity technologies in which they are based on. In addition, the PA FG provides an IP connectivity which allows physical objects to communicate between them and with the BUTLER platform.

The PA FG is connected with dedicated physical interfaces, compatible with the physical layer of the specific sensor networks, one for each of the supported WSN technologies. Obviously the physical layer is only the starting point. A dedicated protocol stack implementation is required too, coupled with a dedicated bridge towards a common Device Access functional component providing a unified interface.

In summary, the IoT Protocol Adapter shall:

---

\(^{20}\) We talk about a Functional Group as different IoT Protocol Adapters will be present in order to support different communication technologies.
• Ensure an implementation of the specific technologies and communication protocols used by the various SmartObjects. In this way the IoT Protocol Adapter will be able to access and manage the various SmartObjects according to the specific protocol;
• Being able to ensure an IP connectivity to the SmartObjects, managing the network and node discovery mechanisms;
• Being able to access the resources offered by each SmartObject, exposing these resources via specific interfaces.

The IoT Protocol Adapter will be implemented within the SmartObject Gateway and, in order to achieve the above mentioned features, it will include the following three main functional subcomponents, which are showed in the block diagram below:

- **Device Access** functional component exposes a protocol agnostic interface towards internal SmartObject Gateway components and provides support for CRUD operations, subscription/notification, etc.
- **Bridges** adapt the specific underlying protocols with the BUTLER functionalities, which are defined in the Device Access interface.
- **Protocol Stacks** are modules that implement the specific communication protocol. These components will be able to handle device discovery and access the SmartObjects resources according to the specific protocol.

The rest of the section describes the IoT Protocol Adapter components starting from the upper layer, the Device Access functional component, passing from the Bridges’ set, ending to the of Protocol Stacks’ set.

### 4.5.1.8.2. Device Access

This functional subcomponent implements the northbound interface of IoT Protocol Adapter Functional Group. It exposes a protocol agnostic API towards higher level internal GW components. It provides support for:

- CRUD operations
- Subscription/notification
- Publish Resource capabilities in the resource tables defined within the SmartObject Gateway
It is based on the BUTLER Resource Model and shall adopt JSON Resource Schema. This schema based approach makes it possible to cover a wide range of applications spanning from home automation, to media, to health care.

The Device Access functional subcomponent dispatches the inbound messages request toward the appropriate southbound IoT Protocol Adapter interface and sends back to the requestor the resulting responses. This dispatching is based on the unique of the required resource. A resource is characterized by an identifier, a resource type, attributes and access methods (see Figure 21). The Resource Access Methods allow interacting with Resources. Access Methods are characterized by a type and a set of input parameters. Access Method types are as follow:

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

Table 16: Methods Description

4.5.1.8.3. Bridge

A Bridge is the glue between a Protocol Stack and the Device Access functional subcomponent. Via the Protocol Stack, it discovers SmartObjects, tracks events occurred on them and executes commands to actuate them. Therefore, a Bridge is necessary for each protocol that the BUTLER architecture will support. Whether the protocol is standardized or proprietary does not matter. As far as the relative Protocol Stack is available and the relative Bridge is implemented on top, the Device Access functional subcomponent is able to support the protocol and provide a unified way to access SmartObjects with the protocol. The Bridge is able to support:

- **Device Discovery.** When a new SmartObject is discovered and gets available, it shall create and register this SmartObject as a service within the IoT Protocol Adapter framework. When a previously discovered SmartObject gets unavailable, it shall unregister the corresponding service.

- **Device Measurement Update.** When a service parameter update occurs on a SmartObject, it shall update the corresponding variable on the SmartObject in the IoT Protocol Adapter framework and trigger update for the corresponding Service.

- **Device Actuation.** For each action in a service that a SmartObject supports, it should implement a protocol specific logic and put it as an action in the SmartObject service that is registered in the IoT Protocol Adapter framework.

- **Device Event and Data.** Its behavior is like an Event Producer and for each SmartObject connected it shall send events and data.

4.5.1.8.4. Protocol Stacks

A Protocol Stack subcomponent represents the implementation of a specific communication protocol. It handle device discovery and access to sensor and actuator resources in a protocol specific way.

This functional subcomponent communicates with the same protocol technology SmartObjects (southbound) and with the relative protocol bridge (northbound).

4.5.1.8.5. Interactions with other functional components

The IoT Protocol Adapter is strongly linked with other functional components defined within the BUTLER Communications Layer. In fact, the following functional components, Network Monitoring, Device Discovery, Device Directory and Device Authentication, exploit the functionalities offered by the IoT Protocol Adapter to interact with the SmartObjects.
4.5.1.8.6. Operations
As reported above, the main purpose of this module is to ensure the interoperability between SmartObjects that use different communication technologies, manage network and node discovery mechanisms and access the resources offered by each SmartObject. More relevant operations are listed in Table 16.

4.5.1.9. User Device Directory

4.5.1.9.1. Description
The User Device Directory functional component maintains a directory for the mapping of BUTLER users that are registered in the system and their associated devices that are connected to the BUTLER system (SmartMobiles). By using this functionality, other components can discover which user device the user is currently active on, or, vice-versa, given a user device that is performing some request, which user can be associated with this device. This component thus maintains the mapping between users and devices in relation to active communications.

The functional component must also provide the functionality to applications to find the best communication method to reach a user through its current active devices, in case several of them are active at the same time. For example, in case a BUTLER application wants to notify to some user that new data from an IoT sensor is available or that some application trigger has been activated, the application needs to find the best communication device to reach the user. A functionality provided by this component will return to the application, given an identifier of the target user, a list of identifiers of currently active devices that can be reached and are associated with the user. Other parameters can be associated with this information so that the application can perform the best decision on how to deliver the required notification to the final user.

Another functionality performed by this component is related to the possibility for other components to list users that are currently active and connected to the BUTLER system by means of at least one BUTLER device, so that an application can for example establish which users to monitor for contextual information according to some application preferences.

4.5.1.9.2. Interaction with other functional components
This functional component interacts with the User Profile Manager and Authorization Server functional components.

4.5.1.9.3. Operations
The User Device Directory functional component provides the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add User Device</td>
<td>Associate a user device (SmartMobile) to a registered BUTLER user</td>
<td>• UserID (identifier)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SmartObjectID (identifier)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Vendor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Release</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• (other device related data)</td>
</tr>
<tr>
<td>Set Active Device</td>
<td>Set a device associated with a BUTLER user as active for the user</td>
<td>• SmartObjectID (identifier)</td>
</tr>
<tr>
<td>Get User Devices</td>
<td>Returns a list of user devices associated with a BUTLER user</td>
<td>• UserID (identifier)</td>
</tr>
<tr>
<td>Get User Active</td>
<td>Returns a list of active devices for</td>
<td>• UserID (identifier)</td>
</tr>
</tbody>
</table>
### 4.5.1.10. User Authentication

#### 4.5.1.10.1. Description

This functional component ensures that only authenticated SmartObjects can join to the BUTLER network. BUTLER does not focuses however into user authentication as there are plenty of solutions and the focus of BUTLER is device authentication and authorization. As described in section 4.4.3, Authentication is usually performed by the Authorization Server. However, other functional components need also to guarantee that users get authenticated. To achieve this scenario, the BUTLER architecture introduces a specific User Authentication functional component that takes care of the authentication of users and uses identity federation procedures, such as those introduced by SAML 2.0 [25].

#### 4.5.1.10.2. Interactions with other functional components

The User Authentication functional component can be used by other functional components within the BUTLER architecture:
- Authorization Server: if delegation of authentication is used.
- Data Management and Marketplace Server: for the authentication of users.

#### 4.5.1.10.3. Operations

The User Authentication functional component provides the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| Authentication Request  | It allows other functional components to delegate user authentication to the User Authentication functional component and request the user to be authenticated | • Identifier of the functional component requesting the authentication.  
• Authentication procedure requested.  
**Return type:** authentication assertion containing the UserID. |
| Authentication           | Authentication of the user. This procedure is run after the User Authentication functional component receives an Authentication Request | • User credentials (depends on the authentication method)                   |
### 4.5.1.11. User Connectivity Manager

#### 4.5.1.11.1. Description

The BUTLER User Connectivity Manager is the component that is embedded within the SmartMobile platform that aims at selecting the right connectivity according to the service needs. Indeed, multiple network connections can be available on a device at a specific time (for instance 3/4G, Wi-Fi, and Bluetooth) and each application can have specific requirements in terms of connection (bandwidth, latency...). Therefore, the User Connectivity Manager will take care of selecting the right connection based on the application invoked by the user. This component will make its choices based on applications description that will be made available in the BUTLER applications repository.

The User Connectivity Manager will only provide means to select the network connectivity, whereas application level protocols will be defined by service developers, with one exception: communications between SmartMobile and core BUTLER SmartServers (e.g. not server made available by third parties using the full BUTLER platform) will rely on the HTTP protocol over SSL channels, to ensure privacy of data that will be exchange between clients and servers.

#### 4.5.1.11.2. Interactions with other functional components

The User Connectivity Manager will potentially interact with any BUTLER application (e.g. SmartMobile application) that will have specific needs in terms of network connectivity.

#### 4.5.1.11.3. Operations

The User Connectivity Manager provides the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get list of available connection mode</td>
<td>Get the full list of available connection mode (e.g. Wi-Fi, Bluetooth...).</td>
<td>None</td>
</tr>
<tr>
<td>Set a specific connection mode</td>
<td>Authentication of the user. This procedure is run after the User Authentication functional component receives an Authentication Request</td>
<td>Connection mode name</td>
</tr>
</tbody>
</table>

#### 4.5.1.12. Server Directory

#### 4.5.1.12.1. Description

The Server Directory is the functional component in charge of maintaining a directory of BUTLER servers. A server is taken as a BUTLER server if it provides the functionalities of at least one of the modules referred in the BUTLER architecture. Those functionalities can be called BUTLER infrastructure services. The directory registers all the relevant information from each server.

When a BUTLER server candidate owner wants to connect it to the BUTLER network, s/he has to register it in the Server Directory. The first step is to get access to the Server Directory through the BUTLER community. Once his/her access has been granted, s/he will have to fill a registration form providing information regarding the server. It includes an infrastructure services report. After sending the request, the authentication step may start. That process includes a connectivity check and an authorization token delivery. After that, the BUTLER server candidate will be included into the Server Directory.

---

© BUTLER – Page 111/191
4.5.1.12.2. Interactions with other fictional components
The Server Directory functional component can be used by any other functional component within the BUTLER architecture able to un/register itself and to request lists of BUTLER servers in order to locate BUTLER infrastructure services. But there are specifically two components that are going to interact frequently with the Server Directory along the registration processes. The Server Connectivity Manager has to verify that the offered infrastructures services by the new BUTLER server candidate are 100% BUTLER compliant and are up and running by executing a connectivity test. Then, an authorization token request to the Server Authentication will be performed, so when the new BUTLER server gets it, it will be allowed to join the BUTLER network.

4.5.1.12.3. Operations
The Server Directory functional component provides the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registration Request</td>
<td>To start the registration process of a BUTLER server candidate. This operation will be available through a web portal and it is asynchronous because the registration process may take an undetermined amount of time</td>
<td>Owner, geographical location, modules’ functionalities supported, IP address, endpoints</td>
<td><strong>Return value</strong>: authorization token</td>
</tr>
<tr>
<td>Unregistration Request</td>
<td>To unsubscribe a BUTLER server from the Server Directory</td>
<td>Owner, IP address, authorization token</td>
<td><strong>Return value</strong>: true/false</td>
</tr>
<tr>
<td>List By Owner</td>
<td>To obtain a list of BUTLER servers that belong to a specific owner</td>
<td>Owner, authorization token</td>
<td><strong>Return value</strong>: list of BUTLER servers</td>
</tr>
<tr>
<td>List By Location</td>
<td>To obtain a list of BUTLER servers that are set in a specific location</td>
<td>Geographical location, authorization token</td>
<td><strong>Return value</strong>: list of BUTLER servers</td>
</tr>
<tr>
<td>List By Services</td>
<td>To obtain a list of BUTLER servers that support a specific list of infrastructure services</td>
<td>Modules’ functionalities list, authorization token</td>
<td><strong>Return value</strong>: list of BUTLER servers</td>
</tr>
</tbody>
</table>

Table 20: Server Directory supported operations

4.5.1.13. Server Authentication

4.5.1.13.1. Description
This functional component ensures that only authenticated servers can join to the BUTLER network. During a new BUTLER candidate registration process, the Server Authentication can be invoked by the aforementioned Server Directory in order to provide an authorization token. From that moment on, any BUTLER service can try to connect to the new BUTLER server. That action implies an authorization step between the server and the clients, involving the token the Server Authentication provided before. Clients will have to request a token too and compare it with the one hosted by the new BUTLER server.

4.5.1.13.2. Interactions with other fictional components
The Server Authentication functional component can be used by any functional component within the BUTLER architecture that are eventually involved in authorization tokens issues. Since multiple BUTLER services request them, the usage of this functional component will be very frequent.
4.5.1.13.3. Operations
The Server Directory functional component provides the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Authorization Token</td>
<td>To register a new BUTLER server candidate in the Server Directory. The generated authorization token will be sent to the indicated BUTLER server</td>
<td>Owner, IP address</td>
<td>true/false</td>
</tr>
<tr>
<td>Revoke Authorization Token</td>
<td>To unsubscribe a BUTLER server from the Server Directory. From the moment this operation is invoked on, the suitable authorization token will not be valid anymore</td>
<td>Owner, IP address</td>
<td>true/false</td>
</tr>
<tr>
<td>Validate Authorization Token</td>
<td>To check whether an authorization token provided by a BUTLER service owner is in force</td>
<td>Authorization token</td>
<td>true/false</td>
</tr>
</tbody>
</table>

Table 21: Server Authentication supported operations

4.5.1.14. Server Connectivity Manager

4.5.1.14.1. Description
The Server Connectivity Manager is responsible for keeping up to date the information registered in the Server Directory regarding the availability of functionalities provided by the BUTLER servers, in other words, it checks whether each server has its BUTLER infrastructure services up and running. This functionality can be invoked during a BUTLER server candidate registration process as well.

All BUTLER servers listed in the Server Directory will offer service endpoints that can be consumed by other servers or client applications using IP connectivity. Every BUTLER server registered in the Server Directory must authorize the Server Connectivity Manager to connect in order to test its infrastructure services periodically.

The Server Connectivity Manager receives the necessary information from the Server Directory in order to establish a connection with BUTLER servers. If it is able to connect successfully, the next step is to check one by one if all the offered infrastructure services are deployed properly and work correctly.

After that, the Server Connectivity Manager sends an answer to the Server Directory. The latter will include or update, depending on the case, according to the report content, the BUTLER server information and its infrastructure services in a public BUTLER directory of servers.

If the Server Connectivity Manager detects any anomaly in a BUTLER server candidate during a registration process, it will be interrupted by the Server Directory.

Periodically, at Server Directory request, the Server Connectivity Manager will connect to every registered server and check the availability of the infrastructure services.

4.5.1.14.2. Interactions with other fictional components
The Server Connectivity Manager functional component can be used only by one functional component within the BUTLER architecture: the aforementioned Server Directory. It requests periodically a
connectivity test for every already registered BUTLER servers in order to update their current status, or during a new BUTLER server candidate registration process.

4.5.1.14.3. Operations

The Server Connectivity Manager functional component provides the following operation:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
<th>Return value: true/false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity Test</td>
<td>To check whether the infrastructure services provided by a registered BUTLER server are up and running properly</td>
<td>BUTLER server endpoint, type of infrastructure service, authorization token</td>
<td></td>
</tr>
</tbody>
</table>

Table 22: Server Connectivity Manager supported operations

4.5.2. Data/Context Management Layer

4.5.2.1. General

In Figure 78 the data/context management architectural layer is represented. This layer in the overall architecture is responsible for all data- or context-related functionalities, including collection and capturing of different types of data, persistent storage and processing of data as events, either simple or complex.

This layer specification includes several functional components and the required data models used by the components. Data models are of different types: context, which represents highly dynamic data, or more static that can be profile/user or device related. A particular type of user data is about the behavior of the user that can be inferred by sensor and contextual information. Another type of data is also represented by events, and by resources.

Functional components in this layer are dedicated to Collecting and Actuating on both user and SmartObject data. In addition, resources must be made accessible by any application or services, and a Resource Access functional component is required to provide a directory of available IoT resources like gateways and SmartObjects.

Both Simple and Complex Event Processing functionalities are represented: depending on specific IoT service or scenario, application logics may want to compute new data based on data events and appropriate rules: new data could be in turn other data events or new contextual information. Location and User Profile data are also part of this layer and two specific functional components are defined here in order to manage these data. By considering the complexity of user events, context and profile data (like: location and movements, preferences, actions, etc.) a specific functional model for Behavior Capturing can produce higher level behavior understanding of the user, in terms of current or expected activities and situations. A generic functionality is represented by Persistent Storage which gives the possibility to applications to store data or context either locally to a mobile device, by a local server of in the cloud for additional replication and availability.
Contextual information related to user, SmartObjects or resources and behavioral information must be provided to interested applications by a Context/Behavior Information Provider functionality, which used a specific context data model to represent context information uniformly.

The Data / Context Management Layer provides also the means for enabling the management and acquisition of data sources: Data Source Management and Data Purchase are enabled through a Marketplace that enables a richer experience from the service developer point of view in close cooperation with the Data Discovery functionality provided in the Services Layer. Service developers are also able to discover entities representing physical entities and manage the Context associated to each of said entities. The functional components in charge of providing management capabilities to the service developer and to the owner of SmartObjects belong also to the System / Device Management Layer and are described there.

Finally since IoT applications may be interested in receiving notifications about changes in context data or SmartObject data, a Generic Notification Mechanism is included in this layer and its relative data model. In the following sub-sections all these functional components and related data models are described in greater detail.

The following functional components have been identified within the Data/Context Management Layer:

- **Generic Notification/Actuation Mechanism**: To provide a lightweight mechanism for the delivery of notifications to SmartMobiles. To provide a generic mechanism to perform actions on devices
- **Persistent Storage**: To keep a cached copy of information generated by SmartObjects within a SmartServer instance
- **Context/Behavior Information Provider**: To provide context and behavior information about users, their devices and the applications they use
- **Resource Access**: To access resource data in a uniform way independently from the type of resource
- **Simple Event Processing**: To process data events from SmartObjects using simple rules and generate new data or trigger applications
- **Complex Event Processing**: To analyze event data in real-time according to complex application rules and generate new data or events
- **Localization Manager**: To compute higher level location data from raging data using different location technologies provided by SmartObjects. To publish high level location data as context information
- **User Behavior Capture**: To collect sensor events, learn and infer high-level user activities such as current physical activities of the user
- **User Profile Manager**: To manage user profile data and user specific application data (it can connect also the user profile to external social networks). It is also a part of the System / Device Management Layer.
- **Device Data Collector / Actuator**: To collect data from SmartObjects.
- **User Portal**: It provides functionalities for the users of the BUTLER system: the Device Owners and the Service Developers. Through the User Portal, the former can manage data sources, data offerings and virtual entities. The latter can purchase data offerings and define contexts associated to entities. It is also a part of the System / Device Management Layer (and is described in section 4.5.4).
  - **Data Management and Marketplace Portal**: Management and purchase of data offerings.
  - **Context Management Portal**: Definition of context associated to virtual entities.
4.5.2.2. Complex Event Processing

4.5.2.2.1. Description

BUTLER introduces a Complex Event Processing engine as one of the core elements of the Data/Context Management layer. An architectural description of said functional component is already anticipated in FIWARE [18].

Complex Event Processing is a technology devised to analyze event data in real-time in order to generate immediate insights and enable instant response to changing conditions without requiring to store data to subsequently analyze it and make a decision. Within BUTLER, Complex Event Processing is used in a variety of scenarios, usually to handle data streams where the data items they comprise are usually generated by SmartObjects.

From a theoretical point of view two different types of entities are connected to a CEP engine in the traffic plane: Event Producers and Event Consumers. Event Producers provide events in a so-called “push mode”. That is, it is the Event Producer the entity pushing events into the CEP engine. On the other hand, event consumers are, for instance, brokers that forward the events they receive to all interested applications based on a publication/subscription model.

Besides the entities that connect to the CEP engine in the traffic plane (the abovementioned Event Producers and Event Consumers), other entities connect to the CEP engine in the control plane. Said entities are responsible for determining the way the CEP engine behaves in the scenarios it is being used. There are different ways to approach the creation of CEP applications (also known as event processing logic or applications) that determine the way the CEP engine behaves. Such ways are usually dependent on the type of engine being used, but generally speaking, said CEP applications are specified using a rule-oriented language that is inspired by the ECA (Event-Condition-Action) concept and may better be described as Pattern-Condition-Action. Rules in one of said languages will consist of three parts:

- A pattern detection that makes a rule of relevance.
- A set of conditions (logical tests) formulated on events as well as external data.
- A set of actions to be carried out when all the established conditions are satisfied.

Pattern detection is the phase where the patterns over a specific class of events (usually within an event processing context, which can be a time window or segmentation) are programmed. If the rules describing the pattern are matched, then a set of actions are executed. Examples for such patterns (described by rules) are:

- **Sequence**, meaning events need to occur in a specified order for the pattern to be matched.
- **Aggregate**, compute some aggregation functions on a set of incoming events.
- **Absent**, meaning no event holding some condition arrived within the time window for the pattern to match.
- **All**, meaning that all the events specified should arrive for the pattern to match.

Conditions are optional and may be added to a given pattern, but regardless of the additional conditions, when a pattern is detected and the rest of optional conditions are matched, an action or set of actions are executed. Said actions specify what should be done when a rule is detected. Examples of what can be done are the forwarding of the events to Event Consumers, creation of new events (new data streams, but combination of information the incoming events carry) and forwarding to Event Consumers, generation of alarms... These actions definitions include the parameters needed for their execution.

CEP engines are thus powerful tools to handle large amounts of information generated by SmartObjects and therefore extract insights, filter incoming data streams, generate new output streams by combination of the inputs, detect meaningful events in order to trigger alarms and so on. However, the way a CEP
engine works is always determined by the event processing logic it is applied. A fully open approach would
allow the consumers of the CEP functionality to freely define the logic they wish to implement and feed in
runtime the CEP engine with said event processing logic. Such an approach would turn the CEP engine into
a general-purpose enabler that could be invoked by any interested party. However, it has not been the
approach taken in BUTLER. We have chosen to define a limited set of CEP applications depending on the
actual requirements of each scenario. That way, for every scenario a specific logic is executed by the CEP
engine. Of course it will be possible for the customers of the CEP functionality (for example, see the high-
level descriptions in sections 4.4.6 or 4.4.7) to introduce the arguments for the conditions the CEP engine
must execute (for example, given an event processing logic defining how to issue an alarm when a
threshold is crossed, it could be possible to define different values for said threshold).

There are different ways of creating CEP applications. One of them is using a high-level scripting language,
usually defined by the vendor of the CEP engine, which requires an interpreter provided by the CEP engine.
This kind of scripting languages, as provided by vendors, is usually optimized for CEP tasks providing native
methods to carry out said tasks. The second approach is to use a general purpose language like C++ or Java.

As a summary of the previous paragraphs at least three interfaces are provided by the CEP engine (there
can be other but are not relevant for the operation of BUTLER):

- Event Input: collection of sensor data provided by SmartObjects.
- Event Output: Data streams created by the CEP engine as a result of the application of specific
  event processing logic.
- Definitions/instructions Deployment: interface for the deployment of the event processing logic
  that is needed for the specific scenario.

In BUTLER, the Complex Event Processing engine is used in different ways. With regard to some
requirements, the Complex Event Processing engine is used as is (with a specific set of applications). This
functional component is used, for instance, to support the Data Discovery and Data Marketplace
functionality. In this scenario, Event Producers are SmartObjects (through the Device Data Collector /
Actuator functional component) while Event Consumers are the BUTLER applications that receive data
items through the Resource Exposition Pub/Sub Broker and the Persistent Storage functional component.
However, it can be used to build or to play the role of other functional components such as the Context /
Behavior Information Provider (see sections 4.4.7 and 4.5.2.11), the User Behavior Capture (see section
4.5.2.4) or the Simple Event Processing functional component (see section 4.5.2.7).

4.5.2.2.2. Interaction with other functional components
The Complex Event Processing engine is heavily used by other functional components within BUTLER. In
some of them it is used explicitly. In other, it is simple a component of another functional component (that
is, a functional component that uses event processing features to provide its functionality).

The following functional relate explicitly to the Complex Event Processing engine:

- Device Data Collector / Actuator
- Data Management and Marketplace Portal
- Context Management Portal
- Context / Behavior Information Provider
- Persistent Storage
- Resource Exposition (Publication/Subscription Broker)

The following functional components use a Complex Event Processing engine as component:

- Energy Services
- User Behavior Capture
4.5.2.2.3. Operations

The following operations are provided by the Complex Event Processing functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Input</td>
<td>Collection of sensor data provided by SmartObjects through the Device Data Collector / Actuator</td>
<td>Input data streams must contain the information needed to identify the streams and to apply the relevant business logic.</td>
</tr>
<tr>
<td>Event Output</td>
<td>New data streams created by the CEP engine</td>
<td>The format of the output data streams will depend on the specific CEP application. It may include the identifiers the receivers of this data streams will need.</td>
</tr>
</tbody>
</table>
| Definitions/instructions Deployment | Deployment of the actual event processing logic needed to implement the desired CEP functionality | • Set of rules describing the pattern to detect and the actions to execute should the pattern be detected.  
• Set of arguments the abovementioned rules need to work (thresholds, for instance)  
• Set of identifiers the output streams must carry out. |

Table 23: Complex Event Processing engine supported operations

4.5.2.3. Persistent Storage

4.5.2.3.1. Description

This functional component aims to keep a cached copy of information generated by SmartObjects within a SmartServer instance. Such information could be accessed subsequently for Service Developers wishing to create new services based on SmartObject cached information (the other possibility available to Service Developers is to subscribe to updates on information being dynamically generated by SmartObjects and therefore not stored). The BUTLER Data Storage functional component relies on NoSQL technology and supports operations for storing measures coming from SmartObjects and for reading such information subsequently.

Among the different types of NoSQL databases, a Document Store has been chosen. Said type of NoSQL databases works with semi-structured data that contain a collection of key-value pairs. Every measurement from SmartObjects (from their Resources) is modeled as a document. Collections group the measured provided by every Resource.

Every document has the following terms at least:

• id. A 64-bit integer. It identifies the Resource the document belongs to.
• location. The physical coordinates of the SmartObject where the Resource is hosted. If dynamic, it records the location of the SmartObject at the time where the measure was created.
• timestamp. A 64-bit integer defining the time where the measure was created.
• data. A 32-bit integer with the actual value of the measure.
• units. A string stating the units of the data value.

4.5.2.3.2. Interaction with other functional components

The Persistent Storage functional component is used from the Complex Event Processing engine (responsible for determining which data items have to be cached) and the Resource Exposition functional component (through which data consumers access said data).
4.5.2.3.3. Operations

The following operations are provided by the Data Storage functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Collection</td>
<td>Creation of a new collection (equivalent to a relational table) for storing the measures from a given Resource</td>
<td>• ResourceID (mandatory)</td>
</tr>
<tr>
<td>Insert Document</td>
<td>Storage of a new measurement</td>
<td>• ResourceID (mandatory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location of the resource (optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Timestamp (optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Measure value (mandatory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Unit (optional)</td>
</tr>
<tr>
<td>Find Documents</td>
<td>Retrieval of measured stored in a given collection</td>
<td>• ResourceID (mandatory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Search criteria (based on timestamp, location or value)</td>
</tr>
</tbody>
</table>

Return value: list of documents

Table 24: Persistent Storage supported operations

4.5.2.4. User Behavior Capture

4.5.2.4.1. Description

User location and activities have become indisputable key enablers for context-aware applications. The ‘user behavior capturing’ functional component in the Data / Context Management Layer will handle the collection of sensor events, learning and inference of high-level user activities such as current physical activities of the user (say walking or running) and cooking in the kitchen. This functional component provides a specific functionality that goes beyond the general-purpose Context / Behavior Information Provider. Therefore, a distinct functional component is needed.

Figure 79 shows the interaction pattern of the ‘user behavior capturing’ functional component with other functional components in the context management layer of the SmartServer. The heterogeneous inputs to the user behavior capturing module vary both in their richness and dynamicity. For example, the functional component relies on the inputs from static user profile manager as well as more dynamic and real-time user generated data (say accelerometer readings). Similarly, it takes input from simple geo-location information providers (say GPS) as well as rich context information providers such as semantic location engine.
Figure 79: Interaction between the user behavior capturing module and other functional component in the context management layer

The output of the User Behavior Capture functional component is published through both the Context Exposition and the Generic Notification / Actuation functional component. Also, the output can be stored in the Persistent Storage functional component for future look backs and logging purposes.

Figure 80: A distributed architecture for behavior awareness

In general, the behavior awareness system is a highly distributed event based system. But depending on the requirements of the use case, it can be implemented as a centralized system or as a distributed one as shown in Figure 80. It is a scalable event-based stream mining architecture that integrates and exposes well-known software building blocks for event processing (Esper [31]), machine learning (Weka [32]) and
knowledge representation (Parliament [33]) as RESTful services. The most up-to-date information about this framework can be found at [35]. The key features include:

- **Feature extraction**: Convert raw low-level data and events into features that are more meaningful for comparison
- **Information fusion**: Aggregate data from different sources to increase the confidence in the quality of the inferred information
- **Domain knowledge**: Leverage background information to narrow down likely activities
- **Probabilistic correlations**: Identify frequent co-occurrences in event streams to derive event patterns of interest

The overall flow for using this architecture is to (1) define complex patterns of events with the specific query language provided by the Complex Event Processing engine, (2) wrap semantic and machine learning classification as custom operators from the Complex Event Processing engine, (3) use complex events for training and classification purposes and (4) add statements and listeners to subscribe to complex events of interest. The objective of this architecture is to offer architectural building blocks for (a) direct activity recognition using sensor events directly to detect actions with complex event processing, and (b) indirect activity recognition using linked concepts to recognize activities through semantic and spatio-temporal reasoning.

### 4.5.2.4.1.1. Direct activity recognition with complex event processing

We will use step counting as a simple motion activity recognition use case, using the accelerometer sensor of a smart phone producing events at 50Hz as input. See Figure 81 and Figure 82.

![Figure 81: Step counting with complex event processing](image)

This is a brief overview of the components involved:

- **AccelerometerEvent**: This component produces a continuous stream of acceleration data, consisting of X, Y and Z values arriving at a certain sampling rate.
- **MagnitudeEvent**: In many cases, we do not know how the accelerometer is oriented at the offset of the motion activity. Furthermore, the orientation of the sensor is subject to change while moving around. Therefore, we carry out the signal analysis on the overall magnitude of the acceleration signal.
- **MovingAverageEvent**: For mobility tracking, e.g. walking or running, we are mainly interested in acceleration peaks that arrive at a frequency of maximum 5Hz (i.e. max 5 steps per second). For this purpose, we use a ‘moving average’ component as a simple low-pass filter to remove high-frequency noise.
• **MinMaxEvent**: A single step is characterized by a pattern of several maxima and minima in the time domain of the acceleration signal. This component extracts these features in the signal for further analysis.

• **StepDeltaEvent**: As not every minimum or maximum corresponds with a step, this component identifies the correct peak for every step in order to correctly count the number of steps and to differentiate between standing still, walking and running (i.e. the peak rate).

Not shown here are other complex events we use e.g. for fall detection:

• **HighpassFilterEvent**: For fall detection, we are interested in sudden and high-frequency changes of the acceleration signal, both in amplitude and orientation. This component implements a high-pass Finite Impulse Response (FIR) filter to extract these features.

• **FallEvent**: This component analyzes the signal magnitude area (SMA) of the high-frequency part of the acceleration signal, and identifies a fall if this feature passes a certain threshold.

![Sequence diagram of event-based motion activity recognition](image)

**Figure 82**: Sequence diagram of event-based motion activity recognition

This is the overall event registration and processing flow:

1. Register the source input event 'AccelerometerEvent' with x, y, z and timestamp attributes
2. Define all the intermediate events ‘MagnitudeEvent’, ‘MovingAverageEvent’, etc. For example, the magnitude event would copy the timestamp from the accelerometer event, and computes the magnitude as \(\sqrt{x^2+y^2+z^2}\).
3. Subscribe to a complex event of interest (e.g. 'StepDeltaEvent').
4. Send the raw accelerometer events.
5. Compute, process and trigger the intermediate events.
6. Notify the application in case a step event was recognized.
7. Notify the application in case a fall event was recognized.
4.5.2.4.1.2. Indirect activity recognition with location-based activity classification

Beyond matching patterns of events and feature extraction, our architecture can also leverage background knowledge stored in a semantic database to increase the meaningfulness of an event. For semantic and spatio-temporal reasoning, it uses a GeoSPARQL [33] enabled storage backend. The benefits of such a building block are manifold:

- Describe the spatial characteristics of different locations in your environment.
- Use the W3C SSN ontology [36] to describe the sensors and their position.
- Translate positions in coordinates into semantic locations (e.g. Bedroom).
- Semantically link locations with activities (e.g. Sleeping in a Bedroom).

This background information can be semantically modeled with ontologies as follows:

```
ex:Room a owl:Class; rdfs:subClassOf geo:Feature
ex:LivingRoom a ex:Room; rdfs:label "Living Room"; geo:hasGeometry ex:GeoLivingRoom
ex:GeoLivingRoom a sf:Polygon ; geo:asWKT "POLYGON ((290 600, 580 600, 580 700, 900 700, 900 260, 290 260, 290 600))"^^sf:wktLiteral .
ex:activity a owl:DatatypeProperty; rdfs:domain ex:Room; rdfs:range xsd:string
ex:LivingRoom ex:activity "Watch TV"
ex:LivingRoom ex:activity "Play Game"
ex:MasterBedroom ex:activity "Sleep"
```

As locations and activities are not deterministically correlated, our architecture embeds Weka’s learning and classification algorithms (e.g. Bayesian networks or J48 decision trees) to probability infer these spatio-temporal dependencies. Today's IoT environments have many-to-many mappings between multiple smart applications, inferred high-level contexts and their context sources. Moreover, given the mobile capabilities of the user and their SmartObjects, the operating conditions are continuously changing. This gives rise to non-trivial challenges such as sensor ambiguities (e.g., sensor failures/availability and missing data). Hence we have implemented a modular, distributed Bayesian framework suitable for dealing with dynamic and heterogeneous context sources and their uncertain availability. Figure 83 shows the HARD-BN (A Heterarchichal, Autonomic, Recursive and Distributed Bayesian Network) framework which is a run-time probabilistic model of user behaviors for simultaneously inferring multiple user contexts. User activities are modeled as high-level context nodes ($X_a$) and are inferred from multiple sensors ($Z_j$). The estimates from each of the high-level contexts $X_a$ will be exposed using APIs.
Figure 83: HARD-BN: A Heterarchichal, Autonomic, Recursive and Distributed Bayesian Network framework for flexible and scalable activity recognition

Technically the proposed framework is a collection of Bayesian networks where each individual Bayesian network models a unique high-level context. For the ease of implementation each Bayesian network is modeled with naive Bayes assumption. Figure 83 shows a typical instance of the proposed framework where different contexts are modeled as separate individual Bayesian networks BN_i (enclosed in a box). The parent node in each individual Bayesian network is the high-level context node to be inferred (e.g., physical activities or semantic locations of the user) and their respective children nodes correspond to the sensors used to infer them (e.g., accelerometer or GPS data). Note that the double headed arrows in Figure 83 is to illustrate the mutual influence of random variables (contexts) on each other and are implemented as separate Bayesian networks for each of the context nodes.

Learning in the proposed network is simpler compared to a generic Bayesian networks because their naive assumption guarantees fixed network structure. Only the conditional probability density of the nodes representing high-level context have to be learned. First, the prior probability distributions of the high-level contexts are safely assumed to follow a beta distribution with equal prior probability distribution. Then, the likelihoods of the high-level contexts are estimated separately conditioned on (1) the sensor nodes and (2) other dependent high-level context variables and can be easily parallelized as concurrent tasks. Nevertheless, learning the conditional probability densities of the high-level contexts conditioned on other high-level contexts are dependent on the learning in other individual Bayesian networks conditioned on sensor inputs requiring synchronization of those tasks.

Inference in the proposed framework is done in bootstrap mode by recursively executing 2 steps. In the first step, an estimate of each high-level context nodes is generated by combining objective prior probabilities and evidence presented by the other high-level contexts. In the second step, the posterior estimated from the previous step is used as an informed prior to determine the most probable value for each of the high-level contexts. The objective of the first step is to utilize the global influence of a context on other co-related context information for improved prediction. The second step acts as a correction step where the informed prior values are adjusted based on the evidence from the local observations of the low-level sensor inputs.
4.5.2.4.1. Interaction with other functional components
This functional component interacts with various other functional components usually hosted by SmartServers:

- **Localization Manager**: get x, y coordinates and reference system of user/SmartObject location
- **Context Exposition (Pub/Sub Broker)**: get ground truth data for training purposes and push data in
- **Complex Event Processing Engine**: translate low level events into higher-level ones, match event patterns of interest.

4.5.2.4.2. Operations
The following operations are provided by the User Behavior Capture functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Activity Event Pattern</td>
<td>Recognition of an event as a sequence or pattern of events</td>
<td>• A continuous query (matching an event stream of interest)</td>
</tr>
<tr>
<td>Register Semantic Spatial Map</td>
<td>Storage of a semantically annotated spatial layout of the environment to correlate relevant activities</td>
<td>• A collection of semantically tagged regions and their spatial characteristics defined as a polygon.</td>
</tr>
<tr>
<td>Get Activity</td>
<td>Retrieve a list of likely current activities of an individual, each activity complemented with its corresponding confidence value</td>
<td>• UserID (mandatory) • Time and location (optional) • Confidence threshold (optional)</td>
</tr>
</tbody>
</table>

Table 25: User Behavior Capture supported operations

4.5.2.5. Generic Notification/Actuation

4.5.2.5.1. Description
This functional component aims to provide a lightweight mechanism for the delivery of notifications to SmartMobiles.

4.5.2.5.2. Interaction with other functional components
This functional component interacts with the User Device Directory to determine where to push a notification.

4.5.2.5.1. Operations
The following operations are provided by the Generic Notification / Actuation functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliver Notification</td>
<td>Deliver a notification to a user managing a SmartMobile</td>
<td>• UserID • Text to deliver</td>
</tr>
</tbody>
</table>

Table 26: Generic Notification/Actuation supported operations

4.5.2.6. Resource Access

4.5.2.6.1. Description
Resource Access allows interacting with resources exposed by services and devices. Access to resources is achieved using the Device Data Collector / Actuator functional component.
4.5.2.6.2. Operations

Access and manipulation of the resources exposed by a SmartObject is performed mainly using the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>Gets the data of the resource</td>
<td>-</td>
</tr>
</tbody>
</table>
| set       | Sets the given data value and set of metadata to the resource | • Object value (mandatory)  
• Set of metadata |
| act       | Executes the resource with the given input parameters | • Parameters |
| subscribe | Subscribes the client to the resource | • Subscriber listener (mandatory)  
• Set of conditions  
• Period |
| unsubscribe| Cancels the subscription       | • SubscriptionID (mandatory) |

Table 27 Resource Access supported operations

The next table presents some examples of SmartObjects, Services, Resources and resource access using the resource access operations.

<table>
<thead>
<tr>
<th>SmartObject</th>
<th>Services</th>
<th>Resources</th>
<th>Access to resources</th>
</tr>
</thead>
</table>
| TV              | PlayerService | • volume  
• media  
• play  
• stop  
... | volume.set(volumeLevel)  
media.get()  
play.act(mediaID)  
stop.act() |
| PIR sensor      | PresenceService | someone_detected | someone_detected.subscribe() |
| Ultrasound sensor | DistanceService | distance         | distance.subscribe(distance<50) |
| Slider sensor   | SliderService  | slider_position  | slider_position.get()  
slider_position.subscribe() |

Table 28: Devices Example

4.5.2.7. Simple Event Processing

4.5.2.7.1. Description

BUTLER provides some event processing functionalities. Complex Event Processing (see section 4.5.2.2) can evaluate different sources of events for a period of time in order to detect complex events. BUTLER does also provide simple event processing features (that can be implemented by means of the same Complex Event Processing engine in charge of complex analysis).

Business Event is the term used in BUTLER to describe the simple event processing functionalities provided to Service Developers. An event is a set of conditions that, if satisfied or evaluated to true, determine an action must be carried out. In BUTLER, the conditions an event is made of take as inputs data items that come from SmartObjects. Therefore, when the conditions that take as input SmartObject data are evaluated to true, an action is triggered. Such an action is usually the delivery of another data item to the event consumer. Said data item can be the one that made the set of rules evaluate to true.

At the SmartObject Gateway level, is also possible to provide simple event processing functionalities. Here, the application wishing to consume simple events can subscribe to SmartObjects services and resources to get the appropriate events changes. The subscription is possible according to a set of condition and/or a specific period.
### 4.5.2.7.2. Interaction with other functional components

The Simple Event Processing functional component takes advantage of the same Complex Event Processing engine that is used to support Resource Exposition. Therefore, it takes as inputs data streams provided by the Device Data Collector / Actuator and, according to the configuration received from the Data Management and Marketplace Portal, delivers notifications to the Resource Exposition functional component.

### 4.5.2.7.3. Operations

The following operations are provided by the Context Exposition functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Business Event Definition</td>
<td>The Simple Event Processing functional component receives from the Data Management and Marketplace Portal the rules it needs to detect a Business Event.</td>
<td>• Business Event declaration (according to the CEP native interface). The set of conditions include a BusinessEventID and a Subscription Token (to embed in the notifications) and the Data Sources identifiers (to know which Data Sources must be evaluated in order to detect an event)</td>
</tr>
<tr>
<td>Update Business Event Definition</td>
<td>The Simple Event Processing functional component receives from the Data Management and Marketplace Portal a new set of rules that update the definition of an existing Business Event</td>
<td>• BusinessEventID • New set of conditions and/or action</td>
</tr>
<tr>
<td>Get Business Event Definition</td>
<td>The Simple Event Processing functional component receives from the Data Management and Marketplace Portal a retrieval message to read the definition of an existing Business Event</td>
<td>• BusinessEventID <strong>Return value</strong>: Business Event declaration</td>
</tr>
<tr>
<td>Delete Business Event</td>
<td>The Simple Event Processing functional component receives from the Data Management and Marketplace Portal an order to remove an existing Business Event given its identifier or the identifier of its owner (it deletes all the Business Events defined by the user)</td>
<td>One of the following: • BusinessEventID • UserID</td>
</tr>
<tr>
<td>Reception of Data Items</td>
<td>In order to detect an event, the Simple Event Processing functional component receives data items that will be used as input for the evaluation of rules.</td>
<td>• Data Items (including each a DataSourceID)</td>
</tr>
<tr>
<td>Delivery of Events</td>
<td>Data is delivered to Business Events consumers (applications) according to the available subscriptions.</td>
<td>• Data Items (including each a DataSourceID, a BusinessEventID and a Subscription Token).</td>
</tr>
</tbody>
</table>

Table 29: Simple Event Processing supported operations
4.5.2.8. Localization Manager

4.5.2.8.1. Description
As introduced in section 4.4.4 the Localization Manager interacts with the SmartObject Gateway, from which it receives raw ranging data, and with the Context Exposition (Pub/Sub Broker) to which it publishes estimated positions as context data. Going into more details, Figure 84 shows all the messages exchanged by the Localization Manager with the SmartObject Gateway and the Context Exposition Pub/Sub Broker. In the example showed in the figure, the unknown SmartObject A periodically (e.g. every 2 seconds) sends range measurements (e.g., ToA, AoA, RSSI) to the SmartObject Gateway which in turn sends these data to the Localization Manager using a uniform interface named sendRangingData(...). Moreover, the SmartObject Gateway can send to the Localization Manager anchor node positions by using the interface sendAnchorPosData(...). It is worth noting that this interface can be also used by the operator that deploys the anchor nodes in the environment and updates the anchors’ coordinates by using its own mobile device. More details about the sendRangingData(...) and sendAnchorPosData(...) messages are reported in [7].

The right side of the Localization Manager shows the interaction with the Context Exposition functional component. In particular, as soon as the location estimation of a SmartObject is updated, the Localization Manager sends an update to the Context Exposition by using localizationContextUpdate(...) message. Moreover, the Localization Manager replies to the localization requests made by either the Context Exposition or any other applications through the getLocalization(...) message. More details about the localizationContextUpdate(...) and getLocalization(...) messages are reported in [5].
4.5.2.8.2. Interactions with other functional components
This functional component interacts with the Device Data Collector / Actuator to obtain ranging data from the SmartObject Gateway and with the Context / Behavior Information Provider functional component to which it provides the estimated positions.

4.5.2.8.3. Operations
The following operations are provided by Localization Manager functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Localization</td>
<td>It returns the estimated position of a</td>
<td>Input arguments:</td>
</tr>
<tr>
<td></td>
<td>SmartObject at a given time.</td>
<td>• SmartObjectId</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• typeLoc: “relative” or “absolute”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• time_stamp_input: time at which is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>requested the position</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Return type:</strong> estimated position</td>
</tr>
</tbody>
</table>

Table 30: Localization Manager supported operations

4.5.2.9. User Profile Manager

4.5.2.9.1. Description
As centralized user management system, User Profile Manager allows users to:
• Register in the BUTLER system creating new digital identities or reusing their already existing digital identities (those coming from external identity providers).
• Interact with different BUTLER applications re-using the same identity.
• Connect applications to their social network accounts to enable social features (e.g. cross-posting on external social network).

And client applications to:
• Access and update users’ profiles data.
• Offer social features retrieving users’ social tokens from the connected social networks.

User Profile Manager exposes an interface to allow applications to retrieve and update user’s profile data. The interface is compliant with OpenSocial specification [27].

4.5.2.9.2. Interactions with other functional components
This functional component interacts with the Authorization Server functional component and with any other component that requires or modifies user profile data (such as the Data Management and Marketplace Portal).

4.5.2.9.3. Operations
The following operations are provided by the User Profile Manager functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Account</td>
<td>Register a new user profile in the BUTLER</td>
<td>• Username (login identifier)</td>
</tr>
<tr>
<td></td>
<td>system</td>
<td>• Password</td>
</tr>
<tr>
<td>Get Account</td>
<td>Returns user profile data for the specified</td>
<td>• UserID (identifier)</td>
</tr>
<tr>
<td></td>
<td>user</td>
<td></td>
</tr>
<tr>
<td>Update Account</td>
<td>Update user profile data</td>
<td>• UserID (identifier)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated user profile data</td>
</tr>
<tr>
<td>Delete Account</td>
<td>Delete a user profile from the system</td>
<td>• UserID (identifier)</td>
</tr>
</tbody>
</table>
4.5.2.10. Device Data Collector / Actuator

4.5.2.10.1. Description

BUTLER SmartObjects allows collecting device data collection and invoke actions on devices. The service and resource model described in section 4.4.2.3 defines common methods allowing accessing resources exposed by services: GET, SET, ACT, SUBSCRIBE, UNSUBSCRIBE. These methods are implemented by BUTLER SmartObjects which will invoke specific device methods via the underlying communication protocol. Some examples of device data collection and actuation are presented below.

- **Get sensor values:** a sensor node is able to send its values on demand. For example, to get sensor values from a Zolertia node, a COAP GET request might be send with the following URI: coap://[node_address]/sensors/phidget. The node will send the current value of the phidget sensors [42] connected to the node in JSON format. A Zolertia node with 2 connected phidgets (a 5V phidget and a 3V phidget) will return: {"5V":135, "3V":29.2}. To get sensors values from a Wasp mote node, a Transmit Request might be sent to the node with 0x52 as ApplicationID and sensor_name#value in the payload. For example, to the request tmp#value, a Wasp mote node will answer tmp#value#28.5.

- **Subscribe to events from a sensor:** On some devices nodes, the client can subscribe to sensor values that will be notified according to a condition. This condition can be modified remotely using SET_SENSOR_CONDITION property that allows the sensor to send the value:
  - Periodically at each period of time defined above,
  - Only if the value is superior or inferior to a certain threshold,
  - Only if the value is different of a certain percentage from the previous value.

  For example, in Zolertia nodes a subscription might be done using a COAP OBSERVE request with the following URI: coap://[node_address]/sensors/phidget. To modify the sending condition of a sensor on a Zolertia node, a COAP POST request must be sent with the following URI: coap://[node_address]/admin/deviceManagement? Phidget = phidget_ID with condition=new_condition in the payload, where phidget_ID can be 5V or 3V according to the socket to which the sensor is connected.

- **Modify parameter values on a sensor:** several parameters can be modified remotely on sensors. However, after a reboot, the node will come back to its initial configuration. For example, to modify the scanning period on a Wasp mote node, a Transmit Request is sent to the node with 0x50 as ApplicationID and period#new_period_value#unit in the payload, where new_period is an integer and unit can be s for second, m for minute and h for hour.

- **Act on sensors:** actuators can respond positively to action methods if they are supported by the device. For example, sleep operation asks the devices to enter its Sleep Mode to save some battery. It takes as input the sleep duration of the device. While sleeping, devices stop polling its sensors and can’t receive or send data. In a Wasp mote node, a Transmit Request must be sent with 0x58 as ApplicationID, and sleep#DD:HH:MM:SS in the payload, where DD:HH:MM:SS is the value of the sleep duration as follow: sleep#00:00:05:30. This request asks the device to sleep for 5 minutes and 30 seconds.

4.5.2.10.2. Interactions with other functional components

The figure below illustrates the data collection/actuation flow. Data collection and actuation are mainly triggered following client requests. At the device level, the communication between SmartObjectServices and physical devices is performed using specific device methods and communication protocols (CoAP, ZigBee, etc.). At the SmartObject Gateway level, devices are accessed in a homogeneous way thanks to the service and resource model presented in section 4.4.2.3.
4.5.2.10.3. Operations

The operations for device data collection and actuation correspond to the access methods associated to resources. Table 32 resumes these operations from the standpoint of a SmartObjectService (see Figure 85).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>get</strong></td>
<td>Gets the current data of the given resource. Data collection on the corresponding device could be performed.</td>
<td>• Resource name</td>
</tr>
<tr>
<td><strong>set</strong></td>
<td>Sets the given data value and set of metadata to the resource. Data modification on the corresponding device should be performed.</td>
<td>• Resource name • Object value (mandatory) • Set of metadata</td>
</tr>
<tr>
<td><strong>act</strong></td>
<td>Executes the resource with the given input parameters. Actuation on the corresponding device should be performed.</td>
<td>• Resource name • Parameters</td>
</tr>
<tr>
<td><strong>subscribe</strong></td>
<td>Subscribes the client to the resource in order to collect its data. Observation on the corresponding device should be performed.</td>
<td>• Resource name • Subscriber listener (mandatory) • Set of conditions • Period</td>
</tr>
<tr>
<td><strong>unsubscribe</strong></td>
<td>Cancels the given subscription. Observation on the corresponding device should be canceled.</td>
<td>• Resource name • SubscriptionID (mandatory)</td>
</tr>
</tbody>
</table>

Table 32: Device Data Collection and Actuation supported operations

4.5.2.11. Context / Behavior Information Provider

4.5.2.11.1. Description

BUTLER addresses the extraction of contextual information following different approaches. There are specialized mechanisms to extract very specific contextual information: location is generated through the Localization Manager (see section 4.5.2.8); user behavior is captured by means of the User Behavior Capture (see section 4.5.2.4). On the other hand, BUTLER provides a generic mechanism to extract contextual information associated to Virtual Entities by applying complex event processing techniques to the data generated by the SmartObjects that are associated to Virtual Entity. This approach lies in the possibility of defining each of the attributes the context associated to an Entity is made of as a composition of the data items being generated by a set of SmartObjects.

Thus, the Context / Behavior Information Provider is responsible for providing context information related to Virtual Entities. It relies on a Complex Event Processing Engine to provide its functionality. As a Complex
Event Processing engine it transforms incoming data streams (from SmartObjects) into new outgoing data streams each of them being one of the attributes the context is made of. The transformation process is ruled by the CEP applications created by from the Context Management Portal.

4.5.2.11.1. Interactions with other functional components
The Context / Behavior Information Provider can be used by other functional components within the BUTLER architecture:
- Device Data Collector / Actuator
- Context Management Portal
- Persistent Storage
- Context Exposition (Publication/Subscription Broker)

4.5.2.11.2. Operations
The Context / Behavior Information Provider functional component provides the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Input</td>
<td>Collection of sensor data provided by SmartObjects through the Device Data Collector / Actuator</td>
<td>Input data streams must contain the information needed to identify the streams and to apply the relevant business logic.</td>
</tr>
<tr>
<td>Event Output</td>
<td>New data streams created by the Context / Behavior Information Provider representing each of them an Entity attribute</td>
<td>Data Items (including each an EntityID, and Entity name and a ContextAttributeID)</td>
</tr>
</tbody>
</table>
| Receive Attribute declaration | Deployment of the event processing logic to declare a new Entity attribute. It is invoked by the Context Management Portal | · EntityID (mandatory)  
· Context Attribute Name (mandatory)  
· Context Attribute Type (optional)  
· Context Attribute declaration (set of expressions including some DataSourceID(s)) |

Table 33: Context / Behavior Information Provider supported operations

4.5.3. Services Layer

4.5.3.1. General
In Figure 86 below the services architectural layer is represented. This layer in the overall architecture is responsible for specifying necessary models, components and interfaces for description, discovery, binding, deployment and provisioning of context-aware services. It is also responsible for making said services available to the applications created by service developers so that the BUTLER experience can be extended beyond a limited set of built-in services and applications.

This layer specification includes several functional components to enable the creation of context-aware applications and the interaction of users with the BUTLER environment. Functional components in this layer are dedicated to Expose the different types of data BUTLER is built upon: Resources, Context, Behavior, Location and User Profiles (and usually rely in the corresponding functional components in the Data / Context Management Layer). Resource Exposition is responsible for enabling the access by applications to the data being generated by SmartObjects. A higher level of abstraction is provided through so-called data sources that represent the data flows being constantly generated by SmartObjects. The Services Layer provides also the means for enabling the Discovery of data sources.

BUTLER Services are also managed in this architectural layer. A number of functional components take the role of Service Registration (services are registered with a service description that define service
functionalities and properties; once registered they are available for being discovered and used); Service Discovery; Service Composition (in the form of hierarchical trees showing the dependencies between services); and Service Re-binding (when according to the dynamic nature of services, they can appear and disappear and the right dependencies need to be kept up-to-date).

Finally, a number of built-in services and applications is also provided and, although they do not have always an architectural role, they are naturally included as part of the Services Layer. On one hand, BUTLER provides a Multimedia Service following the BUTLER Service and Resource Model that controls media rendering operations on media SmartObjects via a general-purpose interface. It manages all the BUTLER media services, such as TV services. On the other hand, the Energy Service (in fact a context-aware application) makes use of the Data / Context Management functionalities provided by BUTLER to predict energy demand and consumption from the data sent by SmartObjects modeling energy-related devices (smart meters or home energy generators). An Application Repository stores the description of BUTLER applications and makes them available to end-users.

The following functional components have been identified within the Services Layer. For easing its description and understanding, they have been classified according to several functional groups (just three of them are described in this section; the remaining two are described in section 4.5.4):

- **Data Exposition Services**: This functional group takes care of the exposition of the different types of information the system provides to BUTLER applications.
  - **Resource Exposition**: Resources and data sources in a broad sense.
  - **Context Exposition**: Context information.
  - **Behavior Exposition**: Specific context information related to user behavior.
  - **Localization Exposition**: It exposes the current or past estimated positions of SmartObjects.
  - **User Profile Exposition**: The contents of the user profile.

- **Data and Service Discovery**: This functional group provides the means for the registration, composition and discovery of SmartObject Services and for the management and registration of virtual entities and data sources associated to SmartObjects.
  - **Data Discovery**: This functional component aims to record the descriptions of Data Sources (Resources hosted by SmartObjects) and Entities so that they become usable in the Services Layer.
  - **Service Registration**: It enables the registration of services hosted by SmartObjects.
  - **Service Composition**: It enables the creation of services hierarchies.
  - **Service Re-binding**: It enables the management of services that can appear and disappear dynamically.
  - **Service Discovery**: It enables the discovery of services hosted by SmartObjects.

- **Built-in Context Provider Applications**: These are some context-aware services applications that BUTLER provides to support the domains identified in BUTLER. They work as if they were other applications created by external Service Developers. A Repository is also included here.
  - **Multimedia Services**: They provide TV-based functionalities.
• **Energy Services**: They calculate both energy consumption and renewable energy generation predictions.

• **Applications Repository**: It stores the description of BUTLER applications and makes them available to end-users.

### 4.5.3.2. Multimedia Service

#### 4.5.3.2.1. Description

The Multimedia Service (MMS) is a Service according to the BUTLER Service and Resource Model used to control media rendering operations on media devices via a general-purpose interface. The multimedia service (MMS) is the unique component managing all the BUTLER media services, such as TV services.

The Multimedia Service aggregates media services, such as messaging, player and DVB (Digital Video Broadcasting) services. It is composed by the 3 following interfaces:

- **ButlerDVBAggregateService**: the ButlerDVBService aggregation. Each action on ButlerDVBService can be sent to one SmartObject, or to all SmartObjects, or to all SmartObjects located in a given room.

- **ButlerMessagingAggregateService**: the ButlerMessagingService aggregation. Each action on ButlerMessagingService can be sent to one SmartObject or to all SmartObjects, or to all SmartObjects located in a given room.

- **ButlerPlayerAggregateService**: the ButlerPlayerService aggregation. Each action on ButlerPlayerService can be sent to one SmartObject, or to all SmartObjects, or to all SmartObjects located in a given room.

The MMS service is used by the FollowMe application presented in section 5.2.

#### 4.5.3.2.2. Interaction with other functional components

The Multimedia Service could interact with any other functional component willing to use player, messaging or DVB services. For example, the Monitoring Services could use the Multimedia Service in order to send some monitoring messages to the devices providing displaying functionalities.
4.5.3.2.3. Operations
The main operations of the interfaces composing the multimedia service (i.e. Messaging, DVB and Player) are presented in the following Tables separated by interfaces.

4.5.3.2.3.1. Multimedia Messaging Service

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| displayMessageToRoom| Display the given message to all devices in a specific room.            | • roomId: String – the room identifier  
|                    |                                                                         | • text: String – the text of the message  
|                    |                                                                         | • timeout: Integer – the duration in seconds.                                                                                          |
| displayMessageToAll | Display the given message to all devices.                              | • text: String – the text of the message  
|                    |                                                                         | • timeout: Integer – the duration in seconds.                                                                                          |
| displayMessageToDevice | Display the given message to a device.  | • SmartObjectID: String – the SmartObject identifier  
|                    |                                                                         | • text: String – the text of the message  
|                    |                                                                         | • timeout: Integer – the duration in seconds.                                                                                          |
| disposeMessageToRoom | Dispose the previous message sent to all devices in a specific room. | • roomId: String – the room identifier                                                                                                    |
| disposeMessageToAll | Dispose the previous message sent to all devices.                      |                                                                                                                                               |
| disposeMessageToDevice | Dispose the previous message sent to a device.  | • SmartObjectID: String – the SmartObject identifier                                                                                     |
| displayWebcamToRoom  | Display a refreshed webcam picture in PIP to all devices in a specific room. | • roomId: String – the room identifier  
|                    |                                                                         | • url: String –URL of the webcam picture  
|                    |                                                                         | • timeout: Integer – the duration in seconds.                                                                                          |
| displayWebcamToAll   | Display a refreshed webcam picture in PIP to all devices.              | • url: String –URL of the webcam picture  
|                    |                                                                         | • timeout: Integer – the duration in seconds.                                                                                          |
| displayWebcamToDevice | Display a refreshed to a device. webcam picture in PIP  | • SmartobjectID: String – the SmartObject identifier  
|                    |                                                                         | • url: String –URL of the webcam picture  
|                    |                                                                         | • timeout: Integer – the duration in seconds.                                                                                          |
| disposeWebcamToRoom   | Dispose the webcam PIP to all devices in a specific room.             | • roomId: String – the room identifier                                                                                                    |
| disposeWebcamToAll    | Dispose the webcam PIP to all devices.                                 |                                                                                                                                               |
| disposeWebcamToDevice | Dispose the webcam PIP to a device.  | • SmartObjectID: String – the SmartObject identifier                                                                                     |

Table 34: Multimedia Messaging Service supported operations
The Multimedia Messaging service provides also operations to display SMS and HTML content messages in devices.

4.5.3.2.3.2. Multimedia DVB Service

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| zapForRoom      | Open the TV tuner to a channel on all devices in a room | • roomld: String – the room identifier  
|                 |                                                   | • channelNumber: Integer – the channel number.                             |
| zapForAll       | Open the TV tuner to a channel on all devices      | • channelNumber: Integer – the channel number.                             |
| zapForDevice    | Open the TV tuner to a channel on a device         | • SmartObjectID: String – the SmartObject identifier  
|                 |                                                   | • channelNumber: Integer – the channel number.                             |

Table 35: Multimedia DVB Services supported operations

4.5.3.2.3.3. Multimedia Player Service

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| playToRoom      | Open and play the given media for all devices in a room. | • roomld: String – the room identifier  
|                 |                                                   | • media: MediaModel Json String – the media to play.                      |
|                 |                                                   | • startPositionMs: Long – the position in the stream where to start to play, in milliseconds. |
| playToAll       | Open and play the given media for all devices.    | • media: MediaModel Json String – the media to play.                      |
|                 |                                                   | • startPositionMs: Long – the position in the stream where to start to play, in milliseconds. |
| playToDevice    | Open and play the given media for a given device. | • SmartObjectID: String – the SmartObject identifier  
|                 |                                                   | • media: MediaModel Json String – the media to play.                      |
|                 |                                                   | • startPositionMs: Long – the position in the stream where to start to play, in milliseconds. |
| stopToRoom      | Stop and close the current media for all devices in a room. | • roomld: String – the room identifier                                     |
| stopToAll       | Stop and close the current media for all devices. |                                                                             |
| stopToDevice    | Stop and close the current media for a given device. | • SmartObjectID: String – the SmartObject identifier                     |
| pauseToRoom     | Pause the stream for all devices in a room.       | • roomld: String – the room identifier                                     |
| pauseToAll      | Pause the stream for all devices.                 |                                                                             |
| pauseToDevice   | Pause the stream for a given device.              | • SmartObjectID: String – the SmartObject identifier                     |
forwardMedia | Forward the media from a device to another device.  
| srcSmartObjectID: String – the SmartObject identifier for the source device.  
| destSmartObjectID: String – the SmartObject identifier for the destination device.

Table 36: Multimedia Player Service supported operations

The Multimedia Player service provides also operations to resume a paused media, and for jump forward and backward the media reproduction.

4.5.3.3. Energy Service

4.5.3.3.1. Energy Forecasting Service

The core of this BUTLER application calculates both energy consumption and renewable energy generation predictions based on three different inputs: current consumption rates, current weather conditions and weather forecast. Gathering that information it is able to provide the best and worst moment, regarding to the estimated price, to consume power. The technologies involved are Complex Event Processing, relational database storage and Java SE/EE. It is also under discussion to include contextual information from consumer side. This information can include a previously known calendar and schedule data as well as learned behavioral patterns of end-users.

4.5.3.3.2. Description

Currently his application consists of five different parts:

- Weather forecast data collecting
- Smart meters data collecting
- Data aggregating
- Energy consumption and generation forecasting
- Results exposure

Let’s explain how each part works step by step:

4.5.3.3.3. Weather forecast data collecting

JSON files with 10 days hourly weather forecast for the desired locations are downloaded several times per day (currently from Wunderground.com, through their so-called Anvil plan for developers, with no cost while the number of downloads per day don’t get over 500). Those predicted weather conditions are dumped into the database. Those files also contain current weather conditions, which are sent to the Complex Event Processing engine as events.

4.5.3.3.4. Smart meters data collecting

Several groups of virtual smart meters are sending simulated energy consumption measures regularly from each location to the Complex Event Processing engine as events. It will be also possible to send detailed energy consumption information classified into several SmartObject types in the future.

4.5.3.3.5. Data aggregating

The Complex Event Processing engine recognizes two kinds of events: energy consumption and weather conditions. Its recurrent behavior works as follows:

- It opens a 30 minutes time window and analyzes every single event that is received during its length.
- If a weather condition event is received, the number of wind mills / solar panels installed in the location the weather conditions belong to is queried, and the power generation for that single moment is calculated and stored into the database.
- If an energy consumption event is received, it is buffered until the time window is closed.
Once the time window is closed, all the energy consumption events are aggregated considering their origin location, added and the result is stored into the database. Current weather conditions are also included.

Then a new time window is opened.

4.5.3.3.6. Energy consumption and generation forecasting
There is a process that, using as input the stored historical energy consumptions, generates forecasted consumptions based on linear and multilinear regressions prediction models. The independent values are the weather conditions, both past and forecasted.

In the other hand, it calculates forecasted energy generation too. It takes as input the weather forecast stored into the database and the number of wind mills / solar panels installed in each location and calculates the power generation.

Both results, consumption and generation, are stored into the database as well.

4.5.3.3.6.1. Results exposure
The application can be accessed through a web service that offers three functions per kind of renewable energy (wind and solar). Its logic is based in the assumption that a high renewable power production decreases the power price as well as a high power demand increases it.

- The first one calculates the estimated time for maximum energy price within a period of time (expressed as an amount of hours from the present moment) for a specific location. It queries consumption and generation forecast for each hour within that period and, after comparing them, returns the moment when the difference between both values is the biggest one \((\text{consumption}(t) - \text{generation}(t))\).
- The second one calculates the estimated time for minimum energy price within a period of time (expressed as an amount of hours from the present moment) for a specific location. It queries consumption and generation forecast for each hour within that period and, after comparing them, returns the moment when the difference between both values is the smallest one \((\text{consumption}(t) - \text{generation}(t))\).
- The third function gauges the accuracy of the energy generation forecast. It compares past real-time generation data with past forecasted generation data for a past period in a specific location and calculates the mean absolute percentage error.

4.5.3.3.6.2. Interaction with other functional components
Within the BUTLER architecture, this Energy Forecasting Service has dependences on two elements from two different layers:
- In the Data/Context Management Layer, the Device Data Collector / Actuator in order to receive the data sent by the smart meters.
- In the Services Layer, the Complex Event Processing Engine, in order to aggregate and analyze the data sent by the smart meters, managing those messages as events.

Outside BUTLER, there is a dependency on the weather forecast provider.

4.5.3.3.6.3. Operations
The API is implemented through an Axis Web Service. There are three available operations for each kind of renewable energy:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Time</td>
<td>of maximal price</td>
<td>Input arguments:</td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
<td>Input arguments</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Maximum Wind Energy Price</strong></td>
<td>in the interval of X hours from the current time for the energy produced by the wind farms in a specific location</td>
<td>• Length of the forecasting time window in hours (max 48h): integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location: string</td>
</tr>
<tr>
<td><strong>Get Time of Minimum Wind Energy Price</strong></td>
<td>It returns the time for minimal €-price in the interval of X hours from the current time for the energy produced by the wind farms in a specific location.</td>
<td>• Length of the forecasting time window in hours (max 48h): integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location: string</td>
</tr>
<tr>
<td><strong>Get Wind Forecast Confidence</strong></td>
<td>It returns the forecasting %-confidence of energy price produced by the wind farms in a specific location based in the data stored into the database during the last X hours. If the result is 0, try again with a smaller amount of hours.</td>
<td>• Length of the forecasting time window in hours: integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location: string</td>
</tr>
<tr>
<td><strong>Get Time Of Maximum Solar Energy Price</strong></td>
<td>It returns the time for maximal €-price in the interval of X hours from the current time for the energy produced by the solar farms in a specific location.</td>
<td>• Length of the forecasting time window in hours (max 48h): integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location: string</td>
</tr>
<tr>
<td><strong>Get Time Of Minimum Solar Energy Price</strong></td>
<td>It returns the time for minimal €-price in the interval of X hours from the current time for the energy produced by the solar farms in a specific location.</td>
<td>• Length of the forecasting time window in hours (max 48h): integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location: string</td>
</tr>
<tr>
<td><strong>Get Solar Forecast Confidence</strong></td>
<td>It returns the forecasting %-confidence of energy price produced by the solar farms in a specific location based in the data stored into the database during the last X hours. If the result is 0, try again with a smaller amount of hours.</td>
<td>• Length of the forecasting time window in hours: integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location: string</td>
</tr>
</tbody>
</table>

*Table 37: Energy Service supported operations*

4.5.3.4. Application Repository

4.5.3.4.1. Description

The Application Repository is a web application that stores and makes BUTLER Applications available to end-users. Basically, it allows the deployment of SmartMobile applications (as defined in D4.1 [5] and D4.2 [6]). Hybrid applications can be fully downloaded by SmartMobile clients and mobile web browsers supporting HTML5 can access their web mobile versions.

Moreover, the Application Repository also provides a set of tools for service developers in order to simply deploy their applications: developers get tools to automate the deployment of BUTLER applications to the repository. From his/her development machine, a developer can request a deployment by means of a command-line tool: a script will sign the version with his/her GPG key, the app will then be packaged and sent on the repository. Once uploaded, the server will check the signature, unpack the version and deploy it. Then the application will be available to end-users and listed in the catalogue exposed by the Applications Repository.

4.5.3.4.2. Operations

The Application Repository provides the following operations:
<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get list of available applications</td>
<td>Provides the list of fully deployed applications. This list is mainly used by SmartMobile applications that list available applications to end-users.</td>
<td>None.</td>
</tr>
<tr>
<td>Get list of files for a specific applications</td>
<td>For a specific application, returns the list of files. This is used by SmartMobile to download applications.</td>
<td>Application id (String)</td>
</tr>
<tr>
<td>Upload Application</td>
<td>This operation allows developers tools to send applications to be deployed on the repository.</td>
<td>Application id, Developer id, Signed and packaged version of the application to be deployed</td>
</tr>
</tbody>
</table>

Table 38: Application Repository supported operations

4.5.3.5. Data and Service Directory
This functional group is responsible for maintaining information about BUTLER SmartObjects, Services and Resources in repositories. The BUTLER SmartObject Gateway uses the OSGi registry in order to maintain and lookup-for device related information.

4.5.3.5.1. Service Registration

4.5.3.5.1.1. Description
At the BUTLER SmartObject Gateway service registration is performed using the OSGi service registry. Such a registry is a crucial feature of OSGi that promotes a dynamic model based on the service-oriented paradigm: publish, discover, bind. Services are registered into the registry with a service description (defining service functionalities and properties). Registered services are then available for being discovered and used by different clients. Figure 88 shows the service registration process: A and B are registered into the service registry as providers of the service S.

![Figure 88: Service Registration](image)

4.5.3.5.1.2. Interaction with other functional components
The BUTLER Service Registration interacts with the Service Discovery component: once a SmartObject discovered, the services it provides are published into the registry via the Service Registration component.
Monitoring services could interact with the Service Registration component, for example, to update or remove existing service registrations.

4.5.3.5.1.3. Operations

The BUTLER Service Registration related operations are the following:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| register  | Register the given service into the service directory | • ServiceID (mandatory)  
• Service description (mandatory) |
| unregister| Unregister the given service into the service directory | • ServiceID (mandatory) |
| update    | Updates the given service into the service directory | • ServiceID (mandatory)  
• Service description (mandatory) |

Table 39: Service Registration supported operations

4.5.3.5.2. Service Discovery

4.5.3.5.2.1. Description

Services registered into the Service Directory can be discovered using the service description. The service Discovery component allows thus to look for services based on their provided properties and functionalities. Figure 89 shows the service discovery process: a client asks the registry for services providing the service (interface) S; the registry answers with the references of the A and B providers. The client binds to selected services in order to use their provided functionalities.

![Service Discovery Diagram](image)

Figure 89: Service Discovery

4.5.3.5.2.2. Interaction with other functional components

The BUTLER Service Discovery interacts with components willing to access the Service Directory in order to look for services.

4.5.3.5.2.3. Operations

The following operations are provided by the Service Discovery functional component.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| lookup    | Look for a registered service | • Service description (mandatory)  
• Service properties (optional) |

Table 40: Service Discovery supported operations
4.5.3.5.3. Service Composition

4.5.3.5.3.1. Description
The advantage of BUTLER SmartObject Gateway design is the capability to compose services in the form of a hierarchical dependency tree. Applications can be constructed simply by declaring dependencies to low level services or high level services depending on its needs. However, the goal of service composition at this level is not to provide an automatic composition component or tools. It is up to the service developer to declare dependencies to different SmartObject services and resources and to provide its new services and resources. The following picture shows a service composition example.

In the figure above, high level BUTLER services reuse BUTLER SmartObject services as explained in the following table. The example is taken from the Follow-Me application detailed in section 5.2.

4.5.3.5.3.1. Operations
The following operations are provided by the Service Composition functional component.

<table>
<thead>
<tr>
<th>Composite service</th>
<th>Used services</th>
<th>Used resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multimedia follow me</td>
<td>LocationService</td>
<td>occupiedRoom.get()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unoccupied.subscribe(occupiedRoomId)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>occupiedRoom.subscribe()</td>
</tr>
<tr>
<td>MultimediaService</td>
<td>deviceId.get(occupiedRoomId)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>volume.set(volumeposition)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mediaInfo.get(deviceId)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stop.act()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>play.act(mediaInfo, newRoomId)</td>
<td></td>
</tr>
</tbody>
</table>
BUTLER Service Composition functional component does not provide at this stage specific operation for autonomic composition purpose. Instead, it claims the respect of the resources access API described in section 4.4.2 which already provide a complete interface (GET, SET, ACT, SUBSCRIBE, UNSUBSCRIBE) necessary to any BUTLER composition process. The previous table gives an example.

### 4.5.3.5.4. Service Re-Binding

#### 4.5.3.5.4.1. Description

The execution lifecycle of SmartObject services relay on OSGi platform and iPOJO design lifecycle. Because, OSGi (Service Dependency Management) services can appear and disappear dynamically, this implies dependencies can target a provider which can appear or disappear dynamically. So, dependencies need to manage this dynamism by tracking every time available services. At any moment, a dependency can be unresolved (i.e. no more provider can fulfill the requirement). In the case of a mandatory requirement, the instance becomes invalid (an invalid instance is no more accessible externally, for example provided services are unpublished). If a service, resolving the unfilled dependency appears, the instance becomes valid. In consequence, dependencies affect directly the instance state, and must manage correctly OSGi dynamism to allow a complete unloading when a service goes away. As soon a mandatory dependency cannot be fulfilled, the instance is invalidated.

By default, dependencies are managed dynamically (as previously explained). However, iPOJO supports two other types of binding policies:

1. **Static**: if a bound service disappears, the instance is invalidated and cannot be revalidated (binding broken forever)
2. **Dynamic-Priority**: at each injection, the best provider is injected, or the providers array is sorted according to the OSGi Ranking policy or to a specified sorting algorithm.

Pragmatically, this can be done as follows:

**Field injection**: a field contains the service object. As soon as the field is used, a consistent service object is injected. This injection type fully hides the dynamism

```java
@Requires
Private MotionService motionService;
```

**Method invocation**: when a service appears, or disappears a method in the component is invoked. For each dependency, bind / unbind / modified methods are invoked to notify the component of the event.

```java
@Bind
public void bindMotionService(MotionService motionService) { /*...*/ }
@Unbind
public void unbindMotionService(MotionService motionService) { /*...*/ }
@Modified
public void modifiedMotionService(MotionService motionService) { /*...*/ }
```

### 4.5.3.5.4.2. Operations

The following operations are provided by the Service Re-binding functional component.

<table>
<thead>
<tr>
<th>LocationService</th>
<th>PIRService</th>
<th>motionDetected.subscribe()</th>
</tr>
</thead>
</table>

Table 41: Service Composition supported operations
## 4.5.3.5.5. Data Discovery

### 4.5.3.5.5.1. Description

The Data Discovery functional component aims to record the descriptions of Data Sources (Resources hosted by SmartObjects) and Entities so that they become usable in the Services Layer. This functional component supports the key functionalities related to context management (see section 4.4.7) and data marketplace (see section 4.4.6). Unlike the resolution and discovery infrastructure provided by IoT-A the Data Discovery functional component in BUTLER provides explicit operations for the registration and discovery of a Virtual Entity in itself.

Functionalities related to Entity management are usually invoked from the management functional components that provide a front-end towards end users (such as the Data Management and Marketplace Portal and Context Management Portal). Functionalities related to Data Source management are invoked from a wider variety of functional components. For instance, the registration of Data Sources is usually achieved by integrating, making it accessible or synchronizing the available Resource Registries with the Data Discovery functional component.

It works as a CRUD repository with built-in geodetic functionalities in order to ease management of elements with a shape and a location. Each of the entries recorded by the Data Discovery functional component is known as descriptor. Thus, there are several types of involved descriptors: Entity Descriptors, Data Source Descriptors, and Data Offering Descriptors (see section 4.4.6.2).

### 4.5.3.5.5.2. Interaction with other functional components

The Data Discovery functional component is usually used in functionalities where the user needs to handle their Entities or Data Sources. Thus, its usual clients are the Data Management and Marketplace Portal and the Context Management Portal.

### 4.5.3.5.5.3. Operations

The following operations are provided by the Data Discovery functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lookup Entity Descriptor</td>
<td>Retrieval of the Entity Descriptor given its identifier</td>
<td>EntityID</td>
</tr>
</tbody>
</table>
| Get Entity Descriptors    | Retrieval of the descriptors of a group of Entities given the identifier of its owner or a base location | At least one of the following:  
  - UserID
  - Location (Radius is optional) |
| Create New Entity Descriptor | Registration of a new Entity. It provides an EntityID as output | • Entity Name (optional)  
  • Entity Type (optional)  
  • Entity Description (optional)  
  • Entity Owner (mandatory)  
  • Entity Coordinates (optional) |

### Table 42: Service Re-binding supported operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind</td>
<td>Service binding</td>
<td>ServiceID (mandatory)</td>
</tr>
<tr>
<td>unbind</td>
<td>Service unbinding</td>
<td>ServiceID (mandatory)</td>
</tr>
<tr>
<td>modify</td>
<td>Service modified</td>
<td>ServiceID (mandatory)</td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
<td>Return value</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Update Entity Descriptor</td>
<td>Update of an existing Entity Descriptor</td>
<td>Entity ID, Any of the attributes described in the Create New Entity Descriptor operation</td>
</tr>
<tr>
<td>Remove Entity Descriptor</td>
<td>Removal of an existing Entity given its identifier or the identifier of its owner</td>
<td>Entity ID</td>
</tr>
<tr>
<td>Lookup Data Source Descriptor</td>
<td>Retrieval of a Data Source Descriptor given the Data Source identifier</td>
<td>DataSource ID</td>
</tr>
<tr>
<td>Get Data Source Descriptor</td>
<td>Retrieval of the descriptors of a group of Data Sources given the identifier of its owner or a base location</td>
<td>At least one of the following: User ID, Location (Radius is optional)</td>
</tr>
<tr>
<td>Create Data Source Descriptor</td>
<td>Registration of a new Data Source</td>
<td>DataSource ID, Data Source Name (optional), Data Source Type (optional, type of the data being provided by the Data Source), Data Source End-Point (mandatory, a description of the protocols and semantics of the Data Source), Entity ID(s) (optional, it describes the Entities Data Sources are associated to), Data Source Conditions (optional, such as the frequency of generation of data)</td>
</tr>
<tr>
<td>Update Data Source Descriptor</td>
<td>Update of an existing Data Source Descriptor</td>
<td>DataSource ID, Any of the attributes described in the Create New Data Source Descriptor operation</td>
</tr>
<tr>
<td>Remove Data Source Descriptor</td>
<td>Removal of an existing Data Source given its identifier or the identifier of its owner</td>
<td>DataSource ID</td>
</tr>
<tr>
<td>Get Entity and Data Source Descriptors</td>
<td>Retrieval of the descriptors of all the Entities and Data Sources owned by a given user</td>
<td>UserID</td>
</tr>
</tbody>
</table>

**Table 43: Data Discovery supported operations**

4.5.3.6. Resource Exposition

4.5.3.6.1. Description

The Resource Exposition functional component implements the exposition of the Resources exposed by BUTLER. Resources hosted by SmartObjects can be exposed directly to applications (Resource Consumers) or mediated through a SmartServer. Therefore, different approaches are provided by BUTLER so that different scenarios are supported.

Direct exposition to Resources follows the guidelines provided by BUTLER Security Services (see section 4.4.3). Direct Resource Exposition is done through the set of available actions a Resource Provider provides.
(GET/SET/ACT/SUBSCRIBE/UNSUBSCRIBE). Besides, an access-token previously retrieved from the Authorization Server is required.

Mediated exposition of Resources is done by means of a SmartServer that accesses (or receives notifications from Resources), processes the resource information, cache it if needed and delivers to applications. The BUTLER approach relies on the purchase of Data Offerings, each of them comprising a predefined set of data streams each item being a read of the value of a given Resource. From the Resource Consumer point of view is not relevant how the SmartServer is able to gather and deliver the Resource information. It is the owner of the resources the one that creates Data Offerings and assign proper permissions. BUTLER provides two main paradigms for the mediated exposition of resources:

- Publication/subscription: The Resource Consumer subscribes to resource information (handled as data streams that are delivered to the Resource Consumer).
- Query/response: The Resource Consumer is able to read cached resource information. To do so it must retrieve first an access-token from the Authorization Server.

### 4.5.3.6.2. Interaction with other functional components

The Resource Exposition functional component is usually used by Resource Consumers that wishes to access Resources. Resource Consumers can be applications (in the direct exposition scenario) or the Device Data Collector / Actuator functional component (in the mediated scenario).

On the other hand, the Resource Exposition functional component is expected to receive data streams from Resources through the Complex Event Processing functional component and retrieve cached Resource data from the Persistent Storage functional component. It also receives configuration instructions from the Data Management and Marketplace Portal.

### 4.5.3.6.3. Operations

The following operations are provided by the Resource Exposition functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET/SET/ACT/SUBSCRIBE/UNSUBSCRIBE</td>
<td>Direct exposition scenario&lt;br&gt;Resources provide the following set of available actions:&lt;br&gt;• GET: Get the value attribute of the resource&lt;br&gt;• SET: Set the value of the resource with a given new value&lt;br&gt;• ACT: Execute action identified by the resource&lt;br&gt;• SUBSCRIBE: Subscribe to resources that are sent in terms of events&lt;br&gt;• UNSUBSCRIBE: Remove a previous subscription</td>
<td>• Access-Token (embeds the ResourceID)&lt;br&gt;• Application payload</td>
</tr>
<tr>
<td>Set Subscription Configuration</td>
<td>Mediated exposition scenario following the publication/subscription paradigm&lt;br&gt;The Resource Exposition functional component receives the configuration information it needs to handle subsequent user subscriptions requests. Such configuration information contains the credentials associated to the user for accessing a specific Data Offering</td>
<td>• Offering identifier&lt;br&gt;• DataSourceID(s)&lt;br&gt;• User Credentials&lt;br&gt;• Subscription Token (per offering)</td>
</tr>
</tbody>
</table>
### Subscription Request

**Mediated exposition scenario following the publication/subscription paradigm**

In order to receive data streams, the Data Consumer (application) must issue a subscription request for every Data Offering being previously purchased.

- User credentials
- Subscription Token

### Reception of Data Items

**Mediated exposition scenario following the publication/subscription paradigm**

In order to expose data streams from Resources, the Resource Exposition functional component receive data items that will be delivered to data consumers according to available subscriptions.

- Data Items (including each an offering identifier and a DataSourceID)

### Delivery of Data Items

**Mediated exposition scenario following the publication/subscription paradigm**

Data is delivered to Data Consumers according to the available subscriptions.

- Data Items (including each an offering identifier and a DataSourceID)

### Read of Data Items

**Mediated exposition scenario following the query/response paradigm**

The Data Consumer retrieves cached data from Resources.

- Access-token
- DataOfferingID
- Filtering conditions (data ranges, thresholds...)
- User identifier (UserID) on behalf of which the request is done

### Table 44: Resource Exposition supported operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| Set Subscription Configuration | The Context Exposition functional component receives the configuration information it needs to handle subsequent user subscriptions requests. Such configuration information contains the credentials associated to the user for | EntityID
- Entity name (optional)
- ContextAttributeID
- User credentials
- Subscription Token (per context attribute) |
In order to receive data streams, the Context Consumer (application) must issue a subscription request for every Data Offering being previously purchased.

In order to expose data streams representing the context attributes, the Context Exposition functional component receive data items that will be delivered to data consumers according to available subscriptions.

Data is delivered to Context Consumers according to the available subscriptions.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
</table>
| Get Activity               | This allows the user or application to specify a known user id, his or her location, a timestamp and a threshold for the minimum confidence in the activity and get a list of activities | • User id  
• User location (optional)  
• Time of activity (optional)  
• Threshold for confidence value (optional) |
| Get Activity Confidence    | This allows the user or application to specify a known user id, his or her location, a timestamp and the anticipated user activity and get the confidence in the prediction of | • User id  
• User location (optional)  
• Time of activity (optional)  
• User activity |
the activity being performed by the user

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Required Parameters</th>
</tr>
</thead>
</table>
| Get Semantic Location             | This offers a way to translate geo-location information of a user into semantically more meaningful information. | • Label of the map
• Co-ordinates (x,y) within the map |
| Get Steps                          | This allows the user or application to specify a user id, the date and time period when the number of steps is required by the application/user | • User id
• Date of activity (optional)
• Period of time of activity (optional) |
| Get Calorie Expenditure           | This allows the user or application to specify a user id, the date and time period when the calories spent information is required by the application/user | • User id
• Date of activity (optional)
• Period of time of activity (optional) |

Table 46: Behavior Exposition supported operations

4.5.3.9. User Profile Exposition

4.5.3.9.1. Description
User profile data in BUTLER is exposed by a set of APIs that allow any client application or server with proper authorization to perform the operations described in the following.

- Register User Profile
- Get User Profile
- Update User Profile
- (optionally) Delete User Profile

For each operation authorization is need according to the BUTLER general authorization and authentication mechanisms. All these operations are defined in order to create, read, update and eventually delete a user profile, which has been modeled in terms of data structure by the User Data Model.

4.5.3.9.2. User Data Model
The data user model for a user profile consists of a set of attributes, both plain and structured, generic enough to describe any relevant user profile data, including: name, several email addresses, photo, birthdate, etc. The designed data model allows also the addition of application specific attributes that any BUTLER service may need to store in a user profile. The intention of this attributes is to be profile-specific and not context-specific, meaning that only attributes that do not change so often are to be sorted in the BUTLER user profile; more frequently changing application specific attributes should instead be handled as if it were contextual information and therefore be updated and obtained through the context data layer.

The BUTLER data model has been designed following the de-facto OpenSocial standard [27], which is already used by several applications and is highly flexible in terms of interoperability and extensibility.

The following is a list of the main elements of the BUTLER User Profile data model, as defined in the Person OpenSocial object. Beside the Person object, other sub-objects are defined to store Addresses, Name components, Account, AppData.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addresses</td>
<td>Plural-Field &lt;Address&gt;</td>
<td>A physical mailing address for this Person.</td>
</tr>
<tr>
<td>appData</td>
<td>Plural-Field &lt;AppData&gt;</td>
<td>A collection of AppData keys and values.</td>
</tr>
<tr>
<td>displayName</td>
<td>string</td>
<td><strong>Required.</strong> The name of this Person, suitable for display to end-users. Each Person returned MUST include a non-empty displayName value.</td>
</tr>
<tr>
<td>emails</td>
<td>Plural-Field &lt;string&gt;</td>
<td>E-mail address for this Person.</td>
</tr>
</tbody>
</table>
In addition to the table above, other fields more related to social aspects include the following:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>number</td>
<td>The age of this person.</td>
</tr>
<tr>
<td>birthday</td>
<td>Date</td>
<td>The birthday of this person. The value MUST be a valid Date.</td>
</tr>
<tr>
<td>books</td>
<td>Plural-Field &lt;string&gt;</td>
<td>Person's favorite books.</td>
</tr>
<tr>
<td>ethnicity</td>
<td>string</td>
<td>Person’s ethnicity.</td>
</tr>
<tr>
<td>fashion</td>
<td>string</td>
<td>Person’s thoughts on fashion.</td>
</tr>
<tr>
<td>food</td>
<td>Plural-Field &lt;string&gt;</td>
<td>Person’s favorite food.</td>
</tr>
<tr>
<td>gender</td>
<td>string</td>
<td>The gender of this person. Values, if appropriate: male, female, or undisclosed</td>
</tr>
<tr>
<td>nickname</td>
<td>string</td>
<td>The casual way to address this Person in real life, e.g. &quot;Bob&quot; or &quot;Bobby&quot; instead of &quot;Robert&quot;</td>
</tr>
</tbody>
</table>

Table 48: Person data object attributes

4.5.3.9.3. Interaction with other functional components

This functional component interacts with the User Profile Manager functional component.

4.5.3.9.4. Operations

The following operations are provided by the User Profile Exposition functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register User Profile</td>
<td>Register a new User Profile in the system</td>
<td>• Person: a Person data object with all user profile information that the client application want to register in the user profile. In particular the <strong>preferredUsername</strong> parameter can be specified to select a username to use in the system</td>
</tr>
<tr>
<td>Get User Profile</td>
<td>Get a stored user profile from the system</td>
<td>• UserID: the identifier of the user. In case the client application want to obtain user profile data about the currently authorized user itself (as obtained by authorization grant),</td>
</tr>
</tbody>
</table>
the id parameter can be valued to a constant like: @self

Return value:
- **Person**: an `Person` data object representation of the found user profile

Update User Profile

<table>
<thead>
<tr>
<th>Update user profile information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Person</strong>: Person data object with user profile to be updated. All attributes MUST be included in the Person object: both those that are going to be changed and those that must remain the same.</td>
</tr>
</tbody>
</table>

Return value:
- **Person**: an OpenSocial `Person` data object representation of the updated user profile

Delete User Profile

<table>
<thead>
<tr>
<th>Delete a user profile from the system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UserID</strong>: the identifier of the user to be removed from the system. In case the authenticated and authorized user itself must be removed, this parameter can be valued to a constant like: @self</td>
</tr>
</tbody>
</table>

Table 49: User Profile Exposition supported operations

4.5.3.10. Localization Exposition

4.5.3.10.1. Description

The Localization Exposition exposes the current or past estimated positions of SmartObjects. In particular, this functional component uses the positions that are estimated by the Localization Manager. Positions can be expressed according to both a relative reference system (x, y, z) and the absolute one, i.e. in terms of latitude, longitude and height. In addition, this service provides the estimated position accuracy and the time at which the position has been calculated.

4.5.3.10.2. Interaction with other functional components

Within the BUTLER architecture, the Localization Exposition interacts with the Localization Manager, which belongs to the Data/Context Management Layer, to take the estimated positions at a time specified as input parameter.

4.5.3.10.3. Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
<th>Return value</th>
</tr>
</thead>
</table>
| Get Localization| It returns the estimated position of a SmartObject at a given time. | - SmartObjectID  
- typeLoc: “relative“ or “absolute“  
- time_stamp_input: time at which is requested the position | estimated position |

Table 50: Localization Exposition supported operations
4.5.4. System/Device Management Layer

4.5.4.1. General

In Figure 91 below the system and device management architectural layer is represented. It has to deal with a great number of heterogeneous networked sensor/actuator devices. This is a very challenging task because of scalability and communication protocols heterogeneity. To overcome this challenge, a management solution, which occurs at each layer of the BUTLER functional architecture, is proposed.

This helps to split the scalability and heterogeneity problems by layer, towards management actions standardization and to cover all management requirements for each layer. A detailed survey of work done is presented in the D4.3 [7] and the following figure resume significant management components.

---

**Figure 91: BUTLER Overall Architecture – System/Device Management Layer**

The following functional components have been identified within the System / Device Management Layer. All of them logically belong also to other layer. However, only the functional components belonging also to the Communications Layer have not been described in detail in this section:

- **Network Monitoring**: This functional component monitors the performance and the network parameters of BUTLER SmartObjects. It logically belongs also to the Communications Layer (and has been described there).

- **Device Monitoring**: It provides a basic set of device management operations to set several administrations and monitoring parameters used by device platforms. It logically belongs also to the Communications Layer (and has been described there).

- **User Portal**: It provides functionalities for the users of the BUTLER system: the Device Owners and the Service Developers. Through the User Portal, the former can manage data sources, data offerings and virtual entities. The latter can purchase data offerings and define contexts associated to entities. It is also a part of the Data / Context Management Layer.
  - **Data Management and Marketplace Portal**: Management and purchase of data offerings.
  - **Context Management Portal**: Definition of context associated to virtual entities.
• **User Profile Manager**: To manage user profile data and user specific application data (it can connect also the user profile to external social networks). It is also a part of the Data / Context Management Layer (and has been described there).

• **System Management**: It provides miscellaneous functionalities for Service deployment and monitoring and for the monitoring and software update of SmartObjects. It is also a part of the Services Layer.
  - **SmartObject Management Portal**: It provides an easy-to-use web GUI for information collection and update from to the BUTLER SmartObject Gateway
  - **Deployment**: Following OSGi principles, it is used to manage SmartObject services hosted by the BUTLER SmartObject Gateway.
  - **Service Monitoring**: It is used to monitor SmartObjects and the Resources and Services they host.
  - **Software Update**: It provides Over-The-Air software update capabilities for SmartObjects.

4.5.4.2. SmartObject Management Portal

4.5.4.2.1. Description

SmartObject Management Portal provides an easy-to-use web GUI for information collection and update from to the BUTLER SmartObject Gateway. The SmartObject Management Portal defines access policy to different related SmartObjects services and resources such as, permission granting and group management. This portal should also provide diagnostics and monitoring statistics. In addition, the web portal is used to register new clients with the SmartObjects services and provide additional profile information, for example preferred SmartObjects and consumer access protocol. The SmartObject Management Portal uses OSGi Web console which has a minimal management facilities and basic security functions. This has been extended to fit all the requirements presented partially by the following Figure 92.

![Figure 92: SmartObject Management Portal](image)

The SmartObject Management Portal allows performing administration tasks and secured access to the BUTLER SmartObject Gateway environment. It allows also overseeing services and events including the configuration of the platform and services at runtime. In addition, it might be used to fix some associated problems with the platform and to install new SmartObjects services or to make an update on these services. Figure 93 and Figure 94 show screenshots of the web SmartObject Management Portal.
4.5.4.2.2. Operations
The following operations are provided by the SmartObject Management Portal functional component.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registration</td>
<td>Install new services</td>
<td>Service Name (mandatory) URL (mandatory)</td>
</tr>
<tr>
<td>Lifecycle monitoring</td>
<td>STOP/RUN/Update services</td>
<td>Service Name (mandatory) URL (mandatory)</td>
</tr>
<tr>
<td>Authentication</td>
<td>User access policy to the admin interface</td>
<td>User name (mandatory) Password (mandatory)</td>
</tr>
<tr>
<td>Overseeing services and events</td>
<td>Log file generation</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 51: SmartObject Management Portal supported operations

4.5.4.3. Service Monitoring

4.5.4.3.1. Description
Service Monitoring functional component allows administrating the operation of services at runtime. Service Monitoring tasks include basically the tracking of services and the modification of their behavior at runtime.
The OSGi framework gives full control over service bundles: bundles can be installed, uninstalled, started, stopped, and updated). Services running on the BUTLER SmartObject Gateway benefit thus from the service administration operations provided by the OSGi framework. Moreover, services based on the iPOJO component model, benefit of additional service administration operations, in particular for managing services factories and instances.

Service Monitoring use then the basic operations provided by the underlying platform. The Apache Felix Web Console is an example of monitoring service running on OSGi. It allows determining which the installed service bundles are, to start and stop them, to install and uninstall bundles, to see the state of running services, and so on. The web console has an extensible architecture allowing to add specialized monitoring functionalities.

Using the BUTLER Device API, monitoring services can get the description of available devices, collect their provided data (resources), and if needed execute actions on them.

4.5.4.3.2. Interaction with other functional components
Service Monitoring interacts with management functional services in order to re-configure, update or deploy monitored services. They could interact also with contextual provider services in order to determine actions to be performed on services considering the current execution context.

4.5.4.3.3. Operations
Service Monitoring is mainly based on the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Starts the given service</td>
<td>ServiceID (mandatory)</td>
</tr>
<tr>
<td>stop</td>
<td>Stops the given service</td>
<td>ServiceID (mandatory)</td>
</tr>
<tr>
<td>update</td>
<td>Updates the given service</td>
<td>ServiceID (mandatory)</td>
</tr>
</tbody>
</table>

Table 52: Service Monitoring supported operations

The monitoring operations related to resources exposed by devices are based on the operations presented in Table 27.

4.5.4.4. Deployment
Service deployment consists basically on installing and uninstalling packaged services on the runtime platform. The Deployment component uses then the basic operations provided by the underlying platform. For example, the Deployment component running on the BUTLER SmartObject Gateway benefit from the deployment operations provided by the OSGi framework.

During the deployment phase, necessary libraries and services can be automatically looked-for and deployed using a known directory.

4.5.4.4.1. Interaction with other functional components
The Deployment component interacts mainly with the Service Directory components, such as the Service Discovery functional component, in order to look for packaged services and deploy them.

4.5.4.4.2. Operations
The Deployment component provides the following operations:

---

21 The BUTLER SmartObject Gateway relies on the OSGi platform (cf. section [38]).
### 4.5.4.5. Software Update

#### 4.5.4.5.1. Description

Software Update is a functional component, which is handled by the System Management functional group of the BUTLER Services Layer, responsible for monitoring the current software version in each component of the BUTLER network, managing the update files and controlling the software updates. Its main subcomponent is the Firmware Resource, which stores different versions of software/firmware in particular component/device of the network. This Firmware Resource gets the update files from an authenticated user or service developer. Then it stores the update file and retains older stable version in local repository (for rollback operations if needed). With the update command invoked by the authenticated user or service developer, the Firmware Resource can activate the real update process in the desired component/s of the network.

![Software Update model](Image)

Now, the update in SmartObject is explained in more detail. The Firmware Resource is a part of the SmartObject Gateway, which gets the updated files from an authenticated user or administrator via a SmartMobile. Upon the update request from the administrator, the Firmware Resource activates the update command. Next the File Parser gets activated. It parses the update file into the suitable format to be used by the SmartObjects (which might be CoAP, ZigBee, etc.). The update status, which may be success or failure, is monitored in Firmware Resource. The update process on such wireless devices, named Over-The-Air Update (OTAU), is explained as follows.

OTAU refers to methods of distributing new firmware updates, configuration settings and encryption keys to wireless devices without physical access. The high level OTAU architecture is based on ZigBee specification [40]. It mainly consists of the OTAU client, which resides on one, if the end points on every upgradeable device, and the OTAU server, which resides on whichever devices initiates the upgrade process. This might be the coordinator of the ZigBee network. The server side of the OTAU cluster resides on what is commonly referred to as the upgrade access point (UAP). This may be a dedicated physical
device which implements the server-side cluster or a multi-function in-network device which implements the OTAU service as an add-on piece of functionality.

The network protocol for transferring firmware images is shown below:

![Protocol for transferring firmware images](image)

**Figure 96: Protocol for transferring firmware images**

4.5.4.5.1. Interaction with other functional components
Software Update mainly interacts with SmartObject Management portal of System Management, and User Authentication, Network Monitoring, Device Monitoring and IoT Protocol Adapters of Communication layer.

4.5.4.5.2. Operation
The main operation handled by the software update portal is to facilitate service developer or authorized user to upload the new firmware in a resource and update the firmware of the desired device or component of the network.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload</td>
<td>Facilitate service developer or authorized user to upload the new firmware in a firmware resource</td>
<td>New firmware file (format required by the target devices)</td>
</tr>
<tr>
<td>Update</td>
<td>Activate the update process in particular device, SmartObjects</td>
<td>Target device/network address, firmware version</td>
</tr>
</tbody>
</table>
4.5.4.6. Data Management and Marketplace Portal

4.5.4.6.1. Description
This functional component provides an interface for users to manage Virtual Entities, Data Sources and Data Offerings.

BUTLER addresses the requirements to mediate the exposition of data generated by BUTLER SmartObjects so that third-party developers are enabled to create new services and applications on top of BUTLER. It does so by means of a horizontal BUTLER SmartServer architecture that mediates between SmartObjects and applications. Thus, Service Developers can discover and use data from SmartObjects through a single access point. Two main roles are thus defined: the aforementioned Service Developer and the so-called Device Owner:

- **Device Owners** are the actual owners (or administrators) of SmartObjects. They are the ones that make a decision on sharing with third-parties the information coming from their SmartObjects. Device Owners can use the Data Management and Marketplace Portal to manage Virtual Entities, associate Data Sources to Entities and create Data Offerings for subsequent selling.
- Said third parties, **Service Developers**, are in turn those wishing to use data coming from SmartObjects to create new services and applications. Service Developers can use the Data Management and Marketplace Portal to search suitable Data Offerings and purchase them. They can also define so-called Business Events. These are a set of conditions on SmartObject data; when the conditions are evaluated to true the user receives a notification.

The Data Management and Marketplace Portal works usually in close cooperation with the Data Discovery functional component, providing the interface for users to interact with the Data Discovery functional component.

4.5.4.6.2. Interaction with other functional components
The following functional relate explicitly to the Data Management and Marketplace Portal:
- Device Data Collector / Actuator
- Complex Event Processing engine
- Data Discovery
- Resource Exposition (both the Pub/Sub Broker and the Pull Interface)

4.5.4.6.3. Operations
The following operations are provided by the Data Management and Marketplace Portal functional component:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve Entity</td>
<td>Description</td>
<td>Two possibilities:</td>
</tr>
<tr>
<td>Description</td>
<td>Retrieval of the description of an existing Entity given its identifier, the identifier of its owner or a base location</td>
<td>• EntityID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Or:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UserID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location (shape or radius is optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Return value</strong>: Entity Descriptor (graphically shown)</td>
</tr>
<tr>
<td>Create New Entity</td>
<td>Creation of a new Entity</td>
<td>At least one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Entity Name (optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Entity Type (optional)</td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Update Entity Description</td>
<td>Update of an existing Entity description</td>
<td>• Entity Description (optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Entity Owner (mandatory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Entity Coordinates (optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Entity tags (optional)</td>
</tr>
<tr>
<td>Remove Entity</td>
<td>Removal of an existing Entity given its identifier or the identifier of its owner</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• EntityID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UserID</td>
</tr>
<tr>
<td>Create Business Event</td>
<td>Creation of a new Business Event</td>
<td>Set of conditions and actions including some DataSourceID(s)</td>
</tr>
<tr>
<td>Update Business Event</td>
<td>Update of the definition of an existing Business Event</td>
<td>• BusinessEventID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New set of conditions and/or action</td>
</tr>
<tr>
<td>Retrieve Business Event</td>
<td>Retrieval of the definition of an existing Business Event</td>
<td>• BusinessEventID</td>
</tr>
<tr>
<td>Delete Business Event</td>
<td>Removal of an existing Business Event given its identifier or the identifier of its owner</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• BusinessEventID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UserID</td>
</tr>
<tr>
<td>Retrieve Data Source</td>
<td>Retrieval of the description of Data Sources given its identifier(s), the</td>
<td>Two possibilities:</td>
</tr>
<tr>
<td>Description</td>
<td>identifier of its owner or a base location</td>
<td>• DataSourceID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Or:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UserID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Location (shape or radius is optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return value:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data Source Descriptor (graphically shown)</td>
</tr>
<tr>
<td>Update Data Source</td>
<td>Update of the definition of an existing Data Source</td>
<td>At least one of the following:</td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td>• DataSource Name (optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• EntityID(s) (optional, it describes the Entities Data Sources are associated to)</td>
</tr>
<tr>
<td>Search Data Offerings</td>
<td>Search of Data Offerings</td>
<td>Searching arguments:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Data Offering owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Data Offering Location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Data Offering Tags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Data Offering price</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Data Offering service conditions</td>
</tr>
<tr>
<td>Retrieve Data Offering</td>
<td>Retrieval of the description of Data Offering given its identifier(s), or</td>
<td>Two possibilities:</td>
</tr>
<tr>
<td>Description</td>
<td>the identifier of its owner</td>
<td>• DataOfferingID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Or:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UserID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return value:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data Offering Descriptor</td>
</tr>
</tbody>
</table>
### 4.5.4.7. Context Management Portal

#### 4.5.4.7.1. Description

This functional component provides an interface for users to declare context attributes associated to Virtual Entities. The user of this management functionality will be Service Developers (users wishing to use data coming from SmartObjects to create new services and applications). Besides providing an interface towards the user, the Context Management Portal enables the user to pick up relevant Entities and Data Sources and, by means of a set of predefined rules, declare which attributes the context of a given Entity is made of, and how the value of said attributes is computed from the available Data Sources. Next, it translates said context declaration into CEP applications that are handled by the Context / Behavior Information Provider (based on a CEP engine). To sum up, this functional component handles the following process: Data Sources selection → context rules declaration → deployment to the Context / Behavior Information Provider (a CEP engine).

The Context Management Portal also works usually in close cooperation with the Data Discovery functional component, providing the interface for users to interact with the Data Discovery functional component.

#### 4.5.4.7.2. Interaction with other functional components

The following functional relate explicitly to the Context Management Portal:
- Device Data Collector / Actuator
- Context / Behavior Information Provider (Complex Event Processing engine)
- Data Discovery
- Context Exposition (Pub/Sub Broker)

#### 4.5.4.7.3. Operations

The following operations are provided by the Context Management Portal functional component:

<table>
<thead>
<tr>
<th>Operation Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create New Data Offering</td>
<td>Creation of a new Data Offering</td>
</tr>
<tr>
<td>Update Data Offering Description</td>
<td>Update of the definition of an existing Data Offering</td>
</tr>
<tr>
<td>Remove Data Offering</td>
<td>Removal of an existing Data Offering given its identifier or the identifier of its creator (it deletes all the Data Offerings defined by the user)</td>
</tr>
<tr>
<td>Purchase Data Offering</td>
<td>Purchase of an specific Data Offering</td>
</tr>
<tr>
<td>Retrieve Purchased Data Offerings</td>
<td>Retrieval of all the Data Offerings purchased by a given user</td>
</tr>
<tr>
<td>Remove Purchased Data Offering</td>
<td>Removal of a Data Offering already purchased.</td>
</tr>
</tbody>
</table>

**Table 55: Data Management and Marketplace Portal supported operations**
<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
<th>Arguments</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve Entity Description</td>
<td>Retrieval of the description of an existing Entity given its identifier, the identifier of its owner or a base location</td>
<td>Two possibilities: • EntityID Or: • UserID Location (shape or radius is optional)</td>
<td>Entity Descriptors (graphically shown)</td>
</tr>
<tr>
<td>Retrieve Data Source Description</td>
<td>Retrieval of the description of Data Sources given its identifier(s), the identifier of its owner or a base location</td>
<td>Two possibilities: • DataSourceID Or: • UserID Location (shape or radius is optional)</td>
<td>Data Source Descriptors (graphically shown)</td>
</tr>
<tr>
<td>Create Attribute declaration</td>
<td>Declaration of a new Entity attribute</td>
<td>• EntityID (mandatory) • Context Attribute Name (mandatory) • Context Attribute Type (optional) • Context Attribute declaration (set of expressions including some DataSourceID(s))</td>
<td></td>
</tr>
<tr>
<td>Update Attribute declaration</td>
<td>Update of the declaration of an attribute</td>
<td>• ContextAttributeID New set of expressions</td>
<td></td>
</tr>
<tr>
<td>Delete Attribute declaration</td>
<td>Removal of an existing attribute declaration given its identifier</td>
<td>• ContextAttributeID</td>
<td></td>
</tr>
</tbody>
</table>

Table 56: Context Management Portal supported operations

4.6. MATCHING OF BUTLER REQUIREMENTS TO ARCHITECTURE

4.6.1. SmartHome – Monitoring and controlling home appliances

The solutions of Ericsson Spain, ST Microelectronics and IHL are connected to the cloud publishing their data and control API to the IHL control and visualization server. This server accesses the gateways and web services in order to fetch data coming from the attached sensors as well as from the prediction algorithms (green lines). It processes the data and finally generates the information to be visualized and sent to the TV as well as the tablet (red lines). Interaction with the user happens over the tablet.
Figure 97: PoC Cluster SmartHome

<table>
<thead>
<tr>
<th>Requirements for the use case</th>
<th>Butler Architecture - Functional component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connecting ZigBee devices to IHL Gateway, ST gateway</td>
<td>IoT Protocol Adapters - Communications Layer</td>
</tr>
<tr>
<td>Between Donald’s SmartMobile (tablet, TV) to the IHL visualization server</td>
<td>User Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>IHL server, Ericsson server, BUTLER cloud</td>
<td>Server Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Monitor components and network parameters</td>
<td>Network Monitoring / Device Monitoring - Communications Layer</td>
</tr>
<tr>
<td>BUTLER SmartServer discovers and authenticates the sensors and other SmartObjects via IHL Gateway, ST Gateway</td>
<td>Device Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Authenticate Donald</td>
<td>User Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Discover IHL Server, ERC server</td>
<td>Server Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Connecting ZigBee devices to IHL Gateway, ST gateway</td>
<td>IoT Protocol Adapters - Communications Layer</td>
</tr>
</tbody>
</table>

|                                                                                              |                                                                                   |
| Gathering the necessary data from Donald’s home, from sensors of ERC                        | Device Data Collector - Data / Context Management Layer                           |
| Keep track of location of Donald                                                            | Localization Manager - Data / Context Management Layer                            |
| Gathers the Donald’s behavior modeled data                                                  | User Behavior Capture - Data / Context Management Layer                           |
| Provide contextual information from Donald’s activities, the used devices and applications  | Context / Behavior Information Provider - Data / Context Management Layer         |
| Extracts Donald’s profile from all raw data                                                 | User Profile Manager - Data / Context Management Layer                            |
| Identify conclusive events out of collected data like best time to go for Holiday in Chalet | Event Processing - Data / Context Management Layer                                |
Send notification to Donald about such conclusive events                   Generic Notification / Actuation - Data / Context Management Layer

| User profile exposition, behavior exposition, context exposition, resource exposition | Data Exposition Services - Services Layer |
| Configuration of SmartObjects, monitoring, software updates | System Management - Services Layer |
| Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services | Data and Service Directory - Services Layer |
| Temperature monitoring services, Alarm services, Energy monitoring services. | Built-in Context-Aware Services and Applications - Services Layer |

### 4.6.2. SmartHome – Multimedia everywhere

This demonstrator consists of a typical multiroom audio / video environment, consisting in at least two rooms, each of them being equipped with a followme-capable multimedia player, a TV set, and presence of location sensors. The presence of the user in one room is detected, and as soon as he leaves the room, the system put the playing video in pause, power off the TV set in this room, power on the TV set in the other room, and waits for the user to enter the second room to resume the playing video.

![Diagram of SmartHome Multimedia everywhere](image)

**Figure 98: SmartHome Multimedia everywhere**

<table>
<thead>
<tr>
<th>Requirements for the use case</th>
<th>Butler Architecture - Functional component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linking media renderer with USB / Bluetooth / 802.15.4 / 6LoWPan</td>
<td>IoT Protocol Adapters - Communications Layer</td>
</tr>
<tr>
<td>Manage connectivity of Alice at different rooms</td>
<td>Butler User Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Manage media server and internet connections</td>
<td>Server Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Monitor network parameters of user device</td>
<td>Network Monitoring / Device Monitoring - Communications Layer</td>
</tr>
<tr>
<td>Discover and authenticate the user device</td>
<td>Device discovery / Authentication /</td>
</tr>
</tbody>
</table>
(smart phone / sensors) | Authorization server - Communications Layer authentication
---|---
Authenticate user | User Discovery and Authentication - Communications Layer
Discover and authenticate media servers and media renders | Server Discovery and Authentication - Communications Layer

Assemble the necessary data from ZigBee sensors, 6LoWPAN sensors | Device Data Collector - Data / Context Management Layer
Keep track of user location | Localization Manager - Data / Context Management Layer
Gather the Donald’s behavior modeled data | User Behavior Capture - Data / Context Management Layer
Provide contextual information from user activities, the used devices and applications | Context / Behavior Information Provider - Data / Context Management Layer
Extracts user profile from all raw data | User Profile Manager - Data / Context Management Layer
Identify conducive events out of collected data like user moved to another room | Event Processing - Data / Context Management Layer
Send notification to decided room’s media renderer to resume the playing video | Generic Notification / Actuation - Data / Context Management Layer

Presence or user in particular room can be exposed by other applications | Data Exposition Services - Services Layer
User login and registration, authentication | User Portal - Services Layer
Configuration of sensors, user device monitoring, software updates | Management Portal - Services Layer
Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services | Data and Service Directory - Services Layer

### 4.6.3. SmartHealth – Personalized diabetes self-healthcare

This Proof of Concept (PoC) allows a patient diagnosed with diabetes to manage its health condition by means of a context-aware diabetes diary running on a mobile handheld. The added value of this proof of concept is the personalized recommendations based on the typical context of the user.
Figure 99: SmartHealth personalized diabetes self-healthcare

<table>
<thead>
<tr>
<th>Requirements for the use case</th>
<th>Butler Architecture - Functional component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common bridge between Bluetooth (PolyMap Wireless for glucometer), Fitbit (accelerometer), android smartphone</td>
<td>IoT Protocol Adapters - Communications Layer</td>
</tr>
<tr>
<td>Manage connectivity of these SmartObjects</td>
<td>User Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Manages connectivity to the BUTLER SmartServer</td>
<td>Server Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Monitor network parameters of PolyMap Wireless, Fitbit, and smartphone.</td>
<td>Network Monitoring / Device Monitoring - Communications Layer - Communications Layer</td>
</tr>
<tr>
<td>Discover and authenticates the user device (smart phone / sensors)</td>
<td>Device Discovery / Authentication / Authorization Server - Communications Layer</td>
</tr>
<tr>
<td>Authenticate user</td>
<td>User Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Discover and authenticate servers</td>
<td>Server Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Gathers data from glucometer, accelerometer</td>
<td>Device Data Collector - Data / Context Management Layer</td>
</tr>
<tr>
<td>Keep track of user location</td>
<td>Localization Manager - Data / Context Management Layer</td>
</tr>
<tr>
<td>Gathers the users past and planned activities</td>
<td>User Behavior Capture - Data / Context Management Layer</td>
</tr>
<tr>
<td>Provide contextual information from user activities, the used devices and applications</td>
<td>Context / Behavior Information Provider - Data / Context Management Layer</td>
</tr>
<tr>
<td>Manages user profiles</td>
<td>User Profile Manager - Data / Context Management Layer</td>
</tr>
<tr>
<td>Identify conclusive events out of collected data and decide if insulin dosage necessary</td>
<td>Event Processing - Data / Context Management Layer</td>
</tr>
<tr>
<td>Send notification to if insulin dosage is necessary</td>
<td>Generic Notification / Actuation - Data / Context Management Layer</td>
</tr>
<tr>
<td>User glucose level, activities can be exposed by other smart health use cases</td>
<td>Data Exposition Services - Services Layer</td>
</tr>
<tr>
<td>Configuration of SmartObjects, monitoring, software updates</td>
<td>System Management - Services Layer</td>
</tr>
<tr>
<td>Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services</td>
<td>Data and Service Directory - Services Layer</td>
</tr>
<tr>
<td>Suggest insulin dosage</td>
<td>Built-in Context-Aware Services and Applications - Services Layer</td>
</tr>
</tbody>
</table>

4.6.4. SmartCity – Smart parking space management

The SmartParking PoC allows citizens to interact with the parking spaces in a smarter way by using a regular smartphone and NFC tags. In this way, citizens can make reservations and payments for a controlled parking space. Along with some context-aware dynamic parking allocation techniques, these functionalities enrich the user experience while decrease the amount of time needed to find a free parking space.
Figure 100: SmartParking space management

<table>
<thead>
<tr>
<th>Requirements for the use case</th>
<th>Butler Architecture - Functional component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common bridge between NFC module, GPRS module</td>
<td>IoT Protocol Adapters - Communications Layer</td>
</tr>
<tr>
<td>Manages connectivity of SmartObjects</td>
<td>User Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Manages connectivity to the BUTLER SmartServer</td>
<td>Server Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Monitor network parameters of NFC module, GPRS module</td>
<td>Network Monitoring / Device Monitoring - Communications Layer</td>
</tr>
<tr>
<td>Discover and authenticates the user smartphone / NFC module</td>
<td>Device Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Authenticate user</td>
<td>User Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Discover and authenticate servers</td>
<td>Server Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Collect parking space data</td>
<td>Device Data Collector - Data / Context Management Layer</td>
</tr>
<tr>
<td>Keep track of user location</td>
<td>Localization Manager - Data / Context Management Layer</td>
</tr>
<tr>
<td>Gather the users activities</td>
<td>User Behavior Capture - Data / Context Management Layer</td>
</tr>
<tr>
<td>Provide contextual information from user activities, the used devices and applications</td>
<td>Context / Behavior Information Provider - Data / Context Management Layer</td>
</tr>
<tr>
<td>Manage user profiles</td>
<td>User Profile Manager - Data / Context Management Layer</td>
</tr>
<tr>
<td>Identify conclusive events out of collected data to decide if user is close to available parking space</td>
<td>Event Processing - Data / Context Management Layer</td>
</tr>
<tr>
<td>Notify user if parking is available</td>
<td>Generic Notification / Actuation - Data / Context Management Layer</td>
</tr>
</tbody>
</table>
4.6.5. SmartShopping – Context-aware sparkdeals

The PoC integrates context information and user parameters so as to create a very precise advertisement tool. The main idea is to analyze the different possibilities to distribute selectively the content generated by commerce. The viralization of marketing campaigns is also a key element of the PoC. Next figure represents the interaction with application in terms of production, consumption and processing of data.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Butler Architecture - Functional component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between smartphone, GPS, internet</td>
<td>IoT Protocol Adapters - Communications Layer</td>
</tr>
<tr>
<td>Manages connectivity of smartphone and other devices</td>
<td>User Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Function</td>
<td>Layer</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Monitor network parameters</td>
<td>Server Connectivity Manager - Communications Layer</td>
</tr>
<tr>
<td>Discovers and authenticates the user smartphone</td>
<td>Network Monitoring / Device Monitoring - Communications Layer</td>
</tr>
<tr>
<td>Authenticates user</td>
<td>Device Discovery and authentication - Communications Layer</td>
</tr>
<tr>
<td>Discover and authenticate servers</td>
<td>User Discovery and Authentication - Communications Layer</td>
</tr>
<tr>
<td>Collects GPS data, navigation information related ads, city schedule</td>
<td>Device Data Collector - Data / Context Management Layer</td>
</tr>
<tr>
<td>Location system used on demand, user can disable it</td>
<td>Localization Manager - Data / Context Management Layer</td>
</tr>
<tr>
<td>Gathers the users’ activities, interests, history of used products.</td>
<td>User Behavior Capture - Data / Context Management Layer</td>
</tr>
<tr>
<td>Provide contextual information from user activities, the used devices and applications</td>
<td>Context / Behavior Information Provider - Data / Context Management Layer</td>
</tr>
<tr>
<td>Manage and update user profiles</td>
<td>User Profile Manager - Data / Context Management Layer</td>
</tr>
<tr>
<td>Identify conclusive events out of collected data – decide related ads and inform commerce about events that can increase visitors.</td>
<td>Event Processing - Data / Context Management Layer</td>
</tr>
<tr>
<td>Push notifications / On demand access to information</td>
<td>Generic Notification / Actuation - Data / Context Management Layer</td>
</tr>
<tr>
<td>Location service of BUTLER can be exploited here</td>
<td>Data Exposition Services - Services Layer</td>
</tr>
<tr>
<td>Configuration of devices, monitoring, software updates</td>
<td>System Management - Services Layer</td>
</tr>
<tr>
<td>Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services</td>
<td>Data and Service Directory - Services Layer</td>
</tr>
<tr>
<td>Context based ads</td>
<td>Built-in Context-Aware Services and Applications - Services Layer</td>
</tr>
</tbody>
</table>

### 4.6.6. SmartTransport – Safe transportation of school kids

The selected SmartTransport service is planned to offer aid to school-buses and transportation of kids with public transportation. A tag with identification number, as form of key-chain necklace, is provided to all the kids that are allowed to get on/off the bus/other public transportation. The accompanying person (e.g. teacher) knows how many children are supposed to be in/out when the bus/tram/train is ready to close the doors and intends to leave. A very accurate indoor localization system is therefore able to localize exactly all the tags inside/outside the bus (all members of the group need to be in or out respectively) and it allows for the doors to be closed only when all the kids are safety in or out with some minimum distance from the doors. It is clear as such system must be very accurate: it is compulsory to avoid that a kid approaching the bus, or even worse, stepping on the entrance stairs, would be closed off by the doors as result of a few cm. localization errors.

<table>
<thead>
<tr>
<th>Requirements for the use case</th>
<th>Butler Architecture - Functional component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link between ZIGPOS modules, tags, gateway / server</td>
<td>IoT Protocol Adapters - Communications Layer</td>
</tr>
</tbody>
</table>
4.7. REVIEWED USE CASES AND REQUIREMENTS

4.7.1. SmartHome – Monitoring and controlling home appliances

The solutions of Ericsson Spain, ST Microelectronics and IHL are connected to the cloud publishing their data and control API to the IHL control and visualization server. This server accesses the gateways and web services in order to fetch data coming from the attached sensors as well as from the prediction algorithms (green lines). It processes the data and finally generates the information to be visualized and sent to the TV as well as the tablet (red lines). Interaction with the user happens over the tablet.
<table>
<thead>
<tr>
<th>Context / Behavior Information Provider</th>
<th>Provide contextual information from Donald’s activities, the used devices and applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent Storage</td>
<td>To keep a cached copy of information generated by SmartObjects within a SmartServer instance</td>
</tr>
<tr>
<td>Resource Access</td>
<td>Independent resource access</td>
</tr>
<tr>
<td>User Portal</td>
<td>Manage purchased data sources to user and associated context</td>
</tr>
<tr>
<td>User Profile Manager</td>
<td>Extracts Donald’s profile from all raw data</td>
</tr>
<tr>
<td>Simple / Complex Event Processing</td>
<td>Identify conclusive events out of collected data like best time to go for Holiday in Chalet</td>
</tr>
<tr>
<td>Generic Notification / Actuation</td>
<td>Send notification to Donald about such conclusive events</td>
</tr>
</tbody>
</table>

### Services Layer:

<table>
<thead>
<tr>
<th>Data Exposition Services</th>
<th>User profile exposition, behavior exposition, context exposition, resource exposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Management</td>
<td>Management, configuration of SmartObjects, service deployment, monitoring, software updates</td>
</tr>
<tr>
<td>Data and Service Directory</td>
<td>Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services</td>
</tr>
<tr>
<td>Built-in Context-Aware Services and Applications</td>
<td>Temperature monitoring services, Alarm services, Energy monitoring services</td>
</tr>
</tbody>
</table>

### 4.7.2. SmartHome – Multimedia everywhere

This demonstrator consists of a typical multiroom audio/video environment, consisting in at least two rooms, each of them being equipped with a followme-capable multimedia player, a TV set, and presence of location sensors. The presence of the user in one room is detected, and as soon as he leaves the room, the system put the playing video in pause, power off the TV set in this room, power on the TV set in the other room, and waits for the user to enter the second room to resume the playing video.

### Communications Layer:

<table>
<thead>
<tr>
<th>IoT Protocol Adapters</th>
<th>Between media renderer and USB / Bluetooth / 802.15.4 / 6LoWpan</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Connectivity Manager</td>
<td>Manages connectivity of Alice at different rooms</td>
</tr>
<tr>
<td>Server Connectivity Manager</td>
<td>Manages media server and internet connections</td>
</tr>
<tr>
<td>Network Monitoring / Device monitoring</td>
<td>Monitor network parameters of user device</td>
</tr>
<tr>
<td>Device Discovery / Authentication / Authorization server</td>
<td>Discovers and authenticates the user device (smart phone / sensors)</td>
</tr>
<tr>
<td>User Discovery and Authentication</td>
<td>Authenticates user</td>
</tr>
<tr>
<td>Server Discovery and Authentication</td>
<td>Discover and authenticate media servers and media renders</td>
</tr>
</tbody>
</table>

### Data / Context Management Layer:

<table>
<thead>
<tr>
<th>Device Data Collector / Actuator</th>
<th>Gathers the necessary data from ZigBee sensors, 6LoWPAN sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localization Manager</td>
<td>Keep track of user location</td>
</tr>
</tbody>
</table>
User Behavior Capture | Gathers the Donald’s behavior modeled data  
Context / Behavior Information Provider | Provide contextual information from Donald’s activities, the used devices and applications  
Persistent Storage | To keep a cached copy of information generated by SmartObjects within a SmartServer instance  
Resource Access | Independent resource access  
User Portal | Manage purchased data sources to user and associated context  
User Profile Manager | Extracts user profile from all raw data  
Event Processing | Identify conclusive events out of collected data – user moved to another room  
Generic Notification / Actuation | Send notification to decided room’s media renderer to resume the playing video  

Services Layer:

| Data Exposition Services | Presence or user in particular room can be exposed by other applications  
System Management | Configuration of sensors, user device monitoring, software updates  
Data and Service Directory | Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services  
Built-in Context-Aware Services and Applications | Localization Service  

4.7.3. SmartHealth – Personalized diabetes self-healthcare

This Proof of Concept (PoC) allows a patient diagnosed with diabetes to manage its health condition by means of a context-aware diabetes diary running on a mobile handheld. The added value of this proof of concept is the personalized recommendations based on the typical context of the user.

Communications Layer:

| IoT Protocol Adapters | Common bridge between Bluetooth (PolyMap Wireless for glucometer), Fitbit (accelerometer), android smartphone  
User Connectivity Manager | Manages connectivity of these SmartObjects  
Server Connectivity Manager | Manages connectivity to the BUTLER SmartServer  
Network Monitoring / Device monitoring | Monitor network parameters of PolyMap Wireless, Fitbit, smartphone  
Device Discovery / Authentication / Authorization server | Discovers and authenticates the user device (smart phone / sensors)  
User Discovery and Authentication | Authenticates user  
Server Discovery and Authentication | Discover and authenticate servers  

Data / Context Management Layer:

| Device Data Collector | Gathers data from glucometer, accelerometer  
User Profile Manager | Manages User Profile. Collects and maintains user data like glucose value, locations, past activities  

© BUTLER – Page 171/191
### Localization Manager
- Keep track of user location

### User Behavior Capture
- Gathers the users past and planned activities

### Context / Behavior Information Provider
- Provide contextual information from user activities, the used devices and applications

### Persistent Storage
- To keep a cached copy of information generated by SmartObjects within a SmartServer instance

### Resource Access
- Independent resource access

### User Portal
- Manage purchased data sources to user and associated context

### Event Processing
- Identify conclusive events out of collected data – decide if insulin dosage necessary

### Generic Notification / Actuation
- Send notification to if insulin dosage is necessary

### Services Layer:

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Exposition Services</td>
<td>User glucose level, activities can be exposed by other smart health use cases</td>
</tr>
<tr>
<td>System Management</td>
<td>Configuration of SmartObjects, monitoring, software updates</td>
</tr>
<tr>
<td>Data and Service Directory</td>
<td>Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services</td>
</tr>
<tr>
<td>Built-in Context-Aware Services and Applications</td>
<td>Suggest insulin dosage</td>
</tr>
</tbody>
</table>

### 4.7.4. SmartCity – Smart parking space management

The SmartParking PoC allows citizens to interact with the parking spaces in a smarter way by using a regular smartphone and NFC tags. In this way, citizens can make reservations and payments for a controlled parking space. Along with some context-aware dynamic parking allocation techniques, these functionalities enrich the user experience while decrease the amount of time needed to find a free parking space.

### Communications Layer:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoT Protocol Adapters</td>
<td>Common bridge between NFC module, GPRS module</td>
</tr>
<tr>
<td>User Connectivity Manager</td>
<td>Manages connectivity of SmartObjects</td>
</tr>
<tr>
<td>Server Connectivity Manager</td>
<td>Manages connectivity to the BUTLER SmartServer</td>
</tr>
<tr>
<td>Network Monitoring / Device monitoring</td>
<td>Monitor network parameters of NFC module, GPRS module</td>
</tr>
<tr>
<td>Device Discovery / Authentication / Authorization server</td>
<td>Discovers and authenticates the user smartphone / NFC module</td>
</tr>
<tr>
<td>User Discovery and Authentication</td>
<td>Authenticates user</td>
</tr>
<tr>
<td>Server Discovery and Authentication</td>
<td>Discover and authenticate servers</td>
</tr>
<tr>
<td>Persistent Storage</td>
<td>To keep a cached copy of information generated by SmartObjects within a SmartServer instance</td>
</tr>
<tr>
<td>Resource Access</td>
<td>Independent resource access</td>
</tr>
<tr>
<td>User Portal</td>
<td>Manage purchased data sources to user and associated context</td>
</tr>
</tbody>
</table>
Data / Context Management Layer:

<table>
<thead>
<tr>
<th>Device Data Collector</th>
<th>Collect parking space data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localization Manager</td>
<td>Keep track of user location</td>
</tr>
<tr>
<td>User Behavior Capture</td>
<td>Gathers the users activities</td>
</tr>
<tr>
<td>Context / Behavior Information Provider</td>
<td>Provide contextual information from user activities, the used devices and applications</td>
</tr>
<tr>
<td>User Profile Manager</td>
<td>Manages user profiles</td>
</tr>
<tr>
<td>Event Processing</td>
<td>Identify conclusive events out of collected data – decide if user is close to available parking space</td>
</tr>
<tr>
<td>Generic Notification / Actuation</td>
<td>Notify user if parking is available</td>
</tr>
</tbody>
</table>

Services Layer:

<table>
<thead>
<tr>
<th>Data Exposition Services</th>
<th>Location service of BUTLER can be exploited here</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Portal</td>
<td>User login and registration, authentication</td>
</tr>
<tr>
<td>Management Portal</td>
<td>Configuration of SmartObjects, monitoring, software updates</td>
</tr>
<tr>
<td>Data and Service Directory</td>
<td>Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services</td>
</tr>
<tr>
<td>Built-in Context-Aware Services and Applications</td>
<td>Dynamic Parking availability</td>
</tr>
</tbody>
</table>

4.7.5. SmartShopping – Context-aware sparkdeals

The PoC integrates context information and user parameters so as to create a very precise advertisement tool. The main idea is to analyze the different possibilities to distribute selectively the content generated by commerce. The viralization of marketing campaigns is also a key element of the PoC.

Communications Layer:

<table>
<thead>
<tr>
<th>IoT Protocol Adapters</th>
<th>Between smartphone, GPS, internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Connectivity Manager</td>
<td>Manages connectivity of SmartMobile and other SmartObjects</td>
</tr>
<tr>
<td>Server Connectivity Manager</td>
<td>Manages connectivity to the BUTLER SmartServer</td>
</tr>
<tr>
<td>Network Monitoring / Device monitoring</td>
<td>Monitor network parameters</td>
</tr>
<tr>
<td>Device Discovery / Authentication / Authorization server</td>
<td>Discovers and authenticates the user smartphone</td>
</tr>
<tr>
<td>User Discovery and Authentication</td>
<td>Authenticates user</td>
</tr>
<tr>
<td>Server Discovery and Authentication</td>
<td>Discover and authenticate servers</td>
</tr>
</tbody>
</table>

Data / Context Management Layer:

<table>
<thead>
<tr>
<th>Device Data Collector</th>
<th>Collects GPS data, navigation information related ads, city schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localization Manager</td>
<td>Location system used on demand, user can disable it</td>
</tr>
<tr>
<td>User Behavior Capture</td>
<td>Gathers the users’ activities, interests, history of used products.</td>
</tr>
</tbody>
</table>
Context / Behavior Information Provider | Provide contextual information from Donald’s activities, the used devices and applications
---|---
Persistent Storage | To keep a cached copy of information generated by SmartObjects within a SmartServer instance
Resource Access | Independent resource access
User Portal | Manage purchased data sources to user and associated context
User Profile Manager | Manage and update user profiles
Event Processing | Identify conclusive events out of collected data – decide related ads and inform commerce about events that can increase visitors.
Generic Notification / Actuation | Push notifications / On demand access to information

**Services Layer:**

| Data Exposition Services | Location service of BUTLER can be exploited here
| System Management | Configuration of SmartObjects, monitoring, software updates
| Data and Service Directory | Registration of services in the BUTLER network, creation of services, dynamic management of the services, discovery of services
| Built-in Context-Aware Services and Applications | Context based ads

**4.7.6. SmartTransport – Safe transportation of school kids**

The selected SmartTransport service is planned to offer aid to school buses and transportation of kids with public transportation. A tag with identification number, as form of key-chain necklace, is provided to all the kids that are allowed to get on/off the bus/other public transportation. The accompanying person (e.g. teacher) knows how many children are supposed to be in/out when the bus/tram/train is ready to close the doors and intends to leave. A very accurate indoor localization system is therefore able to localize exactly all the tags inside/outside the bus (all members of the group need to be in or out respectively) and it allows for the doors to be closed only when all the kids are safety in or out with some minimum distance from the doors. It is clear as such system must be very accurate: it is compulsory to avoid situation like closing off the doors as result of a few cm. localization errors when a kid approaches the bus or steps on the entrance stairs.

**Communications Layer:**

| IoT Protocol Adapters | Between ZIGPOS modules, tags, gateway / server
| User Connectivity Manager | Manages connectivity of the module, tags and gateway
| Server Connectivity Manager | Manages connectivity to the BUTLER SmartServer
| Network Monitoring / Device monitoring | Monitor network parameters
| Device Discovery / Authentication / Authorization server | Discovers and authenticates the ta
| User Discovery and Authentication | Authenticate children, teacher
| Server Discovery and Authentication | Discover and authenticate servers
| Persistent Storage | To keep a cached copy of information
<table>
<thead>
<tr>
<th>Resource Access</th>
<th>Independent resource access</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Portal</td>
<td>Manage purchased data sources to user and associated context</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Data / Context Management Layer:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device Data Collector</strong></td>
</tr>
<tr>
<td><strong>Localization Manager</strong></td>
</tr>
<tr>
<td><strong>User Behavior Capture</strong></td>
</tr>
<tr>
<td><strong>Context / Behavior Information Provider</strong></td>
</tr>
<tr>
<td><strong>Persistent Storage</strong></td>
</tr>
<tr>
<td><strong>Resource Access</strong></td>
</tr>
<tr>
<td><strong>User Portal</strong></td>
</tr>
<tr>
<td><strong>User Profile Manager</strong></td>
</tr>
<tr>
<td><strong>Event Processing</strong></td>
</tr>
<tr>
<td><strong>Generic Notification / Actuation</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Services Layer:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Exposition Services</strong></td>
</tr>
<tr>
<td><strong>System Management</strong></td>
</tr>
<tr>
<td><strong>Data and Service Directory</strong></td>
</tr>
<tr>
<td><strong>Built-in Context-Aware Services and Applications</strong></td>
</tr>
</tbody>
</table>
5. INITIAL PERVERSIVE BUTLER PROOF-OF-CONCEPT

5.1. SMARTHOME SCENARIO (ENERGY FORECASTING)

The PoC-review tells a few scenes out of the life of Donald and Daisy from the storyline created by Swisscom for BUTLER [1].

Donald is 51 years old, works at a bank and is married with Daisy. They have 2 children. They live in a nice house in the suburbs and also own a chalet in the mountains. Donald enjoys eating out, despite his food allergies and he is interested in energy efficiency.

Daisy is 45 years old, a housewife and part-time shop assistant. She is married to Donald and has 2 children. Daisy suffers from diabetes, tries to do fitness workouts regularly and likes shopping with her girlfriends. She is glad that machines support her in her daily household tasks.

![Diagram of SmartHome Storyline]

Figure 102: SmartHome Storyline
The following steps give more detail on the way the energy awareness and forecasting scenarios from SmartHome are going to be used:

- Donald is at home and takes a look at his information panel on the TV.
- He switches from the “home view” to the “holiday house” view.
- He sees the (1) weather forecast, the (2) indoor and the (3) outdoor temperature, the (4) current energy consumption, the (5) actual production and the (6) currently used power source delivering the energy for his holiday house.
- He sees tells BUTLER to cool down the house to 22 °C for tomorrow when he reaches the destination at 13:00 Uhr.  
- When he leaves the house for his holidays, all unused energy consumers are turned off.
- As sometimes, it also gets cold in the holiday house (Donald goes there also in winter time now) he brings a new SmartObject: An old heater he has found in the cellar. When he plugs it in and turns it on, BUTLER automatically recognizes the new load and tells him that he should replace it immediately.
- He points the phones’ camera at it. The phone allows to exactly identifying the type and model of the SmartObject. BUTLER uses this information to recommend a newer SmartObject doing the job with a higher efficiency.

---

22 To make this work, the Ericsson forecast service, the solar system from FBC, the temperature sensor from ST-I and the server as well as the user interface from iHL have to work seamlessly together.

23 For this functionality the load recognition system using the KUL framework, the iHL plugs and server application have to be integrated.

24 TILs AR application and iHLS server work together here.
5.2. SMARTHOME SCENARIO (FOLLOW ME)

The principle of the Media Follow-Me Application is to allow a house occupant playing an audio or video file on a multimedia SmartObject, located in his/her current location (e.g. a smart TV at the living room), and to follow him/her by transferring the current media reproduction to an available SmartObject located in his/her new location (e.g. a tablet PC at the kitchen). To do so, this application uses two Smart Services: a Localization Service which observes changes in the user location by using motion detection sensors (e.g. PIR and ultrasonic sensors); and a Multimedia Service which manages different multimedia SmartObjects (such as smart televisions, tablet PCs, speakers) in order to control the reproduction of media files. A use example of this application is presented in the figure below.

![Diagram of components and services of the Followme application. TV Samsung and TV Lg are examples of TV devices. They run a SmartTV application that actually implements generic Butler TV services. However the implementation of the SmartTV application may differ from one manufacturer to another.](image)

Figure 104: Use case example: Multimedia Follow me application

Figure 104 shows the diagram of components and services of the Followme application. TV Samsung and TV Lg are examples of TV devices. They run a SmartTV application that actually implements generic Butler TV services. However the implementation of the SmartTV application may differ from one manufacturer to another.

The gateway runs the following OSGi bundles:

- SamsungTvBridge manages all Samsung TV devices. It discovers them, and creates an instance of ButlerTvService for each one. Each instance of ButlerTvService communicates bi-directionally with the SmartTV application, written in JavaScript and running in an HTML5 browser. ButlerTvService is the interface used by the Butler applications, either other local OSGi bundles or any external application.
- LgTvBridge is the LG implementation of the Tv bridge, exactly as the SamsungTvBridge is. It creates an instance of ButlerTvService for each LG television.
- MultiMediaService (MMS) presented in section 4.5.3.2.
Even if the Media Follow-Me Application is developed to run on a local server, physically on the same machine of the gateway and the interactions between the application and the services are JAVA/OSGi service calls, different services used in this application can be used by remote services using HTTP.

5.3. SMARTPARKING SCENARIO

For the last ten years, dramatic developments have been observed in new ICT technologies such as RFID, GPS, NFC, and 3G, to name only the most popular. With the development of these sensing, wireless communication, and Internet technologies, we are now living in a world where every day ordinary objects are potentially connected to the Internet. These recent advances open a huge field of new applications where “smart things” are all interconnected into a full ecosystem: the internet of things (IoT).
In the scope of parking management, on one hand we have traditional systems that are characterized for random searches of parking spaces around the destination point, what means longer searching times thus more traffic congestion due to drivers looking for a spot. This results in an increment of air pollution levels. On the other hand, in SmartCities the existing progress in sensor technologies gives a great opportunity to take action in both environmental concerns and citizens’ Quality of Life. Actual trends in new parking solutions are often based in a sensing infrastructure for reporting the occupancy of parking lots within the city. But, we think that on top of that there’s still room for new parking related services.

For developing an evolved SmartParking solution we have studied a bunch of key features and classified them in 3 categories:

- The ones focused in final users
- The ones targeting the whole system management
- And the rest of features related with on-street deployments.

Considering the last group, besides deploying a monitoring infrastructure we propose to display the parking availability to citizens directly on the field by installing a sort of lighting system, for instance using LEDs with different colors: red means busy and green free, while blue denotes a parking space that is reserved and orange (or yellow) means that the space is reserved for being used by load/unload vehicles. In addition, we provide a mobile application with a set of functionalities depending on the user profile. Just as an example, policemen (or local authorities) could be given more privileges at the time of booking certain spots rather than regular citizens. All of these features are being controlled by a Content Management System, which is also in charge of collecting and processing context information of the city. For us, context information is any data that can be used to characterize the situation of an entity. And what’s an entity? An entity is a person, a place, or an object, that is considered relevant for the interactions between the users and the application (including the user and the application themselves). So these are our goals for designing a system that goes beyond state of the art parking approaches.
The following figure describes the different elements that compose the system architecture.

![Figure 107: SmartParking High-Level Architecture](image)

In these 3 green boxes are presented the key elements defined by the BUTLER project. As stated by the project consortium, this SmartPlatforms define roles, not physical implementations:

- The SmartObject is the interface to the physical world
- The SmartMobile is the interface to the user
- And the SmartServer is the interface to the digital world

Concerning the SmartParking system, the SmartObject is the element providing M2M connectivity from other system elements to the physical world. The SmartMobile is the SmartObject owned by the final user, which by means of a mobile parking application is provided with a GUI. This application is fed with relevant functionalities provided by the SmartServer, which handles high computing capabilities, many memory resources and can communicate with any network protocol standard.

Finally, using the Message Sequence Chart presented in Figure 108, we are going to explain how the system behaves during the procedure of:

- Reserving a parking space.
- Parking the car
- And finishing (checking-in) the reservation
The process starts with the exchange of the parking space availability in the whole city. The user requests to the server the bunch of parking opportunities available in a certain location. The user’s SmartMobile sends then a request to the SmartServer. Upon reception, this request is combined with relevant context information and selected algorithms process the output as a list of available parking spaces. Once the schedule time is approaching, the user drives to the select parking space. After parking, a vehicle detection sensor reports to the SmartObject the presence of a new vehicle, and this one to the SmartServer. The last step of the process consists in confirming the parking reservation by means of a direct user operation that can assure the system that the reservation is being done by a legitimate user, for instance, NFC technology.

5.4. SMARTHEALTH SCENARIO

The “Personalized Diabetes Self-care” proof of concept (PoC) allows a patient diagnosed with diabetes to manage its health condition by means of a context-aware diabetes diary running on a mobile handheld. With smartphones and tablets likely to overtake the PC market in the next couple of years and HTML5 being an enabling technology that promises to work seamlessly across mobile platforms and browsers, Gartner claims in its IT predictions for 2013 report that the future for web application consumption is mobile. The HTML5 application illustrated in Figure 109 is developed with the PhoneGap-based SmartMobile framework.
Figure 109: Diabetes Companion: a HTML5 enabled personalized diabetes self-care application

It runs as a hybrid HTML5 application on Android with direct access to sensors of interest (e.g. the accelerometer). The application also runs with reduced functionality in a browser as certain sensors are not directly available [34].

The specific aim of this application was to not just replace a paper log book with a portable electronic device, but to capture and integrate the relevant user context into the data log in order to better compare measurements with previous situations. The augmented data logs and similarity analysis helped find trends and advise the individual more accurately and tailored to the current situation. The major challenge was to identify classes of activities, such as eating and exercise that have an effect on blood glucose levels. As activities of daily living (ADL) typically present recurring behavioral patterns, we investigated the presence of correlations between time and location on the one hand and types of activities on the other hand, and used these correlations to find similar situations of the past as a recommendation for the patient. The advantage of this approach is that the system offers personalized assistance, not only leveraging the results of the blood glucose sensor itself, but also basing its advice on the current context and anticipating what is likely going to happen next.

Time and location are first class entities in context management. Beyond time and location we use the triaxial accelerometer readily embedded in many smartphones as a pervasive, non-intrusive and mobile sensing unit for activity recognition. However, compared to time and location which are easy to sense directly on a mobile platform, we need fairly more complex and computationally intensive algorithms to classify the activity of the user (standing, walking, running, falling, etc.) and track the number of steps taken each day as a measure for the exercise level and active lifestyle of the user.

This SmartHealth application makes use of the behavior recognition framework to receive feedback about the energy expenditure of the individual. Figure 110 provides an overview of the interaction with the framework and the interactions among them.
5.5. SMARTTRANSPORT SCENARIO

The selected SmartTransport service is planned to offer aid to school-buses and transportation of kids with public transportation. A tag with identification number, as form of key-chain necklace, is provided to all the kids that are allowed to get on/off the bus/other public transportation. The accompanying person (e.g. teacher) knows how many children are supposed to be in/out when the bus/tram/train is ready to close the doors and intends to leave. A very accurate indoor localization system is therefore able to localize exactly all the tags inside/outside the bus (all members of the group need to be in or out respectively) and it allows for the doors to be closed only when all the kids are safety in or out with some minimum distance from the doors. It is clear as such system must be very accurate: it is compulsory to avoid that a kid approaching the bus, or even worse, stepping on the entrance stairs, would be closed off by the doors as result of a few cm. localization errors. The PoC is as described in section 7.6 of [4]. As an initial integration step of the BUTLER project, there are few modifications in the interface and the architecture as shown in Figure 111 to realize the proof of concept.
The User Tags, which are the ZIGPOS modules as described in [4], communicates with the co-ordinator. The coordinator computes the ranging data and sends it to BUTLER SmartObject Gateway, which in turn sends the ranging data to the SmartMobile or SmartServer using the defined interface in Butler like Restful over http. The SmartMobile or the SmartServer can then send the notification to the vehicle driver when the entire children group is inside the bus or can directly send the command to activate the door opening/closing process.

5.6. SMARTSHOPPING SCENARIO

The city of Santander is a perfect choice for developing a SmartShopping solution due to its high concern in integrating new ICT technologies in everyday human interactions. Thanks to the recent SmartSantander FP7 project, Santander has recently enriched the city with a brand new deployment of a high-scale wireless sensor network, providing one of the biggest live city labs in Europe. In addition, a total number of 2000 stores and boutiques, 410 buses and bus stops and 200 taxis have been provided with QR and NFC tags reachable for common customers by means of using any regular smartphone. The project has also launched three multi-platform mobile apps, strengthening interactions between citizens and local authorities.

The SmartShopping Proof of Concept enables individualized advertising campaigns to customers interested in certain goods. The application running in the BUTLER SmartMobile platform receives news and offers from the different fields a user can be interested in. The key of the system is the usage of context information, location, user behavior, interests, already used offers and the like, to improve their profiling activity. With a very clear view of each user, both offers and advertisements are personalized and much more effective than using general distribution channels. The figure below shows the friendly and simple user interface for the SmartMobile application that is being developed.
Existing 2.0 shopping systems allow a limited interaction with potential users, there is a one to one relationship based only on what users say, but there is no exploitation at all of how they behave, their tendencies, the data associated to mass customs and information related to the environment surrounding the commerce. The scope of BUTLER SmartShopping Proof of Concept is to exploit all these sources of information combining them with user proactive information and thus creating a system that maximizes the benefits of commerce but mainly improve users experience creating a more dynamic, adaptable and competitive system where users are the key stone.

This field trial exploits BUTLER architecture by using some of the services offered by BUTLER SmartServers and also by running an application in the SmartMobile following the project’s guidelines.

The core of the system is based on the management of context information; in this case the elements that have been selected are users’ location, city’s events schedule, monitoring of commerce activities (banking transactions) and finally user interaction with the SmartMobile application. Gathering all the raw information using the already defined BUTLER interfaces and architecture, and applying Big Data and Business Intelligence techniques it will be possible to improve user knowledge and increase their satisfaction about the offers they get.
Figure 113: Relationship among entities in the SmartShopping field trial
6. CONCLUSION

The BUTLER Architecture provides a pragmatic approach to the IoT scenarios defined in BUTLER and to the derived requirements. When possible, other solutions and standards have been used in an effort to provide a really useful proposal that could be implemented by other parties.

Actual or perceived limitations of existing solutions have been also tacked: for instance, as IoT-A does not manage explicitly the concept of context, a context abstraction has been introduced in the Information Model. As no specific discovery infrastructure for the entities is available, an explicit entity has been introduced in BUTLER to play such a role. As the other considered research effort, FI-WARE, focuses too much on the delivery of context information and not in its generation, a generic mechanism for context declaration and generation has been proposed. BUTLER also introduces a SOA approach for managing Resources and Services provided by SmartObjects. Security Services built around an Authorization (and User Authentication) Server have also introduced, centralizing and securing the access to Resources.

There are still challenges for BUTLER that may impact on its architecture. To name a few, for instance, the exposition of data is not currently supporting semantic annotation in order to enhance the way it is used by other “machines”. Although data being mediated by BUTLER can be cached for further access, no analytics tasks are being executed on the stored data (neither the mediated data is being analyzed on the fly). The experience from FI-WARE can be helpful and provide new insights over the BUTLER architecture.
7. REFERENCES


[27] OpenSocial Specifications. Web page at https://opensocial.atlassian.net/wiki/display/OSD/Specs


[34] Demo version of the BUTLER SmartHealth Proof-of-Concept (Diabetes Application), University of Leuven – KU Leuven, https://butler.cs.kuleuven.be/samurai/


[40] ZigBee Document 095264r18, Version 1.0, ZigBee Over-the-Air Upgrading Cluster, ZigBee Alliance, 2010