H2020-ICT-2014-1-644332

OpenAIS

Open Architectures for Intelligent Solid State Lighting Systems

Initial architecture of OpenAIS system

Version 2.0, 18 July 2016

This work is partially supported by the Horizon 2020 program of the European Union under grant agreement number 644332

Executive Summary

The OpenAIS reference architecture defines a framework that allows system designers to create vendor specific, but still compatible Lighting Control systems. That is: OpenAIS devices and software services can rely on a common base that allows integration of both software and hardware from different vendors into a single system. This summary provides a quick overview of the basic principles and choices:

- OpenAIS uses IPv6 based communication and UDP as transport layer. IPv6 multicast will be used and 6LoWPAN compression will be supported.
- OpenAIS can be implemented with any (future) physical medium that is IPv6 compatible. Typical larger systems will have a LAN (or Wi-Fi) backbone that uses border routers to integrate small and large cells of local IoT communication networks (BT(LE), 6LoWPAN, PLC, VLC, PoE, etc.).
- All communication uses the well-standardized CoAP protocol, used in the domain of constrained embedded devices.
- Any OpenAIS device may use (any kind of) non-OpenAIS-standardized CoAP communication to extend the functionality of the devices.
- Security and privacy of data communication are achieved by using a combination of end-to-end transport (DTLS) and application (COSE) layer based encryption.
- Well-defined access levels and a role-based access structure ensure the integrity of the application and the privacy of the keys used.
- OpenAIS uses stable communication settings that are established and secured through commissioning tools.
- Legacy systems are incorporated using (application layer) gateways that talk “OpenAIS” on the IPv6 interface. The level of integration is up to the designers of the gateways and may well reach “full”.
- OpenAIS data structure on top of CoAP closely follows LWM2M (with some modifications and extensions) to ease and speed up the development. The choice of LWM2M was driven by the availability of the LWM2M specification and stack.
- OpenAIS extends the CoAP usage beyond a strictly RESTful approach to best cover the event-based character of manual interaction with the system.
- The protocols and stacks used are open (IoT) industry standards. Implementation of these will need some additional work to cover multicast and event orientation.
- OpenAIS provides out-of-the-box functionality, which delivers (insecure) basic operation to ease installation and installation-testing for luminaires, switches, presence detectors and light sensors.
- Mobile devices like tablets and phones can provide user control after being authenticated, and will talk directly to the control algorithms rather than to the devices, thereby providing granular specified access and group control.
- Cloud services will benefit from a data collector service that collects, processes and temporarily stores data locally and allows packing, extended authentication and encryption before sending the data to the cloud.
- The operational concept is strictly based on a two-step process workflow: (physical effect) - Sensor → Control (algorithmic) function → Actuator - (physical effect). Each of these steps may include multicast communication: Sensors may talk to more than one control function, and each control function may talk to multiple actuators.
- Control functions are kind of “local web based” and are not bound to specific hardware. Simple ones may be integrated into sensor or luminaire hardware, more complex ones may supersede the simple ones using the OpenAIS control object stacking mechanisms. This allows vendors to sell specific controls also in the later phases of the system life cycle.
Table of Contents

Executive Summary ........................................................................................................... 2

1 INTRODUCTION .......................................................................................................... 6
  1.1 Goal .................................................................................................................... 6
  1.2 Scope .................................................................................................................. 6
  1.3 Glossary .............................................................................................................. 8
  1.4 Organisation of the report .................................................................................. 8

2 ARCHITECTURE SELECTION .................................................................................. 9
  2.1 Key Drivers .......................................................................................................... 9
  2.2 Main Requirements ............................................................................................ 9
    2.2.1 User Requirements ....................................................................................... 9
    2.2.2 Technically translated user requirements ..................................................... 11
  2.3 Architecture Candidates .................................................................................... 11
    2.3.1 Process ......................................................................................................... 11
    2.3.2 Architecture Candidates .............................................................................. 12
  2.4 Architecture Decision Matrix ........................................................................... 15
    2.4.1 Decision Making Process .............................................................................. 15
    2.4.2 Decision Matrix ............................................................................................ 15
    2.4.3 Comparison of Candidate Architectures ......................................................... 17
  2.5 Final Candidate .................................................................................................. 19
    2.5.1 Comparison of final candidate architecture and baseline architecture ........ 19

3 REFERENCE ARCHITECTURE .................................................................................. 25
  3.1 Introduction .......................................................................................................... 25
    3.1.1 System Context .............................................................................................. 25
    3.1.2 Basic Concepts .............................................................................................. 26
    3.1.3 Architecture Views ......................................................................................... 26
  3.2 Derived technical requirements .......................................................................... 27
    3.2.1 Introduction ................................................................................................... 27
    3.2.2 Error and Recovery handling requirements .................................................... 27
    3.2.3 Time Synchronized Operations Requirements ............................................ 28
    3.2.4 Security requirements .................................................................................... 28
    3.2.5 Out-of-the-box functionality ......................................................................... 30
  3.3 Logical View ........................................................................................................ 32
    3.3.1 Introduction ................................................................................................... 32
    3.3.2 Decomposition ............................................................................................... 32
    3.3.3 Application Layer ........................................................................................ 36
    3.3.4 Infrastructural Layer ................................................................................... 42
    3.3.5 Interface overview ......................................................................................... 44
    3.3.6 Dynamic behaviour and example message flows in OpenAIS ............ 47
3.3.7 Object Data Model ................................................................. 51
3.3.8 Extension Mechanisms .......................................................... 55
3.3.9 General Mechanisms ............................................................. 58
3.3.10 Out-of-the-box design ......................................................... 61

3.4 Physical View ........................................................................ 62
3.4.1 Introduction ......................................................................... 62
3.4.2 Physical devices ................................................................. 63
3.4.3 Networking choices ............................................................. 63
3.4.4 Physical structure of a system .............................................. 64
3.4.5 Device Classes ................................................................. 64
3.4.6 Device Management ............................................................ 65

3.5 Networking View ................................................................... 68
3.5.1 Overview ........................................................................... 68
3.5.2 Open Mobile Alliance Lightweight M2M Standard (LWM2M) ... 69
3.5.3 Constrained Application Protocol (CoAP) ....................... 70
3.5.4 Low Power Radio Access Point ........................................ 71
3.5.5 Network Joining ............................................................... 73
3.5.6 Group Communication ..................................................... 74
3.5.7 Binding and Eventing (Subscriptions and Notifications) .... 78
3.5.8 Discovery .......................................................................... 81
3.5.9 Network Management ....................................................... 82
3.5.10 Low-power "Sleepy" Devices ............................................. 82
3.5.11 Legacy IPv4 Networks Interfacing .................................... 83
3.5.12 PHY/MAC Requirements ................................................ 84

3.6 Security View ....................................................................... 84
3.6.1 Introduction ....................................................................... 84
3.6.2 Overview .......................................................................... 86
3.6.3 Secure Bootstrap .............................................................. 88
3.6.4 Authorization, Authentication, and Key Handling ............. 91
3.6.5 Secure Communication ..................................................... 92

4 ARCHITECTURE REALIZATION .............................................. 94
4.1 Component Requirements and recommendations ............... 94
4.1.1 Introduction ....................................................................... 94
4.1.2 Logical view ..................................................................... 94
4.1.3 Physical view .................................................................... 94
4.1.4 Network view .................................................................... 96

4.2 System Design Considerations ............................................ 99
4.2.1 Control Object General Concepts ..................................... 99
4.2.2 Control Object Southbound Interface, handling Actuators ... 101
4.2.3 Control Object Westbound interface handling: sensor signals .... 102
4.2.4 Control Object Eastbound interface: User control .......... 103
4.2.5 Control Object Northbound Interface: Stacked control ....... 103
1 INTRODUCTION

1.1 Goal

OpenAIS project envisions to “Create an open ecosystem to enable a wider community to deliver the smartness of light. Allow easy adaptability to cater for the diversity of people and demands.” [OpenAIS_D1.1]. By exploiting the smartness of lighting, the OpenAIS system aims at enabling efficient use of buildings of 2020 and beyond with increased comfort and well-being of the users and with significantly reduced operational costs. The key to achieve this goal is to establish the “Internet of Lights” by converging the technologies of Internet of Things (IoT) and Solid State Lighting.

This deliverable provides the reference architecture of OpenAIS - an open service oriented IP-based lighting architecture. Connecting luminaires to the Internet with IP (Internet Protocol) to all nodes enables flexibility and interoperability and it facilitates a service oriented architecture. A transition from the existing closed and proprietary systems to open systems will stimulate investments and innovations. OpenAIS aims not just to match the robustness, reliability and real-time performance of today’s dedicated lighting systems in an Internet connected luminaires world, but to expand those into the functionality of the future world of IoT systems.

The architecture provided by OpenAIS is a reference architecture, i.e. a template for specifying concrete system architectures. This architecture is designed to support a wide range of deployment scenarios and use cases and to fit the requirements of future office buildings. Emphasis is to provide an architecture that is extensible to future technologies yet secure. Interoperability with Building Automation Systems (BAS) and other building systems, support for cloud storage and big data analytics and integration of legacy technologies using gateways are among its core features. Moreover, by enabling a multi-vendor system with vendor differentiation and competition without dropping compatibility, and by its flexible approach to support future connectivity, OpenAIS is set to be future proof.

1.2 Scope

The primary scope of this document is to explain the OpenAIS reference architecture. It does not just define a system, but defines a template for designing a family of systems. The system can be vendor specific or fully multi-vendor-based where customers can choose components from different vendors. The vendor differentiation thus becomes the key business control point. Moreover, this reference architecture allows reuse of standard IT systems and provides extensibility and openness in adopting future developments and technologies, and hence can serve over a longer period of time, typically 20–30 years, as expected from lighting systems.

What will be covered in this document are the following:

- The logical model of interaction and signal flow.
- The communication requirements (transport, protocols and security).
- The commissioning basics (out-of-the-box operation, configuration workflows and the related requirements).
- The extensibility basics (how systems stay flexible and how changes and extensions can be performed throughout the life cycle).
- The interaction basics to Building Automation Systems and other smart building services.
These are accompanied by some high level examples to make the intentions more graspable. However, out of scope for this document are:

- The Interface and Object Model basics (with detailed API and object structures) will be provided in a different document [OpenAIS_D2.4].
- The defence of the choices taken (some arguments are provided to clarify certain issues, but no defence is given).
- The system design itself (this is up to the vendors).
- The limitations for actual installations (these will be system limitations).
- Other Building Automation Systems and services.
- Lighting controls algorithms and user interfaces.

The main focus of OpenAIS is on lighting of indoor offices, and other (non-lighting) aspects of smart buildings such as Building Automation System, Building Information Modelling and Network management system are not part of the architecture. However, the means for enabling these services and data sharing are defined.

Some of the OpenAIS principles may also serve non-lighting systems well (especially blind and shade services that are structurally close to lighting), e.g. by easily adapting the object structure, but this was considered out of scope. Integration of the existing legacy systems is supported through application layer gateways.

The architecture allows flexible control functionality by supporting both distributed and centralized lighting control algorithms. The inner functional details and parameters of the algorithms are not part of the reference architecture, as they are part of a system design.

The details of all objects, interfaces, tools (e.g. commissioning) and services are not covered, these are the points for vendor differentiation and there will be vendor specific capabilities in these devices and tools.

Further not covered are off-site aspects, lighting device design, CPU, OS, sensor technologies, and firmware developments, although we provide some default/recommended choices.

The architecture supports both wired and wireless solutions, as the physical transportation layer used is not fixed in the architecture. Instead the minimum requirements for connectivity are specified. We expect that specific PHY transportation is always coupled to a common LAN backbone with a specific border router.

Although part of the architecture, this document does not include the detailed interface definitions that are mandatory for OpenAIS devices. This is due to the fact, that the object model itself is part of ongoing work, and the results will be provided in deliverable [OpenAIS_D2.4]. Wherever necessary we provide reduced and simplified versions of interfacing examples.

Disclaimer: This document provides the initial version of the “Reference Architecture of OpenAIS”. It puts together all the architectural decisions taken, and provides the reference architecture to create OpenAIS compatible systems. However, the upcoming implementation and testing effort that check the most critical parts of the architecture against reality may of course lead to some need for change, and some revisions may be necessary in the future to achieve reasonable results. A second issue to watch is the ongoing development of new stacks and frameworks in the IoT world. A framework that fits the requirements better may become available in the near future, and if that
occurs some of the more detailed and implementation focused example sections may need revision. All such revisions will be reported in the final version of the Reference Architecture which will be available in early 2017.

1.3 Glossary
A glossary of acronyms and the definition of the key terms used in this document are given in Chapter 9. The icons used to represent OpenAIS system components are listed in Section 9.3.

1.4 Organisation of the report
The rest of the report is organised as follows:

Chapter 2 summarizes the system requirements that are used as the base of the OpenAIS Architecture and the process we followed to arrive at the final candidate architecture. The key features of the candidates, their strength and limitations, and features of the tool used for comparison are described in this chapter. Chapter 3 presents an extensive overview of the proposed reference architecture. It details the main architectural concepts, decomposition of the system and four different architecture views. Chapter 4 elaborates on system design requirements and recommendations based on the reference architecture, and adds some examples and illustrations to help comprehending specific architectural issues and to give a more practical insight on the way this reference architecture works. Chapter 5 focuses on architectural analysis, especially on the risk assessment aspect. It describes the methods used for the assessment and the outcome of the two workshops conducted. Finally, Chapter 6 concludes the report.
2 ARCHITECTURE SELECTION

This chapter summarizes the system and user requirements that are used as the base of the OpenAIS Architecture and the process we followed to arrive at the final architecture candidate. The key features of the candidates, their strengths and limitations, and features of the tool used for comparison are described in this chapter. Additionally, a comparison of the shortlist candidate architectures with a baseline architecture is provided.

2.1 Key Drivers

The OpenAIS Architecture follows a simple goal: To provide an excellent and secure lighting controls framework that fits into and that uses open methods from the IoT world. The key drivers/requirements of OpenAIS are:

• Use the Internet Protocol transport to the final node.
• Avoid translating gateways for future applications.
• Allow third party contribution throughout all aspects of the system. (Device interoperability, standardized open APIs, Software plugins)
• Open up interoperability with other Building Automation Systems (HVAC, Blinds etc.)
• Make the commissioning process as simple and straight forward as possible, use as little (technical and educational) preconditions as possible.
• Allow for wired and wireless and seamlessly mixed installations.
• Stay close to existing IT / IoT frameworks, to allow the use of already existing tools.
• Open up lighting controls for cloud based services, but maintain fast local reaction to manual interaction.
• Support centralized, decentralized and distributed control structures.
• Allow lighting nodes to act as infrastructure for power restricted IoT devices close to the lighting nodes.
• Allow lighting equipment to be part of an overall data management framework.

2.2 Main Requirements

2.2.1 User Requirements

The user requirements that are used as the base of the OpenAIS Architecture have been collected by WP1 and are to be found in a comprehensive version including the reasoning in the WP1 deliverables. This section sums up the major architectural requirements that influence the system architecture to ease the reading of this document. Device and/or system requirements that are not connected to the architecture are not used here. The different wording compared to the WP1 results has been chosen on purpose to support the architectural focus best, the content remains.

2.2.1.1 The system should perform well

• Performance should be independent of connectivity choices: 6LoWPAN, Wi-Fi, LAN, PoE, PLC (DLAN), BT (LE), etc., and should be performing well also with future connectivity systems.
• The system should perform well in mixed installations, where different networks are connected to act as a single system.
  o Switching or dimming lights in a room/floor/facade by users should see reliable, well synchronized visible action in reasonably short time. Both the
waiting time before visible action and the "run_on_time" after stop of
dimming need to be short and consistent.

- The size of the system should scale nicely from single room to campus type
  installations.
- The operational security, the data privacy and the system integrity should be
  provided as an internal feature of the system, and work independently from site
  protecting firewalls.
- The system should support group-/room-/floor-/facade-/stairwell-/building- wide
  control interaction. Overlapping and conflicting groupings need to be managed and
  covered.

2.2.1.2 The system should keep initial cost low

- There should be no expensive minimal requirements to devices or networks.
- Available, and becoming available standard technologies and software stacks
  should be sufficient for the realization of the system with only minimal additions.
- It should allow for (substantial) aftermarket enhancements and services business
  without the need of device replacement.
- It should allow for control algorithms placed in the cloud as a cheap alternative to
  on-site servers where needed.
- The upgrade of the system to higher performance should be possible by third
  parties that did not perform the first installation or commissioning.

2.2.1.3 The system should be easy to install, commission and operate

- Every major step of the installation and commissioning process should allow for a
  simple (self-) check if the performed step worked well and guide easily to the
  remaining activities.
- Organizational (pre-programming, pre-labelling) and technical (indoor positioning,
  guided search) localization (=identification of the placement of the device) should
  be well supported and future possibilities should not be excluded by the system
  architecture.
- The system architecture workflows should support integration into BIM
- The system APIs that guide the installation and commissioning process should be
  standardized to allow competition on the easiness of the process.
- Users should be easily able to use their mobile devices such as smartphones and
  tablets as extended lighting control points.
- In case of device or commissioning failures, the system should be able to continue
  operation with only limited operational damage.
- User preferences should be allowed to move along with the user.

2.2.1.4 The system should add value to the building

- The lighting system should provide infrastructure to upcoming IoT communication
  needs.
- Emergency lighting and other closely related lighting technologies should integrate
  easily into the lighting controls system, reducing the number of separate systems.
- The lighting controls system should seamlessly integrate into the overall Building
  Automation System, easing maintenance and optimization processes.
- The lighting controls system should provide full cloud service access to support
  “smart building” and “smart city” technologies of the future.
- The system should ensure compatibility between different vendors covering
  devices, software and tools.
2.2.2 Technically translated user requirements

Some of the user requirements need a more technical wording to be useful as a base for architectural decisions. E.g. “should react fast” as a user wording needs a translation into the more precise “within 0.4 sec” to serve as a requirement. This section provides this more technical form of requirements:

- The total delay from the sensor action “presence detected” to “lights on” needs to stay substantially below 0.4 seconds.
- The total delay from sensor action (button pressed / released) to (advanced) reaction of light points needs to stay below 0.4 seconds.
- The difference between the first and the last light point to stop dimming when the dimming action is stopped (button released) needs to stay below 0.1 seconds, the resulting difference in intensity change needs to be less than 5%.
- The residual timing difference at the actuators for actions that are requested to start or finish at a specific time (e.g. automated scenes) needs to stay below 0.1 second.
- After a power return the system should be operational within 5 seconds.
- Switching lights needs to be independent from the availability of the internet connection of the site.
- Sensor timestamps need to be accurate to the second.
- System operation needs to stay well defined and operational at all times, also when single devices fail.
- Communication needs to be well encrypted to avoid readability or unwanted message injection by third parties. The level needs to be AES128 or higher.
- All participants (devices, objects, superior control or temporary users) need to be well authenticated before any action can be performed.
- The presence status of rooms shall not be determinable by traffic pattern analysis.
- Additional sensor or status analysis through local tools or cloud connections needs to be available as addition without touching the lighting controls performance.
- Allow for cost effective device design using restricted computing at sensors and actuators, and provide integration means for sleepy battery or environmental powered sensors.

2.3 Architecture Candidates

Before detailing the selected OpenAIS architecture, a shortlist of three architecture candidates was first created. To arrive at three, first a larger set of candidates was collected by looking from multiple viewpoints to potential and desirable system architectures – based on the outcomes of the OpenAIS state of the art analysis done in the first half of 2015. This section outlines the process followed and the three candidates that resulted. More details can be found in OpenAIS deliverable D2.1 [OpenAIS_D2.1].

2.3.1 Process

The shortlist architecture candidates were generated by looking from three different high-level viewpoints:

- Architectures based on new IoT Technology – the re-use of existing IoT technology and application-level frameworks can be a good starting point of defining OpenAIS architecture in order to bring lighting systems directly into the IoT domain. Any aspects that are missing from existing IoT technology will need to be added by OpenAIS.
• Architectures based on Existing Lighting/Building Ecosystems – in other words, the “Heritage” Architectures. Because lighting and building control systems are today dominated by a number of strong ecosystems (like BACnet, KNX, DALI) it would be beneficial for market adoption to extend these existing ecosystems into full IP-to-the-luminaire solutions, using the existing standards as much as possible.

• Architectures based on Strategic Business Goals – these are based on strategic business directions for the lighting and building control industries. This means that the starting point of designing a new architecture is to select a single business goal and optimize the architecture for this single business goal. The goals selected were:
  o Low cost
  o Data-centric, or ready for the big data era
  o Existing IT systems/technology compatible

2.3.2 Architecture Candidates

This section lists the final three architecture candidates that were selected by the OpenAIS WP2 for further specification and comparison. The three candidates are:
  1. IoT-centric Architecture
  2. IT-compatible Architecture
  3. BACnet/IP Architecture

The comparison between the candidates is discussed in Section 2.4.3.

2.3.2.1 IoT-centric architecture

The IoT-centric architecture is based on today’s IoT standards, using a RESTful communication model which is also the basis for the World Wide Web. The application-layer frameworks of OMA Lightweight M2M (LWM2M) and IPSO Smart Object Profiles are used. Wireless connectivity to luminaires will typically use a low-cost IoT-specific technology with 6LoWPAN, while wired connectivity uses Ethernet typically. Any future communication technology that allows for IPv6 connections may be used with IoT centric architecture once it is standardized.

The expectation for this architecture is that lighting will be just one of the many devices connected to the Internet in offices of the future and therefore will have to adhere to standards and not follow a separate, lighting-specific path of evolution. The IoT developments are volatile at the moment and it will take some time before overall winners and leading standards emerge. Therefore this architecture must have the flexibility to incorporate technology changes over time.

The IoT-centric architecture also contains elements of the identified “low-cost” architecture, since low-cost microcontrollers can be used and low-cost, constrained IP networking technology is enabled.

Main characteristics of the architecture are:
  • A platform solution in which not all technology is completely fixed, allowing evolution over time.
  • Protocols used are mostly IP protocols optimized for use in low-resource devices (such as CoAP, DTLS, 6LoWPAN, MPL, CBOR, etc.).
  • Fully IPv6 based solution (IPv6 to the end node). 6LoWPAN is used where needed to support IPv6 over low-bandwidth network links.
• Support of both wired and wireless solutions, with definition of minimum requirements for connectivity technology.
• Integration of legacy (e.g. DALI and other) installations through gateways.
• Allowing arbitrary allocation of control functions, supporting both distributed and centralized control models, as well as mixtures.
• Three classes of embedded devices are distinguished: low-resource, medium-resource and full-resource. Only the full-resource devices are relatively powerful embedded Linux-class platforms.
• Support for stand-alone networks (i.e. isolated; not connected to an IP backbone network).
• Embedded security (node access and communication) is well supported.

2.3.2.2 IT-compatible architecture
The IT-compatible architecture is fully based on today’s IT equipment and best practices. A RESTful communication model is used through the HTTP/TCP protocols. Wireless connectivity uses Wi-Fi, while wired connectivity is over Ethernet LAN. Wi-Fi access points are abundantly placed to enable many luminaires and sensors to connect to the IT network.

The expectation is that Lighting will be “just” one of the many devices connected to the IT network in offices of the future and therefore will have to adhere to IT standards and not follow a separate lighting-specific path. This means that computing power in lighting devices will have to grow significantly to the level needed to support standard IT interconnectivity and protocols.

Administration and some installation/configuration actions are performed by the IT department of a building or a campus. Where such department or expertise is not available, setup and maintenance software on a powerful gateway/controller type device (which may be a temporary placed device) should provide a seamless out-of-the-box installation experience.

Main characteristics of the architecture are:
• A platform solution in which not all technology is completely fixed, allowing evolution over time.
• Protocols used are the same as in today’s IT networks, i.e. not optimized for low-cost/low-bandwidth networks; such as HTTP, HTTPS, TCP, TLS, DNS, DNS-SD, SNMP, NetConf, MLD, JSON/XML, etc.
• Fully IPv6 based high-bandwidth solution (IPv6 to the end node). 6LoWPAN compression is not needed.
• Support of both wired and wireless solutions, with definition of minimum requirements for a connectivity technology.
• Optimized for data collection and the expected large increase of data collection in the future.
• Preferred connectivity technologies are Ethernet LAN/PoE and Wi-Fi. However arbitrary connectivity standards can be used, as long as they comply with the minimum requirements: in the future new standards like sub-GHz 802.11ah may be feasible in some regions of the world.
• Integration of old (e.g. DALI and other) installations through gateways.
• Allowing arbitrary allocation of control functions, supporting both distributed and centralized control models, as well as mixtures.
Specific IT configuration knowledge encoded in commissioning tools, to enable smaller organisations with less IT expertise to install or maintain a lighting system.

Two classes of IP-enabled embedded devices are distinguished: standard-resource and full-resource. Both are relatively powerful embedded-Linux class devices.

2.3.2.3 BACnet/IP architecture

The BACnet/IP architecture is based on the BACnet protocol as it is defined today to run over UDP/IP. It is widely used today for building controls, and when it comes to heating, cooling and ventilation more than 60% of the market today are operated using the BACnet model.

The BACnet system is supported by many vendors in the building management industry, who also offer the administration and commissioning services for the devices and the full network solution by their specialized service-force. There is a variety of independent and also vendor specific tools available that help to perform BACnet administration.

Although there are BACnet to DALI Gateways available from some vendors, BACnet is not widely used for lighting controls as of today, and especially not when it comes to more sophisticated dimming controls, as the BACnet structures can get complicated if more sophisticated or more granular dimming control is required. The necessary combined knowledge on both “advanced lighting” and “advanced BACnet” is usually not available on-site. We could not find any evidence of luminaires with integrated BACNet Lighting nodes, although there is no real technology barrier besides the relatively high cost per light point.

BACnet is today a very restrictive standard. To ensure compatibility and reliability the standard has no flexibility embedded when it comes to additional information or variance in additional object methods. BACnet is designed to be used in closed network environments. There are no reasonable authentication or authorization methods implemented, even though there is a standard on paper for BACnet security.

Wireless connectivity for BACnet/IP will be enabled over Wi-Fi or another high/medium speed technology. Wired connectivity uses Ethernet. Luminaires will incorporate those parts of the BACnet protocols needed to operate all lighting and sensor functions. Extension of the BACnet models will be needed to allow for an easy and seamless installation and commissioning experience using automated tools.

Main characteristics of BACNet are:

- Widely used for large buildings
- Lighting is integrated via gateways
- A restrictive object and interaction model
- A full, but closed (heritage) ecosystem
- No object security
- No authorization and access management
- No RESTful communication architecture
- IPv4 only, special port and protocol used.
- Light point group action is not bandwidth optimized
- No cloud interfaces available
- Not applicable to small lighting systems
2.4 Architecture Decision Matrix

Architecture Decision Matrix is a tool we created to guide us to an appropriate final architecture proposal while making complex architectural decisions. The details of the tool are given in OpenAIS deliverable D2.2 [OpenAIS_D2.2]. This tool is based on the user requirements, and helps us to judge the performance of a system against various aspects of those requirements. To achieve this, the (relatively abstract) user requirements are broken down to technical criteria and these criteria are split into measurable sub-criteria, which finally allowed scoring on all the dimensions of these requirements in a reasonable way. These results have been made available graphically using a spider diagram, which allows judging all dimensions at a glance. This section outlines this process and gives the reasoning for the decisions taken based on the process.

2.4.1 Decision Making Process

The decision making process we adopted starts with identifying every critical aspect of the system. An aspect of a system does not directly imply a decision criterion since it needs further processing. A more complex task was to get the criteria measurable. For creating a comparable decision criterion from a system aspect, the criterion must be well defined in the context of the system and it must be broken down into sub-components. This process requires domain specific expert opinions. After this step, the sub-components can be accepted as sub-criteria of an actual criterion. Subsequently, sub-criteria of a criterion might have different priority levels. An expert opinion is necessary to decide whether there is a priority difference between them or not. If there is a priority difference, the expert must give different weights to each sub-criterion.

For example, in a decision making process a typical user requirement (criterion) like “ease of handling for the electrical contractor” typically creates a “guessing” rather than a measurement. But when it is split into three sub-criteria, namely “ambiguity of connections to be made / antenna positioning”, “availability of connecting materials”, and “ease and clarity of connection testing” and weighed against each other, the three sub-criteria can now be measured using a clear metric which makes the criterion more measurable. Of course this process is not “complete” and it also relies on “expert opinions”, but tests have shown that the results are useful and reproducible.

Each sub-criterion is rated using its own metric. The weighted sum of scores of sub-criteria reflects the score of actual criteria. These scores are then being used for comparing architectures. A “final number” based on the weighted sum of scores of criteria to compare different architectures would make the process easier, but we decided not to implement this, as there is no weighting for criteria that tells us how the diverse aspects (e.g. from “ease of installation” to “power efficiency”) should be related. The aim was therefore to judge if the architectures have no essential weaknesses, and also if they could possibly be enhanced through OpenAIS. This can be done by watching the magnitude for the various aspects on a spider diagram much better than by looking at a “total” value.

2.4.2 Decision Matrix

Deriving the decision making criteria is an important step. Extracting the decision making criteria from the user requirements requires a delicate work and a strong communication between different teams. In OpenAIS, WP1 and WP2 worked together to ensure that every requirement and use case have an equivalent decision making criterion.
This section shows the design assessment criteria in their broken down format and the measurable sub-criteria that have been used to score the “fitness” of the architecture candidates. Following are the decision making criteria and corresponding sub-criteria:

- **Ease of handling (maintenance, installation) for Electricians**
  - Unambiguous connectors (no choices) / antenna position (no choices)
  - Easily available cabling and connectors material
  - Simple “connection works” check (yes/no, nothing in between)

- **Ease of handling (maintenance, installation) for IT**
  - Ease of configuration (during the installation)
  - Ease of maintenance
  - Firmware and configuration updates management

- **Ease of handling (maintenance, installation) for providers (lighting)**
  - Ease of localization of devices
  - Ease of grouping / addressing devices
  - Ease of trouble shooting (triage, error diagnostics)

- **Interoperability with Building Automation Systems**
  - Sensor, output and data transparency
  - Ease of configuration
  - Energy Management

- **Use of open standards**
  - Use of open standards to realize IP connectivity (incl. L1,L2,L3) in field network
  - Use of open standard application-level IoT framework (L4-L7) in luminaires
  - Use of open data models for luminaires/sensors/controls

- **Reuse from IT domain**
  - Uses service integration technologies familiar to IT group staffs (to avoid special training needs to commission or maintain a lighting system)
  - Uses network infrastructure already provided by IT groups
  - Uses existing support, service monitoring and asset management processes of IT groups

- **Upgradability to this architecture from existing architectures**
  - Functional groups span new and old system seamlessly
  - Integrated commissioning methods and tools
  - Integration efforts

- **Compliance with functional requirements**
  - Availability of required functionality (according to Kano model)
  - Tools to enable required functionality
  - Interfaces / access points to the system to enable required functionality

- **Security**
  - The system retains the interconnection and interoperation benefits of IP and is easy to access for legitimate installers, developers and end users
  - The security concept embodies "defence in depth"
  - The security model provides for typical commercial building sector roles and responsibilities, and changes of role holders

- **Power efficiency**
  - Standby power consumption
  - Operational power distribution efficiency
  - Efficiency of controls for effective energy reduction

- **Business control points (vendor differentiation)**
• A vendor can deliver and use device functionality up and above the standard features
• A vendor can deploy additional system functionality to all networked devices
• A vendor can detect and use additional functionality supplied by a foreign device

- Manufacturing, Installation and Commissioning cost (per luminaire)
  - Device Cost of “Connected” Converter (€) [measured @ LED 20W @ purchase level by the luminaire company]
  - Installation & connection cost (h) [measured @ 4m connection to next device in false ceiling]
  - Commissioning and fault finding cost (h)

- Extensibility
  - Integration of new connectivity technology
  - Integration of new sensor (ubiquitous video) or actuator (blinds)
  - Change of IoT standards (e.g. CoAP replaced or new discovery mechanism)

- R&D effort
  - R&D cost for first system development
  - R&D cost for component development in new architecture
  - Transition cost of the organization (training, hiring)

- Scalability
  - Total costs for adding devices and / or functionality
  - Continuous functionality / UI independent of size
  - Performance independent of size

- Reliability (Error resilience)
  - Availability
  - Recoverability
  - Degradability

- Performance (response time, synchronicity)
  - Time to Light (TTL)
  - Synchronicity when switching / dimming a group of devices
  - Start-up time until the system is in regular state (up and running) again after power loss

- Emergency lighting integration capability
  - Integration level
  - Monitoring level
  - Functionality

2.4.3 Comparison of Candidate Architectures

The three architecture candidates presented in Section 2.3.2 have been scored using this catalogue of aspects and criteria, and compared using the spider diagrams as shown in Error! Reference source not found.. The main focus was to understand which architectures have essential difficulties in supporting certain aspects of the requirements, and if their frameworks allow for adaptations that result in a reasonable coverage of the more critical parts. Both the IT-compatible and the BACnet/IP have been found to be less promising than the IoT-centric architecture. Hence the IoT-centric architecture has been selected as the final candidate architecture; but this architecture also needs some essential extensions and adaptations to make it fit. The IoT-centric architecture given in Section 3 provides a good and reasonable coverage of the user requirements. Some additional details are given in Section 2.5.
2.4.3.1 IT-compatible architecture

- IT- Architecture is optimized for high throughput and good bandwidth using larger data packets. It is not optimized for low-cost/low-bandwidth/fast access for small packets. Protocols used are the same as in today’s IT networks.
- Architecture is based mainly on IPv4 and TCP connection. (IPv6 is coming up, and UDP is possible, but less supported in classic IT environment.)
- With IPv4 there are only limited / no ways for multicast available.
- Classic IT commissioning workflows and methods are less advisable for lighting commissioning.
- Architecture forces to use relatively powerful embedded-Linux class devices.

Therefore, this architecture got substantially low scores on the following criteria:

- Manufacturing, Installation and Commissioning cost (per luminaire)
- Power efficiency
- Ease of handling (maintenance, installation) for Electricians and lighting providers

2.4.3.2 BACnet/IP architecture

- BACnet/IP to the node is not used in lighting controls today. BACnet/IP is used to control DALI gateways, supporting relatively simple requirements.
- Architecture is weak on supporting sophisticated dimming control functions, especially with manual dimming of groups.
- Architecture is restrictive rather than flexible.
- Architecture lacks security aspects.
Therefore, this architecture got substantially low score on the following criteria, with little to no outlook that this can be changed through efforts in the OpenAIS team:

- Compliance with functional requirements
- Business control points (vendor differentiation)
- Extensibility
- Security

### 2.5 Final Candidate

The goal of OpenAIS is to achieve an architecture that can stand for more than 10 years, and for more than 20 years for compatible upgrades. Therefore the flexibility to adopt new technologies in an interoperable way and substantial freedom for system designers within the framework of architectural compatibility are the major future concerns. Interoperability with Building Automation Systems and other building systems, seamless integration of heritage systems, ease of handling, cost effectiveness, openness and vendor differentiation are the important aspects for the near future.

The IoT-centric Architecture emerged as the best candidate following the comparison by using decision matrix. Strong aspects of the IoT-centric architecture that made it the best candidate are:

- Ease of handling
- Interoperability with Building Automation Systems
- Manufacturing, Installation and Commissioning cost (per luminaire)
- Compliance with functional requirements
- Scalability
- Security
- Extensibility
- Openness

Of course there have been a few issues with the IoT framework, especially to ensure local functionality, resilience, performance across different network topologies, security for multicasting etc. All these could be addressed, but and in some design aspects we had to deviate from the usual IoT paths. Some even lead to standardization proposals through IETF. After covering all such issues no substantially weak aspects are left, still some concerns regarding development efforts and costs remain. With all these changes, the final candidate now covers the important aspects towards the goal of designing an architecture that can last for many years.

#### 2.5.1 Comparison of final candidate architecture and baseline architecture

The need to push some aspects of IoT further to achieve a reasonable fit to the requirements, and the residual work to pin down some more technical-practical aspects in the wide field of IoT, leaded us to continue using the decision matrix as a tool that supports the decision making process. Alternatives, and collateral changes and their influence could be checked using the decision matrix.

To understand how the final candidate performs against a most recent “heritage” system, the team chose the fully featured Zumtobel “LITECOM” system, that has been successfully introduced in 2014 [LITECOM]. The score of LITECOM is used as a comparison baseline to understand the performance abilities of the proposed architecture.
Please note that “comparing Architecture against a given system” is not well defined. The reference Architecture, as outlined here will allow for both simple and sophisticated implementations; the simple ones usually perform well on the cost side, and the sophisticated ones on the feature side. One of the benefits of the OpenAIS architecture is the capability to allow for both, and have them compatible in the same system. It also allows (with only little limitations) a (partial) upgrade to a more sophisticated version during operation, and (with only little limitations) with continued use of the infrastructure.

The following tables show the scores for the OpenAIS Architecture and Zumtobel LITECOM:
Table 1: Scores of sub-criteria of OpenAIS final candidate and Zumtobel LITECOM

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Sub-Criteria</th>
<th>Weight</th>
<th>OpenAIS Final Candidate</th>
<th>Zumtobel LITECOM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rating</td>
<td>Score</td>
</tr>
<tr>
<td>Ease of handling, maintenance, installation)</td>
<td>Unambiguous connectors (no choices) / antenna position (no choices)</td>
<td>50%</td>
<td>3</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Easily available cabling and connectors material</td>
<td>20%</td>
<td>4</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>Simple „connection works“ check (yes/no, nothing in between)</td>
<td>30%</td>
<td>4</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>11</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>Ease of configuration (during the installation)</td>
<td>45%</td>
<td>5</td>
<td>2.25</td>
</tr>
<tr>
<td></td>
<td>Ease of maintenance</td>
<td>40%</td>
<td>4</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>Firmware and configuration updates management</td>
<td>15%</td>
<td>4</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>13</td>
<td>4.45</td>
</tr>
<tr>
<td></td>
<td>Ease of localization devices</td>
<td>33%</td>
<td>4</td>
<td>1.32</td>
</tr>
<tr>
<td></td>
<td>Ease of grouping / addressing devices</td>
<td>33%</td>
<td>4</td>
<td>1.32</td>
</tr>
<tr>
<td></td>
<td>Ease of trouble shooting (error diagnostics)</td>
<td>34%</td>
<td>5</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>13</td>
<td>4.34</td>
</tr>
<tr>
<td></td>
<td>Sensor, output and data transparency</td>
<td>50%</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>Ease of configuration</td>
<td>25%</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Energy Management</td>
<td>25%</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>13</td>
<td>4.55</td>
</tr>
<tr>
<td></td>
<td>Use of open standards to realize IP connectivity (incl. IEC 61183) in field network</td>
<td>35%</td>
<td>5</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>Use of open standard application-level IoT framework (IETF 7) in luminaires</td>
<td>35%</td>
<td>4</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>Use of open data models for luminaires/sensors/switches</td>
<td>50%</td>
<td>4</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>13</td>
<td>4.35</td>
</tr>
<tr>
<td></td>
<td>Uses service integration technologies familiar to IT group staffs</td>
<td>30%</td>
<td>5</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Uses network infrastructure already provided by IT groups</td>
<td>40%</td>
<td>4</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>Uses existing support, service monitoring and asset management processes of IT groups</td>
<td>30%</td>
<td>3</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>12</td>
<td>3.75</td>
</tr>
<tr>
<td>Reuse from IT domain</td>
<td>Functional groups span new and old system seamlessly</td>
<td>50%</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>Integrated commissioning methods and tools</td>
<td>20%</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Integration efforts</td>
<td>30%</td>
<td>5</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Compliant with functional requirements</td>
<td>50%</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>Availability of required functionality (according to kano model)</td>
<td>50%</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>Tools to enable required functionality</td>
<td>20%</td>
<td>5</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>Interfaces / access points to the system to enable required functionality</td>
<td>30%</td>
<td>5</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>14</td>
<td>6.8</td>
</tr>
<tr>
<td>Security</td>
<td>The system retains the interconnection and interoperability benefits of IPv6 and is easy to access for legitimate installers, developers and end users</td>
<td>34%</td>
<td>5</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>The security concept embodies &quot;defense in depth&quot;</td>
<td>33%</td>
<td>4</td>
<td>1.32</td>
</tr>
<tr>
<td></td>
<td>The security model provides for typical commercial building sector roles and responsibilities, and changes of role holders</td>
<td>33%</td>
<td>3</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>12</td>
<td>4.01</td>
</tr>
</tbody>
</table>
Table 1 shows the scores of the sub-criteria of OpenAIS final candidate and Zumtobel LITECOM. The corresponding scores of the criteria are given in Table 2.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Sub-Criteria</th>
<th>Weight</th>
<th>OpenAIS Final Candidate</th>
<th>Zumtobel LITECOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Efficiency</td>
<td>Standby power consumption</td>
<td>40%</td>
<td>1.6</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>Operational power distribution efficiency</td>
<td>30%</td>
<td>1.2</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>Efficiency of controls</td>
<td>30%</td>
<td>1.5</td>
<td>5.5</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>4.3</td>
<td>14.6</td>
</tr>
<tr>
<td>Business Control Points (vendor differentiation)</td>
<td>A vendor can deliver and use device functionality up and above the standard features</td>
<td>50%</td>
<td>4.2</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>A vendor can deploy additional system functionality to all networked devices</td>
<td>30%</td>
<td>0.9</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>A vendor can detect and use additional functionality supplied by a foreign device</td>
<td>20%</td>
<td>0.6</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>5.9</td>
<td>4.5</td>
</tr>
<tr>
<td>Manufacturing, Installation and Commissioning cost (per luminaire)</td>
<td>Device Cost of „Connected“ Converter (€) [measured @ LED 20W @ purchase level by the luminaire company]</td>
<td>50%</td>
<td>4.2</td>
<td>14.4</td>
</tr>
<tr>
<td></td>
<td>Installation &amp; connection cost (€) [measured @ 4m connection to next device in false ceiling]</td>
<td>30%</td>
<td>1.2</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>commissioning and fault finding cost (€)</td>
<td>20%</td>
<td>5.1</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>14.3</td>
<td>14.5</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Integration of new connectivity technology</td>
<td>20%</td>
<td>4.8</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>Integration of new sensor (ubiquitous video) or actuator (blinds)</td>
<td>50%</td>
<td>2.5</td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>Change of IoT standards (e.g., CoAP replaced or new discovery mechanism)</td>
<td>30%</td>
<td>1.5</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>11.6</td>
<td>11.5</td>
</tr>
<tr>
<td>R&amp;D effort</td>
<td>R&amp;D cost for first system development</td>
<td>35%</td>
<td>1.05</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td>R&amp;D cost for component development in new architecture</td>
<td>45%</td>
<td>1.8</td>
<td>3.15</td>
</tr>
<tr>
<td></td>
<td>Transition cost of the organization (training, hiring)</td>
<td>20%</td>
<td>0.6</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>2.65</td>
<td>4.2</td>
</tr>
<tr>
<td>Scalability</td>
<td>Total costs for adding devices and / or functionality</td>
<td>34%</td>
<td>1.7</td>
<td>4.13</td>
</tr>
<tr>
<td></td>
<td>Continuous functionality / UI independent of size</td>
<td>33%</td>
<td>1.32</td>
<td>4.15</td>
</tr>
<tr>
<td></td>
<td>Performance independent of size</td>
<td>33%</td>
<td>1.32</td>
<td>4.13</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>13.4</td>
<td>12.4</td>
</tr>
<tr>
<td>Availability</td>
<td>Availability</td>
<td>50%</td>
<td>2.2</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td>Recoverability</td>
<td>50%</td>
<td>0.9</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>Degradability</td>
<td>20%</td>
<td>0.8</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>4.9</td>
<td>4.7</td>
</tr>
<tr>
<td>Time to Light (TTL)</td>
<td>Synchronicity when switching / dimming a group of devices</td>
<td>35%</td>
<td>1.4</td>
<td>5.17</td>
</tr>
<tr>
<td></td>
<td>Startup time until the system is in regular state (up and running)</td>
<td>15%</td>
<td>0.45</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>13.8</td>
<td>13.7</td>
</tr>
<tr>
<td>Integration Level</td>
<td>Integration level</td>
<td>25%</td>
<td>1.25</td>
<td>5.12</td>
</tr>
<tr>
<td></td>
<td>Monitoring level</td>
<td>25%</td>
<td>1.25</td>
<td>5.12</td>
</tr>
<tr>
<td></td>
<td>Functionality</td>
<td>50%</td>
<td>2.5</td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100%</td>
<td>14.5</td>
<td>13.5</td>
</tr>
</tbody>
</table>
Table 2: Scores of criteria of OpenAIS final candidate and Zumtobel LITECOM

<table>
<thead>
<tr>
<th>Criteria</th>
<th>OpenAIS Final Candidate Architecture</th>
<th>Zumtobel LITECOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of handling (maintenance, installation) for Electricians</td>
<td>3.5</td>
<td>3.4</td>
</tr>
<tr>
<td>Ease of handling (maintenance, installation) for IT</td>
<td>4.45</td>
<td>4</td>
</tr>
<tr>
<td>Ease of handling (maintenance, installation) for providers (lighting)</td>
<td>4.34</td>
<td>4</td>
</tr>
<tr>
<td>Interoperability with building automation systems</td>
<td>4.5</td>
<td>4</td>
</tr>
<tr>
<td>Use of open standards</td>
<td>4.35</td>
<td>1.35</td>
</tr>
<tr>
<td>Reuse from IT domain</td>
<td>4</td>
<td>2.4</td>
</tr>
<tr>
<td>Upgradability to this architecture from existing architectures</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td>Compliance with functional requirements</td>
<td>4.8</td>
<td>4</td>
</tr>
<tr>
<td>Security</td>
<td>4.01</td>
<td>3</td>
</tr>
<tr>
<td>Power efficiency</td>
<td>4.3</td>
<td>4.6</td>
</tr>
<tr>
<td>Business control points (vendor differentiation)</td>
<td>3.5</td>
<td>1.4</td>
</tr>
<tr>
<td>Manufacturing, Installation and Commissioning cost (per luminaire)</td>
<td>4.2</td>
<td>4.5</td>
</tr>
<tr>
<td>Extensibility</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td>R&amp;D effort</td>
<td>3.45</td>
<td>2.7</td>
</tr>
<tr>
<td>Scalability</td>
<td>4.34</td>
<td>4</td>
</tr>
<tr>
<td>Reliability (Error resilience)</td>
<td>3.7</td>
<td>3.7</td>
</tr>
<tr>
<td>Performance (response time, synchronicity)</td>
<td>3.85</td>
<td>4.35</td>
</tr>
<tr>
<td>Emergency lighting integration capability</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>75.79</td>
<td>65.4</td>
</tr>
</tbody>
</table>

Figure 2: Spider diagram comparing Zumtobel LITECOM and the final candidate architecture
The spider diagram shown in Figure 2 provides the comparison between the baseline architecture and the final candidate architecture. Compared to the baseline strong aspects of the OpenAIS selected candidate architecture are:

- Use of open standards
- Reuse from IT domain
- Business control points
- Security
3 REFERENCE ARCHITECTURE

This chapter provides an extensive overview of the proposed reference architecture for indoor office lighting systems. It starts with a brief overview of system context and the main architectural concepts that will be applied. This will be followed by a summary of derived technical requirements to be taken into account in the architecture, the top-level decomposition of the system and a detailed description of the various architecture views (Sections 3.3-3.6).

3.1 Introduction

3.1.1 System Context

An office lighting system operating in a complex building context is shown in Figure 3.

In this figure we identify the following elements composing the context of the system under discussion. In the centre of the picture we see an office lighting system in a large office building with its various components like luminaires, sensors and connections distributed over multiple areas and multiple floors. This system interacts with many actors in its environment which are listed below:

- Building Automation System, that controls the behaviour and maintenance of the entire building.
- Other building systems like: blinds, HVAC, security, safety, elevators, power etc.
- The IT-infrastructure of the building: Components of a connected lighting system are integrated in the IT-infrastructure of the building. Also parts of the
software of the system may run on general purpose hardware which is part of the IT-infrastructure.

- Cloud storage and data analytics: Connected lighting systems produce a lot of data which may be used for new types of services beyond lighting supporting maintenance (e.g. cleaning schedules) and consultancy (e.g. for energy reduction and space usage optimization) and by the lighting supplier or third parties.
- Installation; different companies and personnel perform the actual installation of the equipment.
- Commissioning: specialized personnel executes the final commissioning of a lighting system.
- Ad hoc user devices like mobile phones and tablets may be used to control the lighting system.

3.1.2 Basic Concepts

The OpenAIS architecture is based upon the concepts and frameworks of emerging IoT standards as much as possible. The expectation underlying this choice is that lighting will be “just” one of the many devices connected to the internet in the offices of the future and therefore will have to adhere to these standards and not follow a separate or dedicated lighting path. The IoT-domain is strongly developing and we expect that it will take some years before winning technologies and leading standards will emerge.

The main characteristics of the OpenAIS reference architecture are:

- A reference architecture description in which not all technology choices are completely fixed, allowing evolution over time.
- Fully IPv6 based solution (IP-to the end node).
- Support of both wired and wireless solutions in a single system, with a definition of minimum requirements for connectivity.
- Preferred physical connections will be indicated. However, the OpenAIS reference architecture will be a platform allowing arbitrary connectivity standards to be used, as long as they comply with the mentioned minimum requirements.
- Integration of old (e.g. DALI and other) installations will be enabled through gateways.
- Allowing flexible allocation of Control functions, supporting both distributed and centralized control with optional hierarchy.
- Scalable architecture to support from small system setups towards campus wide systems.

3.1.3 Architecture Views

The following views will be used in the description of the OpenAIS architecture:

- Logical (or functional) View:
  - How the various functions of a Lighting system are logically structured (decomposed), and how they can be extended (after installation in an actual building). This view includes the functions for control, light point actuators, for sensor data acquisition, data accumulation, reduction, conversion and storage. It also describes the interfaces the functions expose to each other and to other entities like management systems and applications.
3.2 Derived technical requirements

3.2.1 Introduction

The reference architecture to be defined in subsequent sections will fulfil all the user requirements listed in the previous chapter and the references therein. However, there are a number of detailed (derived) technical requirements which are listed here before the actual description of the architecture. In this chapter we will discuss the following aspects:

- Error and recovery handling: Where most of the requirements described in Section 2.2 deal with the desired user functionality attention also needs to be paid to non-standard situations and undesired events like communication errors and broken devices.
- Time synchronisation is an important “derived” requirement needed to comply with a number of the functional/user requirements.
- Security: The security requirements of the system are specified in detail based on an analysis of the main threats.
- Out-of-the-box functionality: Ease of installation and commissioning are main system requirements which are partly “translated” in the requirements for so-called out-of-the-box behaviour.

3.2.2 Error and Recovery handling requirements

User and system requirements mostly deal with the standard or “happy flow” behaviour. However, it is also needed to define the unusual or “unhappy flow” behaviour as recovery and handling of errors and resets. The main items to be addressed are:

- “Missing resource” which can have many causes:
  - Device software hangs or has crashed
  - Network connection lost
  - Hardware broken either in device or network
  - Device power lost
- Etc.
  - “Resetting of a resource”
    - Device has recovered after a period of absence
    - State and messages may have been partially lost

The main requirements to be covered by error and recovery handling are:

- “Safe” default behaviour: Whenever errors are detected (e.g. higher level Control no longer available or sensors no longer available) the system should activate a “safe” state. E.g. when the presence detectors malfunction, the area Control may go to a 50% light level. Whenever an actuator notices its control is no longer active it may revert to local sensor control only, or also set some base level intensity. The architecture does not exactly prescribe all “safe” behaviour and levels, these should be configurable.
- Containment of errors: Errors in a device should not propagate or cause further propagation of errors in the system. So the system should be prepared for “missing” or inconsistent devices and handle these.
- Layered functionality: One of the main “containment” measures is layering in control. Whenever building-level control fails, area/floor control will still operate, and when area control fails local luminaire control will take over.
- Recovery and “back on the rails”. Whenever a device resets, any resulting state inconsistencies (e.g. the light is off where it should be on) should recover within a few minutes.

3.2.3 Time Synchronized Operations Requirements

Time is an essential element of a number of operations in a lighting system. The most important requirements concerning time in the context of this architecture document are:

- Time stamping of logging/tracing events on multiple devices. Accurate synchronization of the time(s) is needed here to allow for debugging the system. For debugging it is important to be able to reconstruct the exact sequence of events and message passing from the collected logging information. Accuracy (of the time synchronization) required for message flow analysis is in the millisecond range.
- Time stamping of data collected at sensors. For off-line analysis a good insight in the timing of sensor data is needed. As most of these data are not high frequency the timing accuracy requirements are quite relaxed, mostly in the seconds range.
- The exception to the above rule on sensor data are situations where a combination of sensor data is used in real-time algorithms, like tracking of persons. It will be needed in these cases to have synchronization between the timestamps in the order of 100’s of milliseconds.
- Setting of values for a group in a synchronous way. E.g. dimming or switching of luminaires in a room. It is important that the synchronicity of these settings is in the order of 200 milliseconds.

3.2.4 Security requirements

3.2.4.1 Security threats

The main security threats as identified for OpenAIS are described below:
• Unauthorized operation of the lighting devices in a building leading to potential harmful situations. An intruder could change the desired state of the lighting rendering a room or area unusable or even dangerous. Examples of such actions are: unauthorized switching on or off the lights, creating changing patterns or flickering, lighting up an entire building at night.
• Data privacy threats: unauthorized use of the operational data of the lights, e.g. detailed timeline lighting profiles that give substantial information about the activities in the lit area, especially if presence sensors are in place.
• System integrity threats: unauthorized use of the resources of the lighting control system e.g. for sending spam.

Such threats could materialize through a variety of attacks:
• Over network connections into the building exploiting the networked character of the system by “faking” a network command to the lighting, executed by applying all known “hacking” approaches.
• Eavesdropping or tapping of any communication within the system or between the system and the back-end in the cloud
• Compromising or replacing devices and/or their software so that they are under hackers’ control. This can be through network attacks (downloading malicious code) as well as direct physical attacks, even so far as buying a luminaire to compromise it.
• Stealing security keys from service, facility management, installation and commissioning personnel.

Note that there are quite a few attacks that are outside the scope of the OpenAIS architecture. One can think of attacks on the data stored in the cloud, hacking or breaking into the building management systems or into the general IT-infrastructure of the office. These kinds of attacks are outside the scope of OpenAIS. The architecture will focus on network attacks and to a lesser extent on physical attacks on the devices, either to directly control the devices or to run malicious software on it.

3.2.4.2 Access levels
Lighting systems require various levels of access with different rights. In the commissioning phase much more is allowed than in normal operational mode. We identify the following modes of operation from a security point of view:

• Level 0: Object detection. This level is used for out of the box functionality, device discovery and service detection.
• Level 1: Reporting only. This level allows access to sensor and other (relatively uncritical) operational data and the device error status. The operation of the system cannot be influenced using this Level.
• Level 2: Standard use. This level allows access to all operational features of a device, including access to operational parameters.
• Level 3: Commissioning use / parametrization services. This level gives access to parameters that commission the operation of the system but does not allow the changing of any grouping or binding information.
• Level 4: Commissioning use / localization and addressing services. This level allows access to all services and parameters including grouping and binding information.
• Level 5: Device Owner. This level allows the change and upgrade of the software of the devices and also allows the device owner to configure the rights/permissions of some security Objects.

The above access levels require the system to support role-based access.

3.2.4.3 Network security requirements

The following main requirements for the lighting network security are defined:

• Role-based authorization for messages to the lighting devices.
• Secure communication between devices and between groups of devices (more precisely between the functions like Sensors, Control and Actuators). This means in detail:
  o Only authorized devices may receive and process messages.
  o Only authorized devices may send messages (or more precisely only messages from authorized devices will be processed)
  o The integrity of messages received must be verified (and so be verifiable)
  o An eavesdropper must not be able to interpret an intercepted message.
  o A message that has been modified by an eavesdropper must not be processed.
  o An intercepted message if replayed must not be processed (note that we will not do anything about an intercepted and unmodified message that is received only once).
• Granularity of security settings per function (Sensors, Control and Actuators, DataCollect).

3.2.5 Out-of-the-box functionality

3.2.5.1 Introduction

All OpenAIS devices are prepared to have some “out-of-the-box” functionality that allows for a preliminary operation of the system before commissioning starts. This out-of-the-box functionality is separated into two parts:

• The single device functionality: the behaviour of a device without any network connection available.
• The networked functionality: The behaviour of the devices that automatically joined an ad-hoc-network without any commissioning.

The basic goal of the out-of-the-box functionality is to check the device integrity and connection quality of the devices, and to allow for some rudimental functionality before commissioning happens. The out-of-the-box functionality does not replace the commissioning effort, commissioning is always needed to secure the IP communication of the system against foreign influence.

The out-of-the-box functionality is valid for non-commissioned systems only, and will be completely replaced by the chosen operational settings with the commissioning of the system.
3.2.5.2 Out-of-the-box single device functionality:
Every OpenAIS device shall have an out-of-the-box “disconnected” functionality that allows the checking of the integrity of the device (self-test result) and the power status of the device also without any network connection available.

The use cases are:
- The electrical contractor gets an “ok and connected to power” signal when mounting / powering the system, and can use that to validate acceptability of the task performed. This supports the “installation check” step, that is part of the system installation & commissioning workflow, and may be part of the commercial workflow (handover / sign off for the installation contractor, if the site is organized like this).
- The site is lit for all the other on site labour performed.

The required features
- Light outputs should switch the lights on when power is available and no internal error occurs.
- Outputs that cannot visibly actuate their output after power up (e.g. motors of blinds that are not allowed to move after power up until the blinds are fully commissioned) should provide an optical feedback (Status LED) that show the internal self-tests are okay.
- Sensors should provide optical feedback after power up and test ok, and also when actuated.
- Other system devices also show the status LED for optical feedback.
- Standalone mini-system (e.g. a luminaire with sensors and a local embedded controller) Lights should go to ON /100% after power up and test ok to allow for a visual check. Dimming and switching to OFF according to available daylight should commence after 20 seconds full ON only.

3.2.5.3 Out-of-the-box networked device functionality
Every OpenAIS device will have a pre-configured embedded network functionality that operates at the moment they connect to the physical network. This functionality will be completely disabled after commissioning.

The use cases:
- The system devices can be checked for performance and network connection quality.
- The site system delivers light whenever needed. Lights can be switched on and off, and PDs switch off the lights when the site is empty.

The required features:
- All toggle switches toggle all the (not yet commissioned) lights on the same network on/off.
- All dimming switches change the dimming level of dimming luminaries according to their dimming value (if many switches are part of the system always the last control action prevails, i.e. the devices act using the principle “latest request served”), and reverts slowly to the default value after a while.
- All additional control (e.g. colour control) change the respective additional setting where available (again last request served), and reverts to the default after a while.
• The scene buttons of a scene panel set the lights to specific dim levels when activated (like the dimming switches), the lights revert to the default after a while.
• Any presence detector sensor switches lights to full on when new presence is detected and lights have been off.
• All lights are slowly dimmed to off when all the presence detector sensors detect no presence any more (only valid after a presence was detected).
• Light sensors will dim up/down the lights in a 50% step whenever a sudden change in sensor level is detected. (50% up when sensor reading suddenly changes to “low light”, and 50% down when the sensor reading suddenly changes to high brightness; the thresholds for the “sudden change” are set in a way that these changes do not occur in normal operation.
• Standalone mini-systems (e.g. a luminaire with sensors and a local embedded controller) decompose to sensors and actuators that perform the tasks as shown above.
• There is no mobile device remote control usage with the out-of-the-box functionality, mobile control always needs commissioning.

Please note that some network connection may need some additional manual interaction: Press a “connect button”, tag with NFC, or alike. It is expected that the physical network connection will be straightforward and made easy, and that it can be considered as part of the electrical contractor’s work. (Some physical media have special workflows for connecting new devices to the network, but also these evolve over time to more useful and easy handling.)

3.3 Logical View

3.3.1 Introduction

In this view we will describe the decomposition of the OpenAIS system in a number of logical functions and their relations. For every logical function a detailed description is provided. We will also describe the interfaces exposed by each of these functions as well as the dynamic behaviour and message flows in the system. It will give an introduction to the Object Model that details all elements of the decomposition and their interfaces. The Object Model will be released as a separate deliverable D2.4 in M24 [OpenAIS_D2.4].

This view also describes the extension mechanisms build into the architecture, allowing evolution over time and upgradability in installed systems. A number of generic mechanisms for error handling, recovery and graceful degradation will also be discussed. Finally, we will discuss the out-of-the-box design that provides a set of functionalities that eases the life of electrical contractors and commissioning personal.

3.3.2 Decomposition

3.3.2.1 Introduction

In the logical view we present a functional decomposition of the system in a number of “main functions. In this decomposition we first divide the system in two “layers” or “clusters” of functions of different character.

• The Application layer: This layer contains all functions that implement the actual lighting functionality like making light (Actuators), sensing presence,
light-level etc. (Sensors), Control of lighting systems, data collection (DataCollect) as well as the interfacing to legacy systems through a Gateway. Furthermore this layer incorporates a number of supporting functions for grouping and scene setting.

- **The Infrastructure layer:** This group of functions takes care of the infrastructure of the system like Communication, Discovery, Configuration and (software) Update.

The logical decomposition is shown in Figure 4.

![Logical decomposition of OpenAIS architecture](image)

*Figure 4: Logical decomposition of OpenAIS architecture*

In the application layer we identify the following functions:

- **Control function(s)** that implement the algorithms for automatic (lighting) behaviour like reacting on sensor values and setting the values for the actuators.
- **DataCollect function(s)** that implement the (intermediate) collection and processing of data from sensors and other functions and forward these to more permanent storage outside the scope of this architecture (somewhere in the cloud or BAS).
- **Actuator function(s)** for the actual generation of light.
- **Sensor function(s)** like presence detectors, light sensors etc.
- **Group function(s):** Control function (often) work on an arbitrary number of actuators, and sensor function(s) may distribute their data to multiple control functions. The group function supports the implementation of this grouping concept.
- **Scene function(s):** A scene is a set of actuator settings that together form a scenario, creating a specific effect e.g. in a presentation room when switching to "presentation setting". The scene function supports the implementation of this concept.
Note that in the application layer also a gateway function is identified. This will be used to integrate OpenAIS systems with existing lighting installations and legacy equipment. The gateway represents the legacy system in OpenAIS and hence is situated at the application layer level. Of course the gateway will also contain some (legacy) infrastructure, but this is not depicted in the infrastructure layer as it is not part of OpenAIS.

Below these application functions we find the infrastructural functions that support the following functionality:

- **Discovery**: Detects the available functions (Actuators, Sensors, Control and DataCollect, etc.) in the system.
- **Communication**: Supports a communication infrastructure between the various functions in the system. This function is described in detail in the network view.
- **Update**: Allows update of software of the system.
- **Security**: Supports authorization, authentication, confidentiality and security of the communication and the integrity of the system against attacks.
- **Configuration**: Supports update of the “static” parameters in the system.
- **Device**: This is the container for the properties of a physical device and implements the functions and parameters that relate to a single device like its IP-address, MAC address, reset, power states, health status etc.

The application layer implements the domain specific functionality that forms the key added value of a lighting system and will be the main area of differentiation between different vendors implementing an OpenAIS system. However, the infrastructure will be implemented using commodity technology as much as possible, as we will see in the physical view where we map these concepts on technologies.

Finally, we will describe a number of generic mechanisms in the architecture that are not strictly allocated to a single element of the decomposition but are aspects of each of them. These are the so-called cross-cutting concerns.

The OpenAIS architecture focuses primarily on the control and communication structure for the lighting systems. The definition of the architecture for the back-end, server or cloud architecture is outside the scope of OpenAIS architecture.

Note that although we describe a functional decomposition, we will extensively make use of “object oriented” terminology to describe the properties of these functions in the remainder of this document.

### 3.3.2.2 Relation between objects

In this section we will detail the specifics of the various objects as defined in the decomposition in Section 3.3.2.1. Furthermore, we will detail the interaction between these objects. The main control paradigm of the OpenAIS architecture is depicted in Figure 5:
Multiple (indicated with n in the picture) Control Object(s) may be linked to multiple (n) Sensor Objects and to multiple (n) Actuator Objects. Control Objects react upon any changes in the states/data of these Sensors and (based on the algorithm) set the Actuators. This is the core of any lighting control system.

In Figure 6 the relations between the “Objects” in the application layer are depicted in more detail:

In Figure 6 we find the DataCollect Object on top that can be related to (n) Sensors but also to the Actuators and even to the Controls, that all have interfaces in which they expose data for storage and analysis. Likewise, the Control Object can be related to arbitrary sets of sensors and Actuators. Control Objects can also relate to other Control Objects, a stacking concept which will be elaborated later. Finally, there are
the Scene Objects and Group Objects that support the operation of the other objects in
the application layer.

The relations between the infrastructure layer blocks are depicted in Figure 7. Note
that in this figure the term “Application” is used to indicate one of the elements from the
application layer like the Actuator, Sensor or Control Objects.

We see that the Device is being updated by both the Configuration (parameter
settings) as well as the Update function (software update). Discovery, Update and
Configuration all make use of Communication where Communication uses Security.
The static parameters of the Application Objects are being updated by the
Configuration function while the Application Objects make use of Communication and
Discovery.

![Diagram: Relations between functions in the infrastructure layer]

**Figure 7: Relations between functions in the infrastructure layer**

3.3.3 Application Layer

3.3.3.1 Control

Introduction
The “Control Objects” in the system determine the settings of actuators based on the information from the sensors, user information, state information from other Control Objects, time and history. Control Objects can receive sensor updates on the actual values/states measured by the sensor. A Control Object then sets the actuators by directly addressing the interfaces they offer for this purpose. Examples of typical Control Objects are: occupancy based lighting control, daylight harvesting based lighting control, user control/on/off/dimming.

Control Objects may be stacked, i.e. a Control Object itself may be controlled by another Control Object as will be explained in the next paragraph.

For the implementation of its functionality the Control object may make use of the Group and Scene Objects.

Stacking of Control Objects
The OpenAIS architecture allows for flexibility in allocation and distribution of Control Objects. Basically all models are supported: from allocation to one central controller that is handling all lighting control, to a network with fully distributed control in all luminaires. OpenAIS presents a reference architecture suitable for many companies and for a period of at least 10 years from 2020 onwards. Therefore, it must be very flexible in allowing new developments. This inherent flexibility implies that there are many possible and allowed solutions to a single (control) problem. Therefore, the solutions and sequence diagrams presented in this section are “just examples” and are not mandatory structures for OpenAIS compliant systems.

Note also that complex constructs are conceivable using stacked and overlapping control deployment. It might be attractive for engineers to dwell upon the boundaries of the architecture, but these are not preferred in practice and the majority of the functionality can be achieved with simple one or two layers of control. Only in exceptional situations complex constructs will be needed. The examples presented in this section reflect that drive for simplicity.

Stacked control concepts can be used in the following example circumstances:

- Override: An additional Control Object is added to a system (after installation) with “advanced” behaviour that overrides the behaviour of default controllers which may be located in a luminaire.
- Levels of control: There are different Control Objects for behaviour at various levels. So a potential configuration could be one Control Object per floor and another per room.
- Fall-back: a Control Object is designed to jump into operation when another control object is failing.

Control Objects may also support control and data interfaces. In this way Control Objects may be stacked in a hierarchical fashion using these interfaces. In the next subsections we discuss a number of examples of the use of the stacked control concept.

3.3.3.2 Sensor
Sensor Object(s) implement the control of and interfacing to the actual sensors of the system. This Object gathers and pre-processes data from the physical sensor devices and (can) forward these data to interested observers. It shields and compensates
where possible any specifics of the physical devices and offers an application level interface to its clients.

Examples of sensor Objects are infrared presence detectors, daylight sensors, but also temperature and other climate condition sensors used in lighting systems. Also sensors for other building management purposes may be linked into the OpenAIS lighting network.

Sensors inherently can be monitored by multiple Control Object Instances, that each would receive similar information at the same rate depending on the configuration of the Sensor Object. As there may be situations in which different listeners have different requirements on sensor processing, update frequencies etc. multiple instances of a logical Sensor Object may relate to the same physical sensor. Each of these will receive the same basic sensor data from the physical sensor but may have different settings for e.g. processing.

### 3.3.3.3 Actuator

Actuator Object(s) execute the control and management of actual settings of devices like LEDs (drivers). It shields and implements where possible all specifics and physical details of the devices from its clients and offers application level interfaces for setting light level, colour, transition behaviour, focus control, direction control etc. They implement the safety and regulation control for the proper functioning of the physical device e.g. maximum power/current control. The Actuator Object(s) also offer data coming from the actuator devices like actual energy consumption, burning hours, actual state and malfunctions, as monitoring data.

Examples for actuators are of course LED light points, dimmable light points, colour light points, tuneable white light points, etc.

Actuators will be controlled by "one control object" usually. However for flexibility and extensibility the architecture provides a multiple- control interface mechanism. The actuator may receive commands from different control objects. There are two modes for this:

A basic mode supported by all Actuator implementations: The requests will be handled as if coming from a single Actuator: Last command served.

In systems where the Actuator needs to receive and separate signals from different Control Objects the Actuator uses multiple Actuator Object instances, each one bound to a specific Control Object. These object instances share the same physical actuator with all its physical properties (actual achieved position, output temperature etc.), but remain different in all the logical properties (last requested action, scene values, etc.). The rule set that transfers the request from the (selected) logical object instances to the actual physical actuator may be actuator specific, a relatively simple but powerful example would be an additional parameter "Priority" per object instance that controls which instance is prevailing. (The others would report a "prio-warning" together with the actual and requested position in their status reports) If the operational switchable priorities are used a fallback mechanism needs to be implemented that allows for recovery if the higher prior channel control is stuck or broken during higher priority operation.
3.3.3.4 DataCollect
The DataCollect Object implements the collection, reduction, pre-processing, and intermediate storage of data. It forwards these to more permanent storage outside the scope of this architecture (somewhere in the cloud or BMS).

The data collector can be configured to store one or more data streams from any source identified in the architecture. It can also combine data across multiple sensors. It contains storage of configurable size that can be in memory, flash or on disk (this is also configurable). It can also contain algorithms for reduction and pre-processing of data. It also incorporates a forwarding function to forward accumulated data either on request or scheduled. The forwarding policy and final destination of data can be configured.

The DataCollect Object interfaces at one hand with all the data generating Objects from OpenAIS, first of all Sensors, but also the Actuators (status data, information on light points) and even the Control Objects. At the other hand the DataCollect Object connects to a back-end which may be in the cloud for more permanent storage. A DataCollect Object may interface to numerous Sensors, Actuators and Control Objects.

Note that the DataCollect Object may be stacked also. So a tree like structure may be created of DataCollect Objects. The actual layout is at the discretion of the system designer in which elements as available storage (at a location in the network) and communication band-width will play a role.

3.3.3.5 Group

Overview
In lighting many tasks are related to a set of Actuators or Sensors. E.g. when switching Actuators, it is often a set of Actuators that is switched, usually called a group. The same applies to Sensors: E.g. a central daylight sensor will deliver its measurements to many Control Objects that technically also form a group. We are speaking therefore technically of a Control Object controlling a group of Actuators, and of a Sensor Object delivering its measurements to a group of Control Objects.

The Group Object identified in the decomposition implements all the functionality supporting groups and grouping of Actuators and Sensors. Each Object (Control, DataCollect, Sensor, and Actuator) which may be involved in grouping uses the Group Object to administer or retrieve group details. For each group an application Object uses, or is a member of, it will relate to an instance of a Group Object that is specific for this group. Main parameter of the Group Object is the Application Group ID that uniquely identifies an Application Group. Further information shared through the Group Objects is security related and (multicast) communication related information.

An Application Group can exist across multiple OpenAIS devices. On each OpenAIS device, multiple Object instances can become member of the group. These Object instances may even be of a different type.

Group Parameters
A group is described by three main “group” parameters:
• **Application Group** – the group visible at application level; comprising of a set of Object instances that have been configured to respond to events or group commands in a mutually consistent manner.
  o An Application Group is represented by an instance of a Group Object.
  o An Application Group ID is a system-wide unique identifier of an Application Group.
  o An Application Group, groups similar entities like Actuators, Control Objects.

• **Multicast Group** – a set of IPv6 nodes that subscribe to the same multicast IPv6 address in order to receive group messages. This is detailed further in Section 3.5.6 as part of the Networking View.

• **Security Group** – a set of sending and receiving nodes that share a common security domain; such that any sending node is able to securely send a message to all the receiving nodes via a group message. The group messages can be for example encrypted, tamper-protected or replay-protected such that non-group-members cannot respectively view, tamper, or replay the message. Security for group communication is detailed further in Section 3.5.6 and in Section 3.6.5.3 in the Security View.

**Group Relations and Group Vector**

The Application Group is the group visible at the application level, for example to address a group of luminaires in a room. The other group parameters need to follow and technically support the Application Groups with a minimum of complexity. For this reason, a 1:1:1 ratio of Application/Multicast/Security Group is preferred. This means that each Application Group should use its own, exclusive Multicast Group associated and also its own, exclusive Security Group not used by others.

However, in some cases there may be a need to deviate from this ideal 1:1:1 ratio due to resource restrictions on constrained embedded devices. For example, the number of Multicast Groups a device can support may be limited and/or the number of Security Groups may be limited.

In such cases, the following additional rule set can be applied:

- Multiple Application Groups may use the same Multicast Group for performing group communication.
- Each Multicast Group should be associated to its own, unique Security Group

In case the number of Security Groups needs to be reduced even more to conserve resources, the following additional rule set can be applied:

- Multiple Application Groups, even though they use different Multicast Groups for communication, may use the same Security Group for securing the group communication.

The term *Group Vector* is introduced here as a shorthand for the vector data type that contains three pointers to the specific three group types, as follows:

\[(\text{Application Group, Multicast Group, Security Group})\]

Any Object instance that is a member of an Application Group therefore is automatically related to a specific Group Vector, which can be constructed from the membership information plus the multicast and security information stored in the
Group Object. In an OpenAIS system with N Application Groups defined, there are exactly N different Group Vectors.

In all cases, any OpenAIS device must be prepared to handle situations where
- multiple Application Groups are using one and the same Multicast Group, or
- multiple Application Groups are using one and the same Security Group, even when these Application Groups use different Multicast Groups.

### 3.3.3.6 Scene

A scene is a set of actuator settings that together create a specific effect (In lighting: a specific (colored) light distribution creating a lighting effect, e.g. in a presentation room when switching to “presentation setting”). The Scene Object supports the implementation of this functionality in the OpenAIS architecture. A Scene Object includes the target Actuator settings (values) and may include a transition pattern. Transition patterns may be specific per Actuator. Scene Actuator values may be given by explicit value or by (algorithmic) reference. (Naming: scene settings by value are named "standard scenes" and scenes using referenced values are part of an "extended scene") The Scene Object instance in this case provides the parameters needed to calculate the actuator values depending on e.g. time or sensor value(s).

Scene Objects are an integral part of the Control Object, all Scenes apply to the group of Actuators the Control Object controls. Each Scene Object instance manages one scene, and only one scene is active at a single time.

In areas where multiple Groups in one room have different Scenes active the total effect for the room is a kind of combined Scene, combining the effects of all Scenes acting at the same time; this "combined Scene" can be seen as a Scene by itself that is defined on the level of the room Control Object, using the group Scenes as "group_actuator scene values". Therefore also stacked higher level Control Objects handle Scenes, by using the Scenes provided by the lower level Control Objects that handle the Actuators.

### 3.3.3.7 Gateway

Gateway function(s) take care of the adaptation and integration of legacy or other non-OpenAIS compliant systems/interfaces, networks or devices to the OpenAIS architecture. This Object is therefore at the application level in network terms. In the gateway, the Objects present in the “legacy” or “non-OpenAIS” network are represented as normal OpenAIS Objects towards the rest of the OpenAIS network. Examples of devices/networks that can be converted using gateways are:

- DALI networks
- Low power devices unable to run an IPv6 network

Note that the Gateway is a placeholder in the architecture. The OpenAIS project will not go further in defining any gateways for specific legacy systems. This is left for the implementation of the architecture in specific projects.
3.3.4 Infrastructural Layer

3.3.4.1 Communication
The Communication function implements the communication infrastructure in OpenAIS. It implements the functions to securely transport commands and data between the various modules of the system over a network. In OSI terminology this module implements all functionality of the first 6 layers of the OSI-model, only the application layer functions like Control, Actuator, and Sensor are not part of the Communication function but build upon it.

It is expected that this function will be mainly mapped directly on existing networks, protocols and transport mechanisms. The OpenAIS project will not be defining its own Communication protocols or mechanisms - only when missing items are identified; extensions to current standards will be suggested. This will be discussed in detail in the networking view.

3.3.4.2 Discover
The Discover function takes care of the discovery of all the available (application) functions (Actuator, Control, DataCollect and Sensor), in the system. Note that the required functions for operation are configured in the system itself (e.g. in the Control functions which know which sensors they have to observe). As such this is a function that is primarily used during commissioning. It also allows the OpenAIS system functions to check if the configured functions are indeed available for fault finding and error logging.

This function consists of two parts. At the application functions (Actuator, Sensor etc.), there is a publishing mechanism through which the function announces its available capabilities. At the Infrastructure functions side there is a mechanism to identify which functions are present. This can be used by the system at start-up or during regular checks.

3.3.4.3 Device
The Device function is the container for the properties of a physical device. Examples of these are software and hardware version numbers and of course the software in the device itself. This function also implements power states and reset/start/initialisation functionality of the physical device. The parameters of the Device function are set through the Configuration function. The software is loaded through the Update function.

3.3.4.4 Configuration
The Configuration function controls the configuration parameters of the system. These are the settings that are not modified during normal operation. Examples of these parameters are maximum levels, timing parameters, network addresses, keys, regulation curves and any other parameterized elements of their functionality. Another important aspect of the Configuration function is commissioning in which primarily the relationships in the system between Control, Sensor, DataCollect and Actuator functions is established. This includes the relationship of device IDs, addresses and physical locations, and the grouping of functions in (logical) areas, rooms and floors.

Static parameters are of course stored in the deployed functions, Controls, Sensors and Actuators. Every function like Sensor, Actuator etc., has its own configuration functionality. The central part of the Configuration function can access these
parameters through a standardized interface (defined in the Object model) and can offer a user interface through which they can be manipulated. Note that the UI of the tools that manipulate the Configuration data will not be standardized in OpenAIS; this is left open for vendor differentiation. For interoperability a database (off-site) will be maintained by OpenAIS vendors which describe the parameter sets for their devices. This allows the use of one generic tool to modify all configuration information in a single OpenAIS system with components from multiple vendors.

Note: This UI may be offered also for mobile devices or remote computers connected to the network. The configuration tools will be password protected and support user accounts, access rights and access levels.

The Configuration function at the server side maintains a database that is used to keep track of the full configuration of the lighting system. This includes devices, functions (Actuator, Sensor, Control, etc.), their identifications, versions, parameter configuration, addresses, commissioned binding relations between functions and locations. This information can also be extracted from “the field” and be re-stored in the devices after replacement or other failures.

3.3.4.5 Update
The update function takes care of the secure update of software in the OpenAIS devices. Note that software update is an optional feature which is highly recommended for OpenAIS devices but it is up to the vendor to decide on its support. The Update function consists of a device side part (which consequently must be present in every device that supports update) that executes the secure download and/or reload of software to the device. This part of the function also supports storage of the new software in permanent storage and its activation and potentially a roll-back option to return to a previous version (optional feature). If required (as this is a manufacturer specific option) also encryption/decryption and verification of the downloaded software may be part of this function. There is also a central part to this function that takes care of the actual downloads to the devices from a central location which is connected over IP to the device.

3.3.4.6 Security
Security is not an isolated function, but is distributed over the entire system. Therefore we have dedicated a separate view to security which is described in Section 3.6. Security protects the OpenAIS system against a series of potential threats which are elaborated in Section 3.2.4.1. These threats include:

- Unauthorized control of the lighting system over the network
- Unauthorized modification of lighting system configuration
- Unauthorized use of the lighting system data (e.g. presence information)
- Download, updating devices of the lighting system with malware either over the network or directly

In the security view Section 3.6 the overall security solution is described in more detail. The Security “function” only forms a part of the total security solution and mainly delivers support for the primitives that allow other layers and functions to execute secure communication. It therefore implements functions to support and accelerate:

- True random number generator
- Hashing primitives (SHA, SHA-2)
- Encryption primitives (AES, ECC, RSA)
- Key management and protected storage
The architecture does not prescribe if this functionality must be executed in hardware or software. The need for hardware acceleration needs to be derived from the capabilities of the chosen computing platform and the performance requirements.

3.3.5 Interface overview

3.3.5.1 Introduction

The various Objects from the decomposition above and their smaller entities (which will be specified in the OpenAIS object model), expose their functionality through interfaces that will be described in detail in the object model and in the networking view. However at an abstract level a number of “generic” interface categories can be defined which will be presented below. In subsequent sections it will be discussed in more detail which of the Objects exposes what interface categories, while the subsequent object model will define these interfaces in all detail. In the Networking and Physical Views the mapping of these interfaces on the chosen network/communication technology will be described.

In general we identify a few types of interfaces for the Objects in OpenAIS:

- **IControl**, control interface through which a caller can execute a certain method in that function, like set a light level or set a colour. The knowledge on when the method is invoked and with what settings is fully embedded in the caller.
- **IData**, data interface through which a function communicates data or changes in its data to the outside world. These data may originate from physical devices like presence detectors or may be generated by the function itself. Handling of these data is determined by the receiver of the data. The producer just multicasts these to all interested entities.
- **IConfig**, configuration interface through which static parameters can be set like addresses, commissioning information, algorithmic parameters, scene values, regulation curves etc.
- **IDiscover**: interface through which the element can be discovered on the network.
- **IDebug**: Interface to configure debugging functionality of the related objects and interfaces to trigger testing/debugging operations. This interface will not be separately indicated in the figures.

An IControl interface is not directly linked to a Control Object. Many Objects can have an IControl interface.

3.3.5.2 Actuator

The Actuator Object offers a control (IControl) and a monitoring (IData) interface. The IControl interface allows the Control Object(s) to set the state/level and all other characteristics of light points. The IData monitoring interface allows its caller to observe the status of the Actuator such as on/off status, dim level and additional values like: burning hours or energy consumption. Also status like malfunction of the underlying hardware is passed through his interface and can be observed.
3.3.5.3 Sensor

The Sensor Object offers a monitoring (IData) interface through which a caller can observe the data from the sensor. The monitoring interface also supports error/status information about the sensor. The module also supports interfaces for configuration like setting of timings/curves and characteristics of the sensor data processing and an interface for discovery (IDiscover).

3.3.5.4 Control

The Control Object offers an IData monitoring interface for its status (like on/off). Data interfaces can also express “aggregated” status for the entire area like “activityOn” to indicate that there is some sort of lighting level in an area, which information may then be used by “stacked” Control Objects or DataCollect Object(s). A Control Object can also implement IControl interfaces again, for example to clip the energy usage after a demand response event. So Control Objects can be hierarchically stacked as indicated above. These IControl interfaces can also be used by a UI to control the “Control Object” like setting a level for the entire area or even set a scene or control an individual luminaire from this Group. These interfaces will then be linked to user interfaces running on e.g. a personal control device like a mobile phone.
The Control Object also supports the standard interfaces for configuration and discovery.

Note that the interfaces shown here are the interfaces offered from the Control Object to its users (higher level or so called stacked Control Object, UI, configuration tool). Next to these interfaces the Control Object has two important “required” interfaces, the Sensors that are bound to its inputs and the Actuators that it controls.

### 3.3.5.5 DataCollect

The DataCollect Object has an IData interface to allow other entities to monitor its accumulated data (e.g. an agent that forwards data to the cloud storage). DataCollect Object(s) mainly monitor IData interfaces of Sensors (and to a lesser extent Actuators and Control Objects). Having an IData interface again on the DataCollect itself allows these also to be stacked like Control Objects. Next to the IData interface there will be an IControl interface for triggering storage routines or flush the data of the DataCollect. There will also be an IConfig configuration interface through which the following items can be set:

- Storage capacity
- Policies for forwarding, or pre-processing
- Items for commissioning like the Sensors to be monitored.

Finally, the DataCollect Object also has an IDiscover interface.

### 3.3.5.6 Group

Group Objects, which basically are supporting Objects to administrate the grouping of Actuators, Controllers, DataCollect and Sensors have an IConfig interface to add/remove members to/from the group. It also has an IData interface to retrieve information from the Group Object like group members and their identities.
3.3.5.7 Scene

Scene Objects are a supporting Objects for implementing the scene behaviour as described in the decomposition. Scene Objects are incorporated in Control Objects and have an IConfig, IData and an IControl interface.

3.3.5.8 Gateway

A gateway is a converter from a non-OpenAIS domain towards OpenAIS. It will therefore either implement Sensor, Actuator, Group or Control interfaces or a combination of those.

3.3.6 Dynamic behaviour and example message flows in OpenAIS

3.3.6.1 Introduction

OpenAIS has a Sensor, Control, Actuator structure in which the main information flow is from sensor to control objects and from control object to actuators. Multicast communication is used here for efficient group communication.

There is additional information flow from the actuators to the control object reporting the status of the actuator, and from the control object to a single actuator (unicast) for
detailed (individual) control. Some information flow is envisioned also between (stacked) control-objects, one controlling and reporting to the other. All other information flow will be from tool/cloud to device and is not discussed here.

We will now discuss the message flow between the various OpenAIS objects following the above guidelines and approach.

### 3.3.6.2 Sensor Object message flow

Sensors will send their information to a (multicast) group of Control and DataCollect Objects. Sensors will send this message at a change in the sensed physical value. A typical message provides then "value + time" information, that allows the recipients to guess the time of possibly missing event messages.

Example: the messages for a push button:

- There are two physical events: press and release. The "press" event will usually not be transmitted itself immediately, as there is a timeout that separates a click from a hold event. The following messages are therefore foreseen
- Message when the button is only shortly pressed: "click" + time. This time information allows repetition of the click messages without possible misinterpretation.
- Message when the button is pressed longer than hold time: "hold" + time since the press (that's when the hold starts). This allows repetition of hold messages without the danger of misinterpretation.
- Message at a break from hold: "break" + time of the break. This allows reconstruction of the length of the hold period in repeated messages if the actual break event signal (break + time since=0) was lost.

The time since counter for repeated click or break messages should stop incrementing at a reasonable value that is equivalent to "long ago". The time since counter for the "hold" information (equivalent to press) should change the signal to "stuck contact" at a reasonable value. Using these considerations the event and the repeated messages are looking identical, but can be easily and simply separated at the control object using sequence numbers for the repeated messages.

Example: messages for a presence detector

- There are two states a standard presence detector provides: Presence and no presence. Messages indicating the state can therefore simply be repeated
- Although not directly needed we will also add time stamp and sequence for possible future use.

Example: messages for a Light Sensor

- Light sensors usually provide actual sensed values which can be send at regular intervals.
- Also here the "time since" is added as it may help to understand the strength of the trend, short time since values denote a rather quick change in sensed value.
- All sensors to send a message with a comprehensive status report (including the actual value, some statistics etc.) on request by a specific command.
Sensor Summary:

Sending (repeated) status value messages with a time and sequence number attached allows for repetition of messages in (randomized) intervals without additional measures to prevent possible double-takes by the control object. Possibly some additional information that adds to the understanding of the quality of the sensed values may be made systematically part of the value message. E.g.: Cell temperature, batt voltage, signal to noise, etc. This should be made part of the value message if it is needed to interpret the values correctly.

3.3.6.3 Actuator Object message flow

Actuators listen to commands from the control object(s) and perform the related action(s). With possibly many (hundreds of) Actuators receiving the same message multicast messaging is used between Control and Actuator Objects. We will not use related responding messages to the multicast, as these would easily accumulate in slow networks, and those that actually received the message are not the number one target for corrective measures anyhow. (The "call me if you didn't receive this message" doesn't work). The Actuator Object will therefore report its actual status at a regular (though randomized) interval, allowing the Control object to take corrective actions.

Control Object Messages:

Control Object command messages: These may be absolute settings (go to), relative settings (step), or referenced settings (scene recall). Some of these commands may be targeted to the group, some to the individual actuator. Note that the Control Object will repeat its commands only once by default, it will mainly rely on the Actuator status messages (see next bullet) to guarantee the reception of its commands and check the aliveness of the Actuator.

Actuators status message: Actuators report their status (achieved and requested output value(s), warning and error flags) independently from the control commands in (randomized) intervals, using unicast messaging. The control objects will compare the Actuator status to the expected status and send out a unicast corrective command if necessary. If a device has a good reason to expect that its actual setting is corrupt (e.g. after power up or recovery from exception etc.) it just issues a standard status message to the control object.

Actuators extended status message: Actuators can also send out an extended status message (including energy usage, operational performance issues etc.) on specific request only.

3.3.6.4 Control Object Message Flow

Control Objects derive the control action and target settings needed for the actuators from received (sensor) information. They are responsible for repeating the message to the Actuators once and after that only sets Actuator settings based on the state reported by the Actuator as we have seen in the previous sections. In this section we will look into the details of the Control command messages repeating.
Complications may arise when repeating relative or referencing command messages, as missing or double reception may lead to unwanted effects. (E.g. referencing commands may include timing sequences that break if triggered twice). Special care needs to be taken with continuous smooth changes using stepping commands, e.g. dimming used with the long press of a momentary action switch. If the start or the end command is missed the whole process fails. Therefore, repetitive stepping commands may provide the better and more consistent dimming performance, as only a small deviation from the desired outcome is happening on the drop of one of the signals. To achieve that the "start smooth stepping until stop" should be replaced by a series of "do a short smooth step" messages. A "consecutive step sequence number" can help the actuator to correct missing single short step commands.

Relative messages use the sequence number to allow for corrective action in the actuator if one message of the sequence was missing (e.g.: do a quicker stepping to recover from the missed one). Relative messages are never repeated themselves. Absolute, referenced and idle commands are repeated, with a repeat sequence number, allowing the separation of repeated from re-issued commands.

Please note that with non-relative commands the command needs to be executed whenever the sequence number is smaller or equal than the last received one. With relative commands the corrective action needs to be done whenever a sequence number is missing from the stream.

3.3.6.5 Illustration of dynamic behaviour

In this section a few sequence diagrams of a number of simple scenario’s will be shown to further clarify the operation of the architecture. Note that these diagrams already assume some setup and distribution of control. As OpenAIS supports variation in allocation of Control (see also Section 4.3) this already includes some choices.

The sequences shown are:
- Switching on light in a room, where also daylight regulation is enabled
- Presence detection in an open office, with a time-out

Switching on light in a room

We start with the simplest of lighting use cases, switching on the lights in a room. This is shown in Figure 15. It depicts a room with three luminaires of which one is close to the window. A central switch switches the lights to a predefined level ‘X’. Only the luminaire close to the window reacts to the daylight sensor and adjusts its value based on the daylight measurement.
Of course many variants of this use case including more sophisticated switches allowing multiple scenes (different levels per luminaire) can be derived from this easily.

### Presence detection in an office

The next example, shown in Figure 16 above describes a room where a presence detector is used to switch on light. In this office every luminaire is equipped with a presence detector. When a person is detected by the presence detector, the luminaire is switched on. Furthermore two neighbouring luminaires are always "slaved" and switched to a secondary level, creating a well-lit area around the employee. When the person leaves (or does not move at all for some time) after a certain timeout the system switches the lights off again. Of course the person could walk through the office and switch on other lights on the go.

![Figure 16: Sequence diagram of presence detection](image)

#### 3.3.7 Object Data Model

##### 3.3.7.1 Introduction

The functional decomposition presented above is only the first step in defining a full Object Data Model (ODM) for OpenAIS. The Object Data Model will not be described in this reference architecture document, but in view of its size in a separate document - D2.4 [OpenAIS_D2.4]. In the reference architecture we restrict our description to the structure and modelling approach.

For the OpenAIS project, an approach has been made that closely follows the lines of the LWM2M technical specification, as defined in [LWM2M]. Taking advantage of IETF’s CoAP protocol [RFC7252], LWM2M provides a flexible template for constrained device management.

The IPSO Alliance, based on the LWM2M template, has defined a set of Smart Object Guidelines which have been released in two separate documents, [IPSO14] and
Although these are not the set of ODMs that will be used in OpenAIS, they are a large collection of registered Objects that provides an example base.

To relate clearly to the LWM2M specification and IPSO’s documents, the terminology used in the OpenAIS definition has been aligned with the LWM2M standards. See table below for the terminology used in OpenAIS/LWM2M comparison to “standard” Object-Oriented terminology.

<table>
<thead>
<tr>
<th>Object-Oriented</th>
<th>LWM2M</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Object</td>
</tr>
<tr>
<td>class id/name</td>
<td>Object ID</td>
</tr>
<tr>
<td>object</td>
<td>Object Instance</td>
</tr>
<tr>
<td>method</td>
<td>operation / resource</td>
</tr>
<tr>
<td>method call parameters</td>
<td>Query parameters / payload</td>
</tr>
<tr>
<td>runtime environment</td>
<td>device</td>
</tr>
</tbody>
</table>

3.3.7.2 Tree and Inheritance

In the functional decomposition we have identified large blocks of functionality called functions like the Sensor and Actuator functions. Each of these will be represented in terms of object models as individual Objects which may expose several resources or operations. It will be also possible that multiple Object Instances of the same Object are deployed within a given OpenAIS device.

In object oriented sense for implementation purposes the following “inheritance” tree is suggested:

![Figure 17: OpenAIS Object tree](image)

Elements of this “tree structure” are:

- OpenAIS Object is an adaptation of the basic LWM2M object template that will be used for all OpenAIS functional Object descriptions. This ensures uniformity in interfaces and description throughout. This OpenAIS root Object contains all elements (resource/functionality) which are common to all OpenAIS application
Objects. Here we can think of general configuration settings for e.g. security or configuration.

- On top is a “base” Object for each application level function, like Actuator or Sensor. It defines all common behaviour to all within the same category, e.g. all Actuators or Control.
- At this level, specific LWM2M style Objects for control or other functionality pertaining to communication flow management and/or access rights management are defined. The Objects should be quite consistent across all OpenAIS compatible devices, independent of their operational functionality.
- The next level presents a step up into the level of detail, starting from the “base” Object, specifying standard Objects, e.g. a Sensor Object would be detailed for a motion sensor, light sensor, etc., as well as for specific Actuator Objects.
- Finally, vendors can implement their own object models, derived from OpenAIS basic Objects, and adding differentiating capabilities.

Main purpose of this hierarchy and the definition of basic behaviour is that it allows full interoperability of OpenAIS devices of various vendors on the basic functions, whilst still leaving sufficient space for manufacturers to stand out by providing extended functionality.

### 3.3.7.3 IPSO Smart Object Guidelines

There are already several ODMs defined by IPSO that can be used as example for OpenAIS. In the first public release, [IPSO2014], 18 objects were defined spanning from digital inputs and sensors into a range of actuators like light control or load control. To be noticed that Objects with “control” in their name are closer to actuator functionality rather than control functionality as considered by OpenAIS.

Recently, the “Expansion Pack” was released, including even more ODMs for sensors, some new actuators and control functionality. Furthermore, the second release also introduced a categorization of object models according to their type.

Each object that is defined by IPSO has an Object ID number that is registered at the Open Mobile Naming Authority (OMNA). The same is also valid for several resources that constitute the objects. OMNA this way enables interoperability through the registration and mediation of assigned numbers and names.

In OpenAIS it is not yet decided if the object and resource IDs will be registered with the OMNA. To keep the possibility open to register, following decisions were taken:

- Object IDs are taken from the ext-label range: 2048-10240
- Resource IDs are taken from the object specification range: 0-2047
- Reusable that could be published as reusable resources use an available ID from the range: 2048-32768
- Existing reusable resources are used where it is suitable

### 3.3.7.4 Structure and presentation of ODMs

Keeping in line with LWM2M specification, ODMs for OpenAIS will have a similar structure to those specified by IPSO with some complements that help to support aspects of the architecture. A description follows on how ODMs will be presented for OpenAIS.
Description
Each object model will start with a “Description” section which is a brief text introduction to the content and purpose of the Object. The main purpose for this section is readability for those implementing or interfacing the ODM.

Object Definition
A first table will provide the necessary fields for defining the Object (e.g. IPSO Digital Input Object):

<table>
<thead>
<tr>
<th>Name</th>
<th>Object ID</th>
<th>Instances</th>
<th>Mandatory</th>
<th>Instantiable</th>
<th>Object URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPSO Digital Input</td>
<td>3200</td>
<td>multiple</td>
<td>false</td>
<td>false</td>
<td>urn:oma:lwm2m:ext:3200</td>
</tr>
</tbody>
</table>

- Name: a reference name for how the Object can be identified.
- Object ID: a reference number for the Object which should follow the guidelines defined above.
- Instances: a descriptor field on whether or not multiple instances of the object in a given device are possible.
- Mandatory: a description if the Object is required to exist, at least once in the device.
- Instantiable: field added for OpenAIS which defines if a new instance of the Object can be created (e.g. a single switch device may not allow for multiple instances of the Object to be created)
- Object URN: the Uniform Resource Name for the Object.

Resource definitions
The collection of resources that make the Object are described in a second table (example from IPSO Digital Input Object):

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Access Type</th>
<th>Instances</th>
<th>Mandatory</th>
<th>Persistent</th>
<th>Type</th>
<th>Range or Enumeration</th>
<th>Default value</th>
<th>Reset to default on reset</th>
<th>Reset on reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5500</td>
<td>Digital Input State</td>
<td>R</td>
<td>single</td>
<td>false</td>
<td>true</td>
<td>Boolean</td>
<td></td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>The current state of a digital input</td>
</tr>
<tr>
<td>5505</td>
<td>Digital Input Counter</td>
<td>E</td>
<td>single</td>
<td>false</td>
<td>false</td>
<td>Opaque</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reset the Counter value</td>
</tr>
</tbody>
</table>

- ID: a reference number for the resource which should follow the guidelines defined above.
- Name: a reference name for how the resource can be identified.
- Access type: a descriptor field on whether the resource can be used for read and/or write operations or execute
- Mandatory: a description if the resource is required to exist, at least once in the device or if it is optional.
• Instances: indicates if the resource (can) be instantiated multiple times or not (single).
• Persistent: indication on how the value should be stored in either volatile or non-volatile memory. The current value of Persistent resources should be kept even through hardware reset processes.
• Type: data type used to store the resource value.
• Range or Enumeration: value interval or list of possible values that that the resource may have.
• Default Value: Value that the resource should have if a factory reset is performed.
• Reset to Default on Reset: indication flag if the resource should assume the factory value after a reset process.
• Units: units associated with the resource value
• Description: a short text description of the resource for readability.

Additional information on LWM2M ODM identifiers and resource model can be found in Section 6 of [LWM2M].

3.3.7.5 Data Formats
Data formats for LWM2M are defined in [LWM2M] Section 6.3 “Data Formats for Transferring Resource Information”. In addition to these formats, the OpenAIS architecture will also support the use of Concise Binary Object Representation (CBOR) which is defined in [RFC7049]. This format targets small message sizes, and allows for extensibility without the need of version negotiation.

3.3.8 Extension Mechanisms

3.3.8.1 Introduction
This section describes how an OpenAIS system can be extended in functionality and network size. Extendibility is one of the main goals. OpenAIS provides mechanisms that allow both the extension of functionality of devices and systems, as well as the extension of the coverage of the system including mechanisms to couple legacy systems. Much of this has already been described in the previous paragraphs, this section lists it all in one context for an easy overview of all extensibility mechanisms included in the architecture.

3.3.8.2 Extending the control behaviour
The SCA Architecture selected for OpenAIS as described in Section 0 supports the stacking of Control functions. That means that a new Control function with a higher functionality can be added to a system, and this new Control can extend the already existing functionality by “controlling the Control function”, without the need to replace it or take the existing one out. This way, new functionality can be added at any time to the system. There is no prescribed location for the Control functionality; with reliable and fast communication of the future (and the availability of full UDP IPv6 transport) it could well be even cloud-based.

3.3.8.3 Adding identical Object (instances) to the device
OpenAIS allows the use of multiple Object instances relating to one physical object, with a specific binding (e.g. with unique binding relation parameters) for each Object instance. This mechanism can be used to extend the system behaviour by configuration/commissioning only without updating the software on the devices. It can be used in a way that the basic behaviour of the already commissioned system is maintained.
Adding an Object instance (with its binding) for prepared Objects: the binding determines the way the signals flow. OpenAIS supports multiple Object instances for the same physical resource (both Sensor and Actuator), in this way enabling extension of the functionality of the system. Every Object instance comes with its own parameter set and its own bindings. Various extensions to the system functionality are made possible in this way.

OpenAIS encourages device manufacturers to equip their devices with such Objects that can, if needed, extend the functionality without software changes. Depending on the vendor such Object instances may be either provided as already-prepared Object instances that are “sleeping” on delivery, or they may be created dynamically as new instances of an Object that can be created and purged using the device management API. Object instances in use can be temporarily disabled to ease fault-finding missions and system behaviour analysis.

Examples:

- Add a sensor Object instance (with a new binding) to get frequent data for a trend analysis without changing the existing sensor interfacing from a light sensor:
  - The primary operational sensor Object instance was parametrized in a way that the sensor signal is delivered whenever a 20% change happens. This was made according to the requirements to allow persons to maintain the emotional connection to daylight.
  - Now a second, partly contradicting requirement comes in: For a detailed analysis the sensor should provide its value every 15 seconds.
  - To achieve this, a second Object instance is activated in the sensor that uses of course the same physical sensor hardware, but has a different parameter set that controls the data handling and also the binding. This second instance is set to deliver the value every 15 seconds, and to average the raw value only for 10 seconds. The benefit of proceeding this way is that the operational system that relies on the first Object instance and its configuration remains untouched.

- Add an Actuator Object instance to provide prioritized access to an actuator:
  - Due to the SCA structure of OpenAIS an “only one Control function sets the actuation” setup will be sufficient in most cases and allows for a straight-forward and well working structure.
  - However, there are situations where more than one Control function should be granted access to an actuator function. Imagine an outside blind actuator that is operationally controlled by a Control function.
  - This standard operational Control needs to be overridden if some weather emergency (heavy wind, ice, solar overheat etc.) happens. And the whole setup has a master override that is operated by the fire brigade to open (or close) the blinds to support the evacuation of persons or the fight against fire best.
  - This prioritizing is to be handled by the actuator devices directly, as emergency signalling may not be run through (possibly changing) operational Control functions.

OpenAIS can handle this by using additional actuator Object instance, with each Object instance having its own set of parameters (including the priority). This way the higher-priority signal prevails to finally set the actuator state. As the other Object
instances remain operational, the operational change requests will still be taken, but the execution is postponed until the higher priority Control is withdrawn.

3.3.8.4 Modular software upgrade and plugin modules

Software upgrade allows for full freedom of extending the device functionality, of course. OpenAIS encourages device manufacturers to add the possibility for (trusted) third parties to add plugins or change single modules of device software.

- Module and plugin handling remains system specific, and due to the diversity and the ongoing rapid development of (embedded device) operating systems OpenAIS will not mandate a platform or language for plugins and modules. However, it encourages the device manufacturers to enhance their offer, and recommends it for future standardization, quality check and neutral hosting regarding some more wide spread OS solutions, once established.
- Scripting plugins that may be used as a plugin type of extension suffer from the same issue: scripting solutions for embedded low-resource devices need to match the abilities of the firmware, and therefore cannot be standardized at this moment in time. OpenAIS encourages vendors to go for scripting plugin structures that serve as (trusted) plugins to better cope with the high diversity of lighting applications and system functionality demands over the lifetime of the devices.

System extension by adding new or renewed Objects

Additional or renewed Object (classes) may be added to the device, providing additional features. Please note that such Objects may also be created in a way that alternative communication properties are used, to support specific IoT data protocols or even future competing communication protocols like KNX/IoT, BACnet/IoT etc. Due to the foreign signal handling in a parallel executing object and the flexible mechanisms used to provide it, such additional protocol integration can be achieved without conflicting to the OpenAIS architecture and the continuing functionality of the already commissioned system. Some more simple special services may be created using plugin-type of modules, more complex services will need a module upgrade of the software.

Functionality update / upgrade by new hardware control software

The modular structure, that allows for a standardized interfacing between the Objects and the hardware driving software ("driver"), is also the basis of updates to the driver software that may provide enhanced or additional features. As long as the original API towards the Objects is maintained, such modules may add to the functionality easily without jeopardizing the already commissioned and well working system.

Example: For indoor positioning a renewed driver (hardware control software) is used which modulates the indoor coordinates into the light output using an HF modulation. To achieve this inside an existing system new hardware driver software is needed, that maintains the interface to the existing Objects. Doing so allows for a continued use of the already well working system.

3.3.8.5 Foreign / heritage / legacy systems

OpenAIS is designed to provide integration of heritage/legacy systems. Heritage and non-OpenAIS systems are integrated into an OpenAIS system by using Gateways. Such Gateways use application layer protocol translation to achieve the integration. Integration will vary from “full integration” where the gateway functionality covers almost all aspects of OpenAIS (by using a kind of middleware functionality, that is able
to translate any request in the view of status information on the heritage system) down to "simple minimum coupling" that only covers some simple operational and directly translated communication, but has no provisions for commissioning, maintenance or device management communications available.

3.3.9 General Mechanisms

3.3.9.1 Introduction
There are aspects of the architecture which cannot be allocated to a single Object in the decomposition as these are the result of the combined properties of all Objects. Typical examples of such aspects are e.g. performance, start-up behaviour etc.

Most of these will not be dealt with at the reference architecture level but in the system design of instantiations of the architecture. However, in this document we will discuss error handling and define its basic requirements in this paragraph as error handling concepts are seen an essential property of the architecture. The more networking and implementation focussed aspects will be addressed in the networking view.

3.3.9.2 Error handling

Error handling Concepts
As software and systems are never error free and hardware and communication malfunctions happen, error handling is an essential property of all software intensive systems. Lighting Networks are especially inherently lossy, which is a primary source of system errors. Incidentally messages will be missed by one or more Objects. This will happen without any knowledge of the intended Objects. On top of that common hardware and software faults will occur at random intervals. This introduction describes the most important aspects and concepts for the OpenAIS error handling and recovery mechanisms. For a description of the main requirements for this mechanism please see Section 3.2.

Error handling:

- Containment: One of the two primary requirements for Error handling is containment. Whenever an exception occurs, its handling and effects should be restricted to the entity in which it occurs. So a reset or hang-up in one device should never result in resetting any communicating devices. Proliferation of errors should be avoided at all times.
- Final known state: Error handling should always lead to a controlled transition to a known (defined) state. Note that this state may still be "non-operational" when no recovery is possible like a broken device. Undefined states must always be avoided.

Graceful degradation:

- Malfunction of an Object should leave all functionality not involving that Object operational. Any group behaviour in which the component participated should continue with the exception of only the broken Object. Examples: group behaviour should continue if one of the Actuators fails, malfunction of a Sensor in a luminaire should not impact the operation of the Actuator at that light point.
- System designers may choose to add additional graceful degradation measures, not mandated in OpenAIS. This may even be a differentiator, therefore only a few examples are given here:
Compensate (to a certain extend) the effects of the broken component on the functionality by controlling the remaining components in an alternative way. E.g. when one light point is broken the surrounding lights may be put to a higher value. When a sensor is broken assume another default behaviour.

- Redundancy by duplication of components beyond what is needed to realize the functionality so that a broken component can be fully compensated.

- Local buffering in case of DataCollect functions, which may be used to store a period of non-connectivity to the cloud.

**Recovery:**

- All devices that transition to an error state should attempt to recover to normal operation. However, we should avoid an endlessly repeating process of state transitions and retries (like a luminaire that goes off and on repeatedly until it is replaced). Therefore, the number of subsequent resets must always be limited.

- Inconsistencies in overall system state introduced by (temporary) unavailability of devices or connections, or communication failures must repair automatically within a (to be configured) time. A missed or lost message or temporary absence of a device (e.g. when it recovers) should not lead to persistent deviations in system state like a luminaire that never turns on.

**Security:**

- Lighting networks should show similar messaging patterns in on and off-state. This is done to protect presence privacy against simple pattern sniffing.

**General:**

- Structures and rules to achieve detection and repair of missing messages should be as simple as possible, unexpected special states of the complete system should not happen.

**Error handling mechanisms**

There are still many ways to include these concepts in the architecture and design of OpenAIS compliant systems. This Section describes the chosen approach, where we start the description with error detection.

**Detection of error situations:**

- Many errors are detected automatically as calls return error codes, or communications fail. These do not need special mechanism therefore to be detected. Please note that all known errors should be handled in the software and never lead to unhandled exceptions.

- For communication links that are not used frequently and as such may fail “silently” another situation exists. Also the receiver of an event can never detect if the event is not coming due to a functional reason or a crash of the sender. In these situations we will make use of a “liveness check” to detect errors.
Note however that not all links are equally important. As discussed in the previous paragraph an OpenAIS system has a specific message flow between the elements. If we have a typical set-up we see the following links:

- Sensors – Control: the sensors are not interested in the absence of the control object as they cannot do anything about this. However, the Control Objects are interested in the absence of the Sensor Object as they might adapt behaviour.
- Control – Actuator: The Actuator is not interested in the state of the Control Object, however the Control Object is interested in the reverse as it might implement fall-back mechanism if one or more luminaires fail.
- Control-Control – Obviously the Control Object is interested if its superior Control Object is still alive as it may implement fall-back mechanisms when this is not the case anymore.

So we see that the functional interest is mainly from the Control Object towards its Sensors, Actuators and the (stacked) Control Objects it commands. Furthermore, a Control Object is interested in the presence of its “Superior” Control Objects.

To cope with these silently failing objects we need a mechanism to periodically check if one of the interesting links/counterparts is still there. For this purpose we will use whenever possible the standard operational messages exchanged between the entities to check the state of “upstream” and “downstream” counterparts. This will be achieved by implementing a “repetition” mechanism so that current states or commands are repeated at a predefined frequency.

**Recovery from error situation:**

- Whenever a communication fails, a message will be repeated a few times. Note that to avoid communication peaks the retry interval should be randomized with some median being configured.
- When serious software errors occur that leave a device potentially in an undefined state, always a reset must be executed to bring a device back to a “known” and “stable” state.
- All calls between Object instances will be based upon the state of the caller only. No assumptions will be made on the state of the called Object. Also transactions (i.e. related calls) will not be used.

For missed communication (not unlikely in multicast) or resetting of devices the following mechanisms will be used:

- Sensors re-evaluate their status and resend their settings to (Controls) at regular intervals. These intervals are configurable (and will be randomized) to tune the communication load. Absence of this repeated communication is hence also an indication of a missing sensor.
- Actuators will regularly report what their state is. When a (Control) Object notices e.g. that one of the Actuators is not in the desired state it could resend the desired state.

Note 1: What state the actuators will report depends on their properties obviously. A simple light point that can only switch on/off can only report that, dimmable colour lights will have a much more extensible status describing their current settings.
Note 2: Not all sensors may have a “state”. If we have a toggle switch which just changes the state to the reverse the switch itself does not have a state and just repeating the last send message (a click in this case) without further information would lead to erroneous behaviour. In these cases we will work with timestamps and sequence numbers to indicate if this is a repeated message or not.

Graceful degradation:

- When the absence of an Object instance is detected (through the mechanisms described above), dependent entities should revert to “default” behavior. This behavior is not prescribed by OpenAIS and default there is none, however we can give some examples:

- A Control Object that finds out its sensors are no longer available can switch to a “default” safe state (e.g. lighting at 50%, to be configured).
- Also in the case of stacked Control the “overridden” Control Object instance may resume its normal duty when the “master” is found missing. Note that the “watchdog” will continue and that when an Object instance returns, the Object instance should resume normal operation.

3.3.9.3 Time synchronization

Time is an essential element of a number of operations in a lighting system. The most important requirements concerning time in the context of this architecture document are listed in Section 3.2:

OpenAIS will support mechanisms for keeping/creating a common time base between OpenAIS nodes. It is expected that all devices on which Control functions are deployed will have a Real Time Clock (RTC). This RTC will have to be synchronized at start-up and at regular intervals. For this purpose and because not every embedded device is expected to have an RTC, specific network-based time synchronisation mechanisms are needed. These depend on the PHY/MAC technology chosen, and will not be defined in detail by the OpenAIS consortium. Below the current options considered are listed:

- Ethernet/Wi-Fi high-speed networks may use the Network Time Protocol (NTP).
- 6TiSCH wireless mesh networks may use the Absolute Slot Number (ASN) provided by the stack (which is already needed to do time-synchronized scheduled communication and provides better than 1 ms synchronicity) and pass this up to the application level. The Border Router can use NTP time and feed it into the mesh network as the relation between the wall clock time and the ASN, so that all nodes have accurate wall clock time information.
- For Thread wireless mesh networks a suitable method still needs to be defined. It may be based on multicasting time information from an NTP-enabled Border Router onto all the nodes of the mesh subnet.

3.3.10 Out-of-the-box design

To achieve the “out-of-the-box” across vendors, a specific system layout is necessary. The layout and the way it operates are given below, whereas the protocol and API details within the framework are given in the respective sections:

- All sensors are preconfigured to multicast their sensor values on any change to all Control functions that are preconfigured to listen to those sensor events.
All manual sensors (toggle switches, dim switches, etc.) are preconfigured to send the new sensor value event on any (substantial) change.

All light sensors are preconfigured to send the new sensor value event on any substantial change.

All presence detectors are preconfigured to send the PD_ON event with any newly detected presence and a PD_Hold event every 30 sec with continued presence. A PD_OFF event may be sent, but is not mandatory.

- All light points are preconfigured with a small internal Control function that is controlling the local light point actuation directly (without external communication).
  - This Control function is preconfigured so that it listens to the sensor events.
  - The Control function is preconfigured to execute the events created by manual sensors immediately.
  - The Control function is preconfigured to switch on the lights if the lights are off if a PD_ON event is received.
  - The Control function is preconfigured to slowly dim the lights to off after reception of a PD_ON whenever for a period of 60 seconds no PD-Hold event is received.
  - The Control function is preconfigured to react on sudden substantial changes to light sensor values (e.g. changes that come from manual covering or uncovering a light sensor) with a 50% increase when light sensor readings drop, and a 50% decrease when light sensor readings jump back up again.
  - The Control function is preconfigured to set the light output on the reception of a scene recall event to the value (105 - scene number*5) % of light output (but never to off).

Note that the security of the out-of-the-box operation is very low, as the keys for this operation are widely distributed and they will not be changed to maintain this functionality for all devices in the same way. OpenAIS devices get secured only by performing the commissioning process. After commissioning all out-of-the-box interaction will be totally disabled.

### 3.4 Physical View

#### 3.4.1 Introduction

The physical view in architecture describes the mapping of the abstract logical functions identified above upon real software and hardware components. This includes technology and actual design choices for components. However, as we are describing only a reference architecture in this document we have to take a more flexible approach in this section. As we are defining a reference architecture to be used by many companies over a lengthy period of time, we cannot specify all choices in this document as we would do in a real system design. Therefore, we will only define a minimal number of fundamental requirements that apply to all members of the architecture.

The topics covered in this section are:
- Types of devices identified in the OpenAIS architecture
• Some fundamental choices for communication infrastructure
• Physical structure of the system

In Section 3.5 we will go one step further and make some more choices and recommendations on technology and component selection which we think apply for the immediate future.

3.4.2 Physical devices

In the architecture we identify the following types of physical devices:

• Luminaire: A device with one or more light points (e.g. LED drivers) and zero or more sensor devices connected to it.
• Standalone Sensor: A device with only one or more sensors connected to it. It may be either battery- or AC-powered.
• Power-harvesting device, e.g. a standalone sensor or button.
• Area (floor, building) controller: A device with only networking capabilities, no drivers nor sensors are connected. Naming of these devices normally follows their control scope.
• Routers and switches which form the basic network and IT-infrastructure

As the routers are expected to be COTS-products we will mainly focus the architecture description on the lighting specific devices only. The system architecture assumes that routing and switching can be covered with standard elements and that the actual physical design of that network is not necessarily part of the Lighting system development, although in practice some minimal requirements will have to be defined.

There is one exception to the above rule and that is the so-called “Low power Access Point”, an OpenAIS element that consists of a 6LoWPAN radio and a host running a Linux-like operating system. This access point is a border router that routes traffic between wireless and wired network interfaces. It also takes care of conversion between IPv4 and IPv6 domains if necessary in an OpenAIS system.

It can also serve as a generic contact point within the lighting system that can be used to connect “non-lighting” devices. As long as 6LoWPAN connectivity is not ubiquitous in buildings the Lighting network may be used to connect other devices outside the lighting domain that need this type of connectivity.

3.4.3 Networking choices

In OpenAIS a choice is made for IPv6 communication, supported to every end-node. This IPv6 interface will be one of the main decoupling points in the architecture around which other choices will be made. Note that the expectation is that given the advance of the IoT, IPv6 will be commonly available at the moment of introduction of OpenAIS based systems in the market (2020).

OpenAIS will not mandate certain physical interfaces as long as they can support IPv6 communication. OpenAIS will support wired and wireless interfaces and in the networking view (3.5) default choices for wired and wireless solutions as well as minimum capability of PHYs will be defined.
### 3.4.4 Physical structure of a system

The physical structure of an OpenAIS system is expected to have a physical layout as shown in Figure 18. See Section 9.3 for the list of icons and the components they represent.

![Figure 18: Physical layout of an OpenAIS system](image)

An OpenAIS system consists of clusters of luminaires and/or standalone sensors which are all connected to a local network, also called Field Network. This may be a wired or a wireless network. Within such a local network it is expected that all devices will use the same network technology. Note that these local networks may not be fully separated geographically. For example, in one open office there may be a wired network for the (PoE) luminaires and at the same time a wireless network for the battery operated presence sensors. The local network is connected to the rest of the network through a (border) router e.g. one that connects a wireless 6LoWPAN network to the wired Ethernet domain. Depending on the physical technology chosen the amount of nodes on such clusters must be properly dimensioned for performance. Each cluster (which may be an entire floor) may be associated with a physical (Area, Floor) controller. A single controller may serve multiple clusters. Note that such a stand-alone controller is an optional element in any set-up; the smallest configurations operate without any stand-alone controller. The coordinating Control function can be allocated to one of the luminaires. Also in larger configurations the Control functions may be allocated on other IT-devices like servers and not necessarily to a dedicated lighting area/room/floor controller. It is expected that controllers, if present, will be integrated in the wired IP-domain.

Within clusters gateways may also be present. They translate legacy and non-OpenAIS devices to OpenAIS interfaces. In a gateway these devices are represented as OpenAIS devices towards the rest of the network.

### 3.4.5 Device Classes

Computing capabilities are a very important feature for the devices, as they determine both the capabilities as well as the cost level. There will be three (performance) classes of physical devices identified in the architecture.
In the list below we also indicate the “working assumption” performance levels defined for those in the 2018 timeframe. It will be clear that over the lifetime of this architecture (20 years) there will be major developments in silicon technology which will probably (drastically) change the performance levels available to implementers of the architecture. However, we define here an assumed introduction level for initial development of the system.

The device classes are:

- **Low resource devices:**
  - Limited execution environment, low performance CPU, no or limited OS.

- **Medium resource devices:**
  - Standard embedded device with on chip flash and RAM, capable of running a real time kernel.

- **Full resource devices:**
  - State of the art CPU, external RAM and flash equipped with a “standard” operating system like Linux.

The OpenAIS architecture does not define a specific embedded device platform, nor a specific Operating System (OS) or Real-Time OS (RTOS). The architecture allows free choice of vendors for their preferred platform/OS, and stimulates both innovation and vendor competition in the domain of embedded platforms, and their processes and tooling. To ensure a smooth operation of OpenAIS on an embedded OS/RTOS, a set of minimum requirements towards a platform and OS/RTOS is provided in section 4.1.3.2.

The low resource devices may also be sleepy devices, most likely sensors and battery operated, that are operational only intermittently to save power. This means that these devices will not be able to respond continuously to requests. For the operation of such devices a proxy will be installed on an intermediate device like the low power access point that buffers requests to the device.

Additionally, to these standard devices there may be restricted resource devices that can operate without even battery or power. These devices will most likely not be able to integrate the complete OpenAIS network stack. Therefore, they will be vendor specific and can be integrated into the OpenAIS architecture by using a nearby device, likely a luminaire, as an intermediate “converter” or gateway to the OpenAIS architecture.

### 3.4.6 Device Management

#### 3.4.6.1 Introduction, functions

Device management (of devices in the field) comprises of the following functions:

- Keeping track of the static “status” of devices, such as their ID’s and the version(s) of software and hardware. This also includes any configuration settings of the device. A copy of that setting is maintained at the server to be able to recover if it is lost in the field.
- Managing software updates (checking compatibility of patches and updates with hardware, mandatory updates, e.g. also end of support).
- Managing configuration settings, that includes also all settings that are made during commissioning. In the next section we will present an overview of these settings. In these configuration settings, the list of the installed Object instances
in the device (types, ID's) describes the available OpenAIS functions for this device.

- Keeping track of the “dynamic” status of the device, especially errors (types, frequency of occurrence like number of reboots), wear out indicators like burning hours, (self) test results like memory errors. In short, anything which gives an indication of the “health” of the product (and the need to repair/replace it) or even do preventive actions.
- Any remote service checks/tests or calibrations if they exist.

The extensive sets of data coming from devices like sensors are not counted under device management but as normal application data. Those data are stored in databases for off-line data-analytics.

### 3.4.6.2 Configuration parameters

This section lists the parameters that should be available in OpenAIS devices. In the end these parameters will be modelled in a “device” Object. All the resources shown below will be accessible through the “device management interface”.

#### Device Information:

- **Static information**
  - The Device ID (The Unique MAC provided by the OEM device manufacturer)
  - The OEM product ID (The EAN identifying the actual product)
  - The OEM product string (readable text)
  - The final product ID (The EAN identifying the unit the OEM device is used in)
  - The final product string (readable text)
  - The location reference ID (e.g. grid or position vector)
  - The location reference (readable text)

- **Static operational information**
  - The list and structure of the Object instances the device has built in
  - Communication Interface Addresses:
    - The IPv6 address
    - The IPv6 multicast addresses the device listens to

- **Dynamic information**
  - Time (and date)
  - Device status
  - Device temperature
  - Network interface status
  - Network interface statistics
  - Device power usage (actual power drawn)
  - Device energy usage (incrementing counter)
  - Actual power supply status (battery status etc.)

### 3.4.6.3 Function parameters

On an OpenAIS device a number of functions (e.g. Sensors - switches and presence detectors, Actuators - light points and Controls) may have been deployed. Common parameters for functions (or precisely all Objects within these functions) are:

- **Type / Class of the deployed Objects**
- **Instance identifier(s)**
- **Multicast address / Group binding identifier(s) (to support the broadcast mechanisms)**
• The operational status of the Object instances
• Enabled / disabled (method to disable an Object and show it)
• Operational & failure code

Specific to Sensors Objects:
• Sensor firmware specifics (firmware characteristics)
• General sensor settings (possibly firmware related, e.g. basic sensitivity/attenuation etc.)
• Filter Parameters (low/high pass filters, thresholds, heartbeat intervals) for the sensor event generation
• The actual sensor value and the time stamp when the sensor value has been valid
• Operational status of the sensing device (preheat, operation, sleep etc.) and actual temperature of sensing device

Specific for actuators: (here: Light points)
• Actuator hardware specifics (min./max. output, max. allowed temp, max lumens, light point spectrum information, etc.)
• Priority schedule (needed if two actuators' logic operate the same hardware)
• Operational Information: the actual actuator output, the final /requested actuator output,
  o The time the final actuator output will be achieved (used with slow dimming)
  o The scenario values to be used for the scenario calls
  o The additional scenario parameters (e.g. way of colour change, slope settings, etc.)
  o The actual “dim-step” control (slope, width, etc.)
  o Light point power and energy use
• Operational status: actual temperature of the device,
• Total lumen/hours (%hours) output (this is the wear- out parameter for the LEDs)

Specific for Control:
• Sensor list / per sensor
  o Addresses (including the binding information to the sensor broadcast group)
  o Type/class of the sensor
  o ID of the Sensor
• Actuator list / per actuator
  o Addresses (including broadcast information)
  o Type/class of actuator
  o ID of the actuator
• Control - stacking specific information
  o Override setting (none, heartbeat controlled, full)
  o Last received heartbeat (timestamp)
  o Fallback interval (time after heartbeat the Control function takes active action again)
• Control specific information
  o Max update interval (time before the Control function re-evaluates the settings)
3.5 Networking View

This section presents the networking view onto the OpenAIS system architecture.

3.5.1 Overview

3.5.1.1 Communication stack

The architecture uses IPv6 communication for all devices and for all purposes, up to the lighting end-nodes in the Field Network. The IPv6 interface is one of the main points of technology decoupling in the OpenAIS architecture.

Around the basic choices for IPv6 and the support for both wireless and wired connections, a complete network stack has been selected for OpenAIS. This includes support for both wired (default selected: Ethernet) and wireless (default selected: 6LoWPAN/Thread). In this way both resource-restricted and unconstrained networking is supported. Note that the indicated default selections are just defaults, any PHY/MAC technology that supports IPv6 can be selected as long as it complies with the minimum requirements for OpenAIS specified in Sections 3.5.5 and 3.5.12. The envisaged network stack is depicted in the Figure 19.

UDP will be used for transport. The main rationale for this choice is its use in the constrained domain where UDP is currently used in conjunction with the CoAP protocol. Although there are some problems known with UDP acceptance by firewalls for security reasons, the expectation is that this will not be blocking for this architecture.

On top of the IPv6/UDP layer, this architecture will make use of CoAP (including CoAP multicast – see orange boxes in L7) for all communication between functions and other modules. For unicast transport layer security we will make use of DTLS. All interfacing
between the applications like between Control and Actuators will be through RESTful web service interfaces.

We will use OMA Lightweight M2M (LWM2M) on top of CoAP as a framework for building the RESTful interfaces on. LWM2M is an efficient, emerging IoT framework that enables both device management and application data communication. It supports bootstrapping, (DTLS) security, registration and Object/resource access.

Finally, OpenAIS will develop a data model (blue box in L8) which the applications will use. This model was introduced in Section 3.3.7.

3.5.1.2 Extensions needed on top of the selected standards

Current standards and tools do not yet fulfil all requirements for the OpenAIS architecture. Especially the LWM2M standard is focussed on one to one device to (back-end) server communication, which we call the Device-to-Cloud communication pattern in this document. For lighting applications, local (Peer-to-Peer) communication and multicast communication (also called Peer-to-Peer in this document) is required. Also the LWM2M operational model is based on a permanent connection of all field devices to a server in the cloud, a situation that cannot always be guaranteed in a lighting system deployment.

In Section 3.5.2 the required extensions are described in detail.

3.5.2 Open Mobile Alliance Lightweight M2M Standard (LWM2M)

3.5.2.1 Overview

During the state-of-the-art research phase of OpenAIS, various existing and upcoming IoT standards offering lightweight IP-based management and control were investigated within WP2. One of these standards is Lightweight M2M [LWM2M] by the Open Mobile Alliance (OMA), abbreviated 'LWM2M'. The public Version 1.0 of the LWM2M standard has been chosen as the basis for the OpenAIS networking solution.

LWM2M is a framework for device management and service enablement that defines the LWM2M protocol between a LWM2M Client (i.e. a resource-constrained M2M field device) and a LWM2M Server (which is the configuring entity and typically not resource-constrained). The protocol was developed for use over cellular M2M connections with potentially very low bandwidth and a non-negligible communication cost per kilobyte, but the standard is also applicable if used over any other IP-based network technologies. (Note: The LWM2M naming of “Server” and “Client” differs from the usual interpretation where the initiating entity is called “client” and the responding/queried entity called the “server”)

The protocol is optimized for constrained field devices, that is, devices with limited RAM, limited flash memory, low-bandwidth connectivity and/or limited CPU resources. LWM2M aims to minimize the size of data packets while still being generic and extendable. LWM2M uses the Constrained Application Protocol CoAP (detailed in the next section) as a transport mechanism. It also supports data payloads in multiple data format standards: industry-standard JSON, compact TLV and plain text/numbers.

A key advantage of LWM2M compared to other IoT initiatives is that it has a released specification and implementations already on the market, including open source implementations such as those by the Eclipse Foundation.
A key disadvantage of LWM2M is that not all functions required for the lighting and building control market are supported in version 1.0 of LWM2M. It is expected though that a future revision of LWM2M will address these needs.

### 3.5.2.2 Proposed additions to LWM2M

Specifically, OpenAIS has identified the following functions that have to be added to LWM2M 1.0 in order to satisfy the lighting and building IoT requirements:

1. Peer-to-Peer secure unicast communication; the ability of a constrained field device to directly talk to another field device securely without going through a (cloud) server or other back-end infrastructure.
2. Peer-to-Peer secured and unsecured group communication over IPv6 multicast (see Section 3.5.6).
3. Support for event notification based on pre-configured bindings, including event distribution to a group of subscribers (see Section 3.5.7).
4. Local discovery of services and devices (see Section 3.5.8); to enable optimal commissioning tooling, and configuration/operation of field networks without IP backbone connectivity.
5. Allowing out-of-the-box operation functions, which are operational already before an OpenAIS device is commissioned (see Section 3.2.5).
6. Role-based authorization (see Section 3.6.4); required for a proper and manageable lighting system security.
7. Addition of the compact data format CBOR [RFC7049], which is a binary encoding of Objects similar in structure to JSON. Within OMA, the addition of CBOR as a more compact alternative to JSON has already been discussed.

### 3.5.2.3 Proposed modifications to LWM2M

Also OpenAIS have identified the following functions that need to be modified in LWM2M 1.0 in order to satisfy the OpenAIS requirements:

1. Device bootstrap needs to be supported without relying on a pre-configured Bootstrap Server configuration in each device (see Section 3.6.3).
2. The LWM2M resource model limitation of maximum three levels deep URI paths - counted from the root resource - needs to be extended; to at least five levels (see Section 3.5.6.3).

### 3.5.3 Constrained Application Protocol (CoAP)

Communication messages in OpenAIS are carried by the Constrained Application Protocol (CoAP), a protocol defined by the Internet Engineering Task Force (IETF) as a replacement of HTTP suitable for low-bandwidth, constrained Internet of Things devices and networks. CoAP is a request/response protocol with the same semantics as HTTP but encoded much more efficiently. CoAP is defined to operate on top of the UDP protocol currently; IETF is defining an extension to operate CoAP also on top of the TCP protocol.

The CoAP specification is [RFC7252]. The way in which CoAP is used in OpenAIS is mostly defined in the OMA LWM2M [LWM2M] specification.

Because the OpenAIS architecture adds various functions to the basic LWM2M standard, additional ways of using CoAP need to be defined. This section describes these additions and any differences with respect to the LWM2M 1.0 usage of CoAP.
3.5.3.1 CoAP Blockwise Transfer

CoAP Blockwise Transfer [CoAP-block] is an optional feature of the CoAP protocol to transfer large data blocks in parts. CoAP Blockwise Transfer is not used in OpenAIS for any of the functions defined in the OpenAIS project so far. It MAY be used in vendor-specific functions such as new firmware download, or with vendor-specific CoAP resources, or in upcoming OpenAIS Object Model definitions where large data payloads need to be transported (e.g. with debugging logs).

3.5.3.2 CoAP Observe

CoAP Observe [RFC7641] is an optional feature of the CoAP protocol to register an observing CoAP client to be notified of changes that occur to a specific resource on a CoAP server. It can be used to enable temporary observation of specific resources, initiated by the interested receiver of events.

An observe relation does not persist across device reboots at the CoAP server side, i.e. if the device being observed resets, the observer needs to resend the observe request after detecting that the expected notifications do not arrive anymore. Furthermore, CoAP Observe is limited to unicast notifications only. These two properties make the observe/notify mechanism different from binding which is discussed in Section 3.5.7.

In the LWM2M standard, CoAP Observe support is mandatory and used to implement the “Observe” operation. Hence all OpenAIS devices will support CoAP Observe on the CoAP server side in the way specified by [LWM2M]. Please note that the CoAP server that holds the resource is called “LWM2M Client” in the LWM2M naming.

3.5.3.3 CoAP Multicast

For sending out events, commands or other requests to a group of nodes, or for performing discovery, a CoAP request can be sent out as an IPv6 multicast as specified in [RFC7252]. Guidance on a number of specific topics for CoAP multicast is provided in [RFC7390].

A CoAP server capable of receiving multicast requests MUST support the CoAP No-Response option [CoAP-no-resp] which allows a CoAP client to control which responses it wants to receive or suppress. A CoAP server inside an OpenAIS device has per-resource default multicast response behaviour that is defined by the OpenAIS Object Data Model. Note that multicast requests are only accepted on specific resources that support it. Multicast requests to resources that do not support multicast will simply be ignored by the receiver.

CoAP multicast messages are secured at application layer as defined in Section 3.6.5.3, using the COSE data format specified within IETF.

3.5.4 Low Power Radio Access Point

3.5.4.1 Introduction

A (wireless) access point is the device that wireless devices will connect to, over the wireless channel. It is a control point for deciding which nodes to allow on the wireless network, and it is an IP router that routes IP packets between the wired and wireless network segments. In OpenAIS, the Low Power Radio Access Point (LPR AP) is defined specifically as a wireless access point that provides access to a wireless IPv6 network that uses a low-power wireless protocol such as 6LoWPAN or Thread.
As a router the LPR AP is a piece of network infrastructure, invisible to applications. As a control point, it uses well-standardised mechanisms (often built into the wireless medium access layer specification) to let connecting devices discover it and request access, and to let its operators set policies about which access requests to accept, and to execute authorisation decisions against those policies.

If needed, a low power access point can also host proxies for sleepy devices. (Note that other devices like luminaires can also host such proxies, albeit with less memory available typically for performing proxying functions.)

### 3.5.4.2 Architecture

![Figure 20: Low Power Radio Access Point Architecture](image)

Figure 20 depicts the generic architecture proposed for the Low Power Radio Access Point (LPR AP). Its architecture is similar to a Wi-Fi access point that provides wireless connectivity to Wi-Fi devices. As shown in Figure 20, it consists of two components: a mesh border router and a Linux host interfaced using either SLIP over UART/USB or Ethernet. The Border Router is an edge node in mesh network, which is responsible for creating, maintaining and providing external connectivity to that network. The Linux host provides additional features expected from an access point like VLAN support, authentication services, remote logging, remote management and mesh network management.

OpenAIS does not mandate specific hardware for an access point; instead it defines a minimum set of requirements for hardware to ensure proper functioning of the OpenAIS network. The chosen hardware for the Border Router component should be at least a medium resource class device and the Linux host should be a full resource class device (see Section 3.4.5). In addition to this, the Border Router PHY/MAC should be capable of supporting the minimum requirements defined for Network joining (see next section).
In an OpenAIS wireless network, the LPR AP is the first node that is responsible for creating the network. It allows other wireless OpenAIS devices (e.g. sensors, actuators and luminaires) to securely join the network through a network joining and commissioning process. Wired OpenAIS devices will be connected to the LPR AP through switches/routers available in the backbone network and they are added to an OpenAIS network through the commissioning process.

One of the most important goals for the LPR AP is to enable easy integration of an OpenAIS lighting system into existing IT infrastructure. In order to achieve this, the LPR AP should provide the features expected by IT administrators and also adhere to defined network policies. Below is the list of features expected by IT administrators:

- **VLAN support**: Most IT networks use VLAN (IEEE 802.1q) in order to segregate the network traffic.
- **Authentication services**: In order to authenticate the devices and users of an IT network, strong security measures like site-wide authentication using LDAP is required.
- **Remote logging**: Monitoring the activities in the network is very much essential to identify rogue devices, e.g. flooding the network, and to take necessary actions.
- **Remote management**: A typical IT infrastructure will consist of hundreds of access points. Remote management allows configuration/reconfiguration of those devices from a central server.
- **Mesh network management**: It should be possible for an IT administrator to reconfigure mesh network parameters like radio channel and PAN ID in order to avoid interference with other wireless networks.

Most of these IT requirements are readily fulfilled by standard components in the Linux operating system or can be built on top of existing Linux functionality. Instead of re-implementing functions on our OpenAIS Border Router platform, a Linux host is used in our LPR AP design that meets the necessary IT requirements. The Border Router and Linux host are connected using either SLIP over UART/USB or Ethernet and standard IPv6 routing is used to route packets between them.

### 3.5.5 Network Joining

The Network Joining function is what allows new OpenAIS devices to join others over a network. Because OpenAIS is independent from specific PHY/MAC/IP-stack choices, we do not describe this function in detail but instead we pose a set of requirements towards the PHY/MAC/IP-stack technology to ensure that OpenAIS network joining is easy, reliable and secure.

Summarizing, the OpenAIS architecture aims for a network joining function that:

- Is independent of the specific MAC/PHY technology used and the specific IPv6 network stack (implementation) that is used;
- Should work for both wired and wireless IPv6 subnets, even when these are combined in a single system;
- Integrates well with the specific network joining functions used by common MAC/PHY and IP stack standards. Specifically Ethernet, Wi-Fi and Thread should be supported;
• Allows limited out-of-the-box initial operation of devices over the network, already before devices have been commissioned with control/binding relations or configured with network-related information.

The requirements for network joining that the OpenAIS reference architecture poses upon an architecture realization are listed in Section 4.1.4.1.

3.5.6 Group Communication

The basic OpenAIS group functions have been explained in Section 3.3.3.5. Here, more details are given on how these group functions are handled by a network stack and mapped to the IPv6 network.

3.5.6.1 Features

The OpenAIS architecture aims to offer the following features for group communication, by extending on LWM2M 1.0:

• CoAP Multicast [RFC7252] [RFC7390] protocol over IPv6 multicast.
• Standard LWM2M resource structure supported, as much as possible.
• Support for multiple Application Groups re-using an IPv6 Multicast Group, to obtain efficiency in the usage of IPv6 multicast addresses.
• Comply with the OpenAIS security model, offering authentication and authorization for multicast group commands.
• Multiple instances of a particular Object type are reachable with a single group request, even if the Object instances have different identifiers across different devices or if some of the Object instances happen to reside on the same device.
• Multiple IPv6 Multicast Groups (destinations) supported per OpenAIS device.
• Only one (unicast) IPv6 address is needed per OpenAIS device, even though it can join multiple IPv6 Multicast Groups.

3.5.6.2 Group communication patterns

There are several ways in which CoAP multicast can be used:

• Sense Object → Control Objects: event information is multicast
• Control Object → Actuate Objects: actuator command is multicast
• Control Object → Control Objects:
  o control command is multicast (can be used for stacked control)
  o event information is multicast (can be used for multi-controller coordination)

3.5.6.3 Application group IDs and naming

Application Groups are identified by a short group ID, which is a 16-bit unsigned integer that is used to access the Group Object instance. Each Application Group also has a long ID, the Application Group ID, which is unique over the scope of an entire OpenAIS system. Its format is a UTF-8 string with a number of characters limit to be defined in the OpenAIS Object Data Model. Whether the long group IDs are descriptive (e.g. “Floor3-RoomB”) or opaque (e.g. “3e4a937b”) is up to the vendor. It is RECOMMENDED to use opaque names and store the group name in the separate human-readable ‘name’ field that is stored as part of the group information in the group Object instance. As descriptive IDs are created by commissioner(s) they are not necessarily unique across a system.
3.5.6.4 Multicast requests using CoAP

The CoAP protocol [RFC7252] over UDP is used for group communication, based on multicast CoAP requests sent using the unsecured coap:// scheme. See also Section 3.5.3 for general CoAP usage in the OpenAIS architecture.

CoAP multicast is also used in Discovery (see Section 3.5.8).

A multicast PUT request can be used to change a specific value (e.g. actuator setting) for a group of Object instances. The request payload in this case carries the new resource value to be set, e.g. a new dim level to be set for a group of light-points.

A multicast POST request can be used for a style of communication which is akin to Remote Procedure Calls (RPC). This may also be called command-style communication. The request payload in this case carries a description of the command or action to be executed by all receiving Object instances. For example, a command to execute a fade towards a new light scene over a period of 5 seconds, or a request to step up the current dim level by 10%.

A multicast GET request can be used to read out status information from all Object instances in a group. A GET request triggers a CoAP response from each group member containing the requested status information in the payload. For example, a request to all luminaires on a floor to report their current error log summary to the sender of the request. Note that the CoAP protocol ensures that the response is delayed by a random amount of time to avoid flooding a constrained network with responses. The random response delay interval is configurable with the CoAP parameter “Leisure” which defaults to 5 seconds.

3.5.6.5 Multicast reliability

IPv6 UDP multicast communication is inherently unreliable. Due to the absence of acknowledgements/responses typically, the global system state can become inconsistent as senders will not be aware of failed communications. To deal with this a number of application level mechanisms have been defined in the architecture (see section 3.3.9.2). On top of that on network level a (one time) repetition mechanism for all operational multicast messages will be used. The repeat period will be randomized within boundaries to avoid message traffic peaks.

3.5.6.6 Resource structure for group requests

For group communication, a specific URI structure within a CoAP multicast request is used. This structure is defined as part of the Object Data Model. The following URI template shows how the Group Object instance is accessed:

```plaintext
/<Group-Object-ID>/<object-instance-id>
```

Where <Group-Object-ID> is an OpenAIS defined Object type that provides access to group communication functions. Within this Object instance, various resources will be available to manage group membership and its relation to Multicast Group and Security Group.

The below URI template example shows how a CoAP request could be made addressed to an Application Group. This is an example only; the actual URI will be defined later as part of the Object Data Model.
Where

- `<group-id>` is the system-wide unique Application Group ID. Note that, in case of this example, the namespace of Group IDs MUST differ from the namespace of 16-bit Object instance IDs which are also present under the parent resource `<Group-Object-ID>` and were used in the previous URI template.
- `<handler>` is the incoming message handling resource for the group. It is the responsibility of the handling Group Object instance to distribute the incoming message further, after security validations, to all the Object instances that are a member of the group `<group-id>`.
- `<Object-ID>` is the Object ID to which the group request is targeted. Providing an Object ID here restricts the group operation to only the Object instances of this specific type within the Application Group.
- `<sub-resource>` is an optional deeper (child) resource to target the request to. This is only usable if the OpenAIS Object Model will support these deeper resources, but a provision has been made for this already now.

The above enables group control of multiple Object instances across different OpenAIS devices and also across multiple Object instances residing on the same OpenAIS device.

Note that the three-level URI elements hierarchy of LWM2M 1.0 is in this case extended to at least five levels of URI elements, which constitutes a modification to LWM2M (see Section 3.5.2.3).

### 3.5.6.7 Configuring groups at sender and receiver sides

Since group communication is secured, per Application Group in use a Group Vector (see Section 0) needs to be configured both at the sender (such that sender can use the right key material to send) and at the receivers - such that a receiver can subscribe to the correct IP multicast group and also use the right key material to decode and verify a group message. A node that has been configured this way may act in both sender and receiver roles, or in only one of these.

Read access to group configuration requires a Level 3 or higher authorization level. Writing/modifying the group configuration requires Level 4 or higher authorization level.

The group configuration at the sender(s) consists of these operations:

1. Configuring Application Groups
2. Configuring Multicast Groups
   a. A node that is configured to receive group messages will also start to listen to (i.e., join) the configured IPv6 multicast address.
3. Configuring Security Groups
   a. Including these optional steps: the actual security keys to use may be derived or fetched by the node based on the configured security material, at a later time after the configuration.
4. Configuring Group Vector for each Application Group
5. Creating a binding from a specific sending Object instance to a receiving Application Group. See Section 3.5.7 for Bindings.
The group configuration at the receiver(s) consists of the same operations as above, except that the binding step is replaced by:

5. Configuring which Object instances are part of each Application Group.

3.5.6.8 Security usage and levels

A group communication operation can be used for two purposes:

1. Discovery – in this case unsecured group communication is used, since the discovering device and discovered entities may not yet have a security relation set up.
2. Operational communication including group control – in this case secure group communication MUST be used.

The following authorization levels (see also Section 3.2.4.2) are used with group communication i.e. in a CoAP multicast request:

- Level 0 Service detection: used for multicast based discovery operations
  - This is unsecured at both transport and application layer
- Level 1 Reporting only: used for reporting sensor events in multicast
  - Secured at application layer
- Level 2 Standard use: used for group control operations (e.g. switch scene, dim lights) or requesting parameters/values read-only (e.g. status information)
  - Secured at application layer

Group communication MUST NOT be used nor allowed for operations that require a higher than level 2 authorization. Configuration operations MUST only be possible under a minimum Level 3 “Commissioning” level of security.

Levels 1 and 2 for group communication operations are secured at the application layer using COSE, see Section 3.6.5.3.

3.5.6.9 Configuring and updating group security material

For a Security Group, the security material consists of Access Tokens (AT-KDC) that enable a node to fetch the actual group key in use at the Key Distribution Centre (KDC).

The KDC and Authorization Server (AS) defined in [Ace] are used by nodes to obtain and maintain the proper security keys (tokens, etc.) for operation.

There are two cases:

1. Backbone-connected networks: KDC is available online all the time and hence can be used by nodes to perform the periodic group key update.
2. Stand-alone networks: KDC is only available when a Commissioning Tool (CT) is connected in the local network (e.g. Field Network), such that group key cannot be updated while the CT is away.

Both cases make use of the same mechanisms of KDC/AS; only in case 2 the KDC/AS are only temporarily available at some points in time.
Configuring group security material requires Level 4 or higher access. Reading group security material or keys is disallowed. See Section 3.6 for more details on security concepts.

3.5.6.10 Binding security group to allowed Object instances only

A node N can be member of multiple application groups. So there may be, on a single node, Object instances that are part of a group A and other Object instances that are part of a group B. This implies that all group members of group A trust device N and are entitled to send group requests to it; and the same holds for group B: all group members of B trust device N and are entitled to send group requests to it.

But even in this case, a trusted member of group A should not be able to control any Object instance on node N which is only part of group B but not of group A. So a sender of a group request should not be ‘trusted to be polite’ and it should be enforced to not access Object instances that it is not authorized to access.

Therefore in OpenAIS systems an IPv6 multicast receiving node MUST perform the following check:

- First, the Group Vector of the sender is reconstructed by taking Application Group from the CoAP request URI, Multicast Group from the IPv6 packet destination field, and looking up the related Security Group from locally stored Group Objects.
- Second, compare the Group Vector of sender to the Group Vector of the receiving Application Group.
  - Only if there is an exact match, the group message can be accepted.
  - If no match, the group message is silently discarded.

3.5.7 Binding and Eventing (Subscriptions and Notifications)

3.5.7.1 Binding

The concept of “Binding” is defined here as a persistent, configurable relation between a producer of information and a receiver of this information, where the receiver typically -but not necessarily- resides on a different network Node. In software architecture this is commonly known as a subscription relation or a subscription-notification mechanism. In OpenAIS we do not use the latter terms (subscription/notification) because “notification” is already used in LWM2M context to indicate a specific mechanism that uses CoAP Observe, also called observation/notification mechanism [LWM2M].

An OpenAIS binding is created and configured at commissioning time. For example, a presence-sensing function may be bound to a group G1 of Control Objects. Or a Commissioning Tool configures a binding from a room Control Object to a floor Control Object, such that the floor Control Object is permanently kept up to date about the occupancy and lights status of the room.

Binding is not necessarily restricted to the lighting application domain: the binding mechanism may be applied to other entities – e.g. a binding can be created between a producer of a diagnostic event and a consumer of this information (such as a diagnostic tool).

As a short-lived alternative to binding, a temporary observation relation can be created upon request of an entity that wants to receive event information. Such short-lived
mechanism can be implemented using the CoAP Observe function that is a standard part of LWM2M 1.0.

The OpenAIS implementation of binding is inspired on the principles of the CoRE Interfaces Binding [Core-intf] work and uses standard LWM2M read/write mechanisms to configure the bindings.

An improvement that OpenAIS makes on the [Core-intf] architecture is that OpenAIS distributes the bindings over multiple entities (e.g., Binding Objects or binding resources) instead of keeping one big binding table for the entire device in a single CoAP resource. The latter approach has a scalability issue if the number of bindings grows larger (e.g. in future systems). The exact details of binding configuration will be defined as part of [OpenAIS_D2.4].

3.5.7.2 Eventing

The concept of “Eventing” is used here to mean notification: the method(s) defined in OpenAIS via which Object instances send Events to other Object instances, potentially over the network. Eventing is usually enabled/activated after a suitable Binding has been set up.

The typical use of eventing i.e. sending of events (refer to Figure 21) is as follows:
- On the left (Sense-Function → Control-Function), event information is unicast or multicast to Control-Functions.
- In the middle (Control-Function → Control-Functions), optionally event information is multicast to other Control-Functions.

Figure 21: OpenAIS Sensor-Controller-Actuator (SCA) model and the role of bindings and events in this model

Another usage outlined below is less typical, but could be easily enabled using the generic OpenAIS eventing mechanism:
- On the right (Actuate-Function) events are sent upon an internal (status) change of the Actuate-Function. For example, events for diagnostic purposes.
3.5.7.3 Features

This Binding and Eventing architecture aims to offer the following features:

- Based on existing mechanisms in LWM2M, CoAP, CoRE Link Format [RFC6690], and CoRE Interfaces [Core-intf]
- Support for all three of unicast, multicast (group), and serial unicast (group) ways of communicating Events to the receiving entity/entities.
- Works with the OpenAIS/LWM2M resource model and preferably should work with other application-level languages, if possible.
- Secure multicast using COSE is supported for sending out group events securely
- Secure unicast eventing over CoAP+DTLS is supported
- A sensor device may contain multiple types of sensors, that is, it can host multiple sensing-related Object instances, for which individual different bindings can be configured.
- A single physical sensor is able to generate events for different receivers with different needs with respect to the type of information received and the frequency of receiving this information. In other words, a single physical sensor can support multiple bindings where each binding can be individually parameterized. The receiver per binding in this case may be a single OpenAIS device or a group. However, it is recommended to reduce the number of multicast (group) bindings to the absolute minimum, typically only one, to avoid network scalability issues.
- The event information can have an arbitrary simplicity or complexity, i.e. ranging from simple values (integer, float, boolean, string, etc.) to data structures (CBOR, JSON or TLV). For OpenAIS, both CBOR and simple values are used.

3.5.7.4 Application protocol and resource structure

CoAP [RFC7252] is used for all operations like sending events or configuring bindings. Using CoAP, the URI structure of LWM2M and also the URI structure for Group Communication (see Section 3.5.6.6) need to be supported. Specific resources for binding configuration and event reception will be defined as part of [OpenAIS_D2.4].

3.5.7.5 Sending an event notification to a single receiver

A single event to only a single receiver is sent as CoAP unicast request from a CoAP Client to a CoAP Server. Note that the sending of an event is triggered by a specific Binding.

Using CoAP PUT or POST?

Events are sent typically as CoAP POST requests containing the event information as CBOR encoded data within the payload. Sending CoAP PUT requests is also supported similar to the ‘push’ type binding in [Core-intf]. This is typically used if some value (e.g. parameter, or dim level) at the sending side needs to be replicated at the receiving side.

Besides the push type binding as defined in [Core-intf] a new binding type ‘post’ is defined here to send out CoAP POST requests with event information in the payload. Events sent in this manner could be described as “RPC style” sending of events from sender to receiver. The receiving CoAP resource then parses the event that is inside the CoAP payload and processes it in its own specific way. The processing may have as a side effect one or more changes to resources on the receiving device.
3.5.7.6 Sending an event notification to a group
The CoAP protocol [RFC7252] over UDP is used for group communication using multicast CoAP requests, as detailed in Section 3.5.6.

3.5.7.7 Security
Regular CoAP transport-layer security (DTLS) is used to secure all unicast operations for binding and eventing. Group communication (multicast) security as described in Section 3.5.6 is used to secure multicast event notifications.

Operations to create, modify or delete bindings MUST only be possible under a minimum "Commissioning-structural" level (Level 4) of authorization.

3.5.8 Discovery

3.5.8.1 Introduction
The general function of ‘Discovery’ consists of the following sub-functions:

- Device Discovery – identifying which (OpenAIS, CoAP) devices are present in an OpenAIS network or network segment
- Object Discovery – identifying which Objects and Object instances are present in an OpenAIS device
- Resource Discovery – identifying which (CoAP) resources are present in an OpenAIS device, including resources that are part of an OpenAIS Object.
- Group Discovery and Group Member Discovery – finding which groups are defined in a system, or on specific nodes, and finding which nodes or Object instances are part of a given group.

3.5.8.2 Usage
Discovery is used in the following cases for example:

1. A commissioning tool needs to identify the OpenAIS devices present in a given system, or devices present locally in an IPv6 subnet.
2. A commissioning tool needs to identify device types and device identifiers of a given system, without a-priori knowing the security keys to access these devices
   a. Note: once devices are identified the tool may fetch the required keys securely from a server, to gain further access.
3. A commissioning tool needs to identify what Object instances are present on OpenAIS devices in a given system.

3.5.8.3 Discovery mechanisms
The basic service/resource discovery mechanism as defined in LWM2M 1.0 is supported by all OpenAIS devices. This is based on CoAP [RFC7252] Section 7 and the Link Format RFC [RFC6690].

In addition, also multicast CoAP discovery via the /well-known/core resource as described in Section 8 of [RFC7252] MUST be supported by OpenAIS devices via the "All CoAP Nodes" IPv6 multicast address FF0x::FD for at least the scopes Link-Local and Realm-Local [RFC7346].

A discovery requester SHOULD include filter parameters in a request as defined in Section 4.1 of [RFC6690] to avoid triggering a storm of multicast responses from many
OpenAIS nodes. Furthermore, multicast discovery operations SHOULD NOT be used during normal system operation but rather for (re-)commissioning or installation-time operations only.

Group (Member) Discovery is supported through a combination of Device Discovery or Object Discovery and subsequent individual reading out of group membership information through the OpenAIS Group Objects.

3.5.9 Network Management

Network management in OpenAIS includes any functions to remotely configure and diagnose operational IP networking equipment in the field.

An OpenAIS end node will support at least the network management functions defined by the LWM2M specification. Network equipment such as routers and access points will support standard IT network management protocols such as SNMP or NETCONF [RFC6241].

OpenAIS may add specific wireless network quality monitoring functions, in case that the standard LWM2M statistics are insufficient.

3.5.10 Low-power “Sleepy” Devices

3.5.10.1 Introduction

The OpenAIS architecture supports so-called “sleepy” devices, which are networked devices that have their communication interface switched off for a majority of time in order to save on energy consumption. Very often these are wireless sensors, operating on batteries or energy harvesters.

Some PHY/MAC technologies can support sleepy devices. It is more common for wireless technologies because it provides the advantages of fully wire-free installation and operation.

For OpenAIS we assume that any supported sleepy device will be designed with relatively short sleep times such that CoAP communication to the sleepy device still appears as “regular CoAP communication”. That means the sleep time period of the sleepy device must be shorter than the maximum final time-out for requests in the CoAP protocol, which is 93 seconds for the default CoAP parameter values [RFC7252]. For more information on handling sleepy CoAP devices with even longer sleep times see [Core-sleepy].

A sleepy device typically needs specific support in the PHY/MAC, such as having direct RF connectivity to a non-sleepy parent router. Such PHY/MAC technologies operate the sleep function at L1-L2 using radio duty cycling. They do this in such a way that the devices appear as “always-on” to the higher layers (L3-L7), like always-on devices with a slower average response time or at most an occasional message loss. The sleepy devices will, on wake-up, poll their parent router to see if there is an L2 packet to be received and if so retrieve it and process it.

3.5.10.2 Sleepy device network joining

In order to join a network, a sleepy device may first go into an always-on mode (or a mode in which it appears as a responsive always-on device from the L3-L7 protocol viewpoint). Then it can join in the same manner as regular devices. After the network
join operation and commissioning, a sleepy device goes into a designated sleep mode with regular (periodic or aperiodic) wake-ups. The selection of sleep mode here may be influenced by the settings made during commissioning / device management operations.

3.5.10.3 Sleepy device multicast operation
Sleepy sensors must be able to send out multicast events containing information about a sensor state change. Therefore, a wireless IPv6 networking stack that supports sleepy devices MUST have a mechanism for sleepy devices to transmit IP multicast UDP packets, either directly or via an always-on parent router.

3.5.10.4 Sleepy device eventing/binding/configuration
A sleepy sensor device is typically also a producer of events. A sleepy sensor can wake up its communication module upon a detected physical event (e.g. substantial change in a sensed parameter), connect to the network and send out a CoAP request containing the event. After this, and perhaps after doing optionally a poll for incoming data, the sensor goes back to sleep.

If a sleepy sensor needs to be (re)configured e.g. bindings need to be changed, an authorized device such as a commissioning tool or remote service should be able to do these configurations quickly, so the wait time until the next wake-up of the sleepy device should not be too long. This can be ensured by choosing a short enough maximum sleep period as a default, or during device commissioning.

3.5.11 Legacy IPv4 Networks Interfacing
Most of the office buildings in which IPv6 based OpenAIS networks will be deployed might have a legacy IPv4 backbone network. The design goal of the IPv6 protocol was to solve many limitations of the IPv4 protocol and to add new features, but not backward compatibility. Due to this, IPv6 packets cannot be directly routed in IPv4 networks.

OpenAIS recommends use of transition technologies like tunnelling [RFC7059] and NAT64/DSN64 [RFC6146] in order to enable easy integration of IPv6 networks with legacy IPv4 networks.

The NAT64/DNS64 [RFC6146] is an IETF defined standard to enable IPv6 only hosts to communicate with an IPv4-only server using an IPv4-only network. The standard defines algorithms necessary to transparently translate between IPv6 and IPv4 headers depending on the direction of communication.

A Low Power Radio Access Point (LPR AP) that connects an IPv6 network to a backbone network can make use of NAT64/DNS64 to translate IPv6 headers to IPv4 headers in unicast messages sent by an OpenAIS IPv6 end node before sending them onto to the IPv4 backbone network. This is used to contact legacy IPv4 hosts from IPv6-only OpenAIS devices, or vice versa.

Unicast between two OpenAIS IPv6 devices across an IPv4 backbone can be implemented using tunnelling [RFC7059] by letting the LPR AP of both subnets set up a tunnel towards a central tunnel server. In this way, multiple tunnels (one per subnet) may be set up at a single tunnel server.
IPv6 multicast between OpenAIS devices across subnets is handled in the same way, using tunnelling.

3.5.12 PHY/MAC Requirements

As discussed the OpenAIS architecture considers the network to consist of non-constrained and constrained parts. The architecture will support a chosen set of default physical layers, but will not preclude the use of any well specified (i.e. fully specified, e.g. by IEEE/IETF) wired or wireless physical layer that can support IPv6, either directly or via the 6LoWPAN adaptation layer. Such a non-default PHY/MAC must meet a set of minimum requirements defined by the OpenAIS reference architecture. These requirements are listed in Section 4.1.4.2.

We have verified that the following default set of IPv6 supporting network technologies selected for OpenAIS can meet the requirements: 6LoWPAN/Thread mesh networks, Ethernet/PoE wired networks, Wi-Fi wireless networks.

3.6 Security View

3.6.1 Introduction

This section describes the security concepts and the end-to-end security in the OpenAIS architecture. It defines a security design which complies with the basic security requirements as defined in Section 3.2.4.

Next to the security requirements also some other key requirements have to be taken into account when designing the security system:

- Message communication in groups must have low latency and high synchronicity (latency < 200 milliseconds).
- The system must be able to operate in isolation without connection to servers or the internet.

3.6.1.1 Network security

To support the security requirements we have the following approach:

- Use standard network communication security mechanisms as far as possible to protect all unicast communication above access level 2.
- IPv6 multicast communication will be used to comply with the latency requirements for group communication, up to access level 2.
- Protection of the Peer-to-Peer and Device-to-Cloud communication (end to end).
  1. For all multicast communication, COSE [Cose-msg] based object security will be used
  2. Unicast messaging can also use COSE based object security. Alternatively DLTS may be used here.
- Use symmetric keys for the COSE end-to-end security in multicast communication.
- Authorization of any CoAP client/server communication: which device may send commands and to whom in the network will be configured at commissioning time or recommissioning time.
- Nodes are pre-configured with credentials and private keys for initial authentication and authorization.
• An Authorization Server must be present at commissioning time. The Key Distribution Centre need not be available at commissioning time and needs to be available only when the devices “need a key” e.g. during key update or before first use. This will allow a commissioner to configure groups when no key distribution server is present and such a server can distribute group keys when network connectivity is available.

• Link layer protection over (6LoWPAN) wireless networks with symmetric key encryption.

• No link layer security on Ethernet wired connections.

3.6.1.2 Malicious software

One of the main threats to be countered is the installation and execution of non-authorized (malicious) software. Such software could be loaded either over the network (by hacking the firmware update mechanism) or by attacking the device over its physical interfaces like I2C, serial line etc. after opening a luminaire. Although the OpenAIS reference architecture does not define strict rules for the security level of devices (there is manufacturer freedom in differentiating on security aspects), at least protection measures defined in the above section and authorization levels need to be implemented for the software update function. For the physical protection no minimum requirement is given. This section gives an overview of the optional measurements that are suggested for device protection.

The main mechanism for device protection is a so-called secure boot mechanism. In a secure boot the system enforces that only software proven to be from the original manufacturer can execute on the device. A few aspects to consider:

• The CPU must be forced into running known code. It should be prevented that someone loads code in memory and just sets the program execution there. Therefore secure boot is a hardware mechanism that enforces the CPU to start at a certain point after any reset/stop. This is normally a piece of ROM code in the IC that executes the loading and checking of the first code from flash memory.

• Note that there is sometimes an entire chain of loads needed. The ROM code loads the first piece of software from flash (at a designated address) and validates that this originates from the manufacturer. This could then be the real boot loader that loads (and checks) the rest of the software. In this way only software that has been checked can execute.

• Sometimes the secure boot can be activated with blowing a fuse, which can be done after manufacturing. This is a one-way process, blown fuses cannot be reversed.

• The signature is normally a hash/checksum of the code which is encrypted with some private key. The public key is stored in hardware in such a way that it cannot be overwritten in the device and is used for decryption. As long as the private key remains secret in the organization of the manufacturer this is pretty safe.

Some additional (optional) measures suggested are:

• To deal with situations where the private key leaks out one could actually store a few different public keys in the IC. The current key can then be selected by a hardware selection mechanism e.g. blowing a fuse.
• Of course this still means that potentially all devices will be potentially corrupted if a private key leaks out. However giving all devices unique keys and matching uniquely signed software is logistically infeasible.

• To speed up the decryption and to safely reuse the HW by software in subsequent loading steps a number of algorithms most likely need to be implemented in hardware (like SHA, AES, and ECC).

Finally note that the update mechanism itself is protected to avoid download of code:

• Configured devices may be upgraded only if the access level 5 credentials of the system are applied. If a device is upgraded without level 5 credentials all the keys and the binding information need to be deleted before the upgrade can be performed. This reduces the risk of a leaking the key of a manufacturer: the local system credentials are also needed to get the device operational and do its malicious task; this reduces the effect of a stolen key dramatically.

### 3.6.2 Overview

In this section, we give an overview of the Security Architecture of the OpenAIS system. The objective of the security architecture is to reuse the LWM2M specification as far as possible but include any additional changes required for the lighting application. In particular, the following (lighting specific) factors influence the security architecture:

• Support for role-based access control.
• Support for Peer-to-Peer communication which includes secured and unsecured group communication.
• Bootstrap process that does not depend on Internet connectivity to a central server.

#### 3.6.2.1 Actors, functions and terminology

Figure 22 shows a high-level overview of the system and displays the different actors and roles that are relevant to the security architecture. Note that all these actors might not be present in every installation. The grey boxes represent permanently installed devices and white boxes denote devices that may appear temporarily in the network. Often, these temporary devices are linked to a single or multiple (authorized) users who are allowed to access different resources on the OpenAIS network.

The terms used in this figure are further detailed in the Glossary (Section 9). Note that throughout the Security View section, the LWM2M Bootstrap Server is also referred to as ‘bootstrap server’; and the LWM2M Server as ‘device management server’.

In Figure 22, the solid lines denote (link or mesh-local) network connections between nodes and dotted lines denote application layer connections. For instance the wired nodes are shown to be connected to the L2 Ethernet switch via a solid line and the wireless nodes are shown to be connected to a border router via a solid line.
3.6.2.2 End-to-end security and authorization policy

The objective of end-to-end security is to ensure that the OpenAIS authorization policy that specifies access rights of different CoAP clients for services offered by OpenAIS CoAP servers is enforced. There are essentially two types of CoAP Client/Server interactions that occur in our system:

1) Peer-to-Peer communication: This type of communication occurs between permanently installed lighting devices for lighting operational communication. Example includes turning a (group of) light(s) on/off using a push button after the push-button and luminaire(s) have been configured to work together by a commissioner. Such communication is secured using Object security at the application layer.

2) Device-to-Cloud communication: This type of communication typically occurs between a permanently installed device and a centrally running software application. Note that the software application may be running also on a mobile device or a Commissioning Tool (CT). Examples of such communication include: Device-to-LWM2M Server, Device-to-CT, Device-to-KDC, Device-to-AAA Server, Device-to-LWM2M Bootstrap Server, etc. Such communication is secured used DTLS sessions.

The OpenAIS authorization policy, which is applicable to both Peer-to-Peer and Device-to-Cloud type communication, stipulates that only authorized CoAP clients are allowed to access CoAP server resources. In typical applications, the authorized CoAP clients are naturally categorized into one of multiple roles (e.g. lighting operational, commissioning, maintenance etc.)

Therefore, the device’s CoAP resources within any Object are categorized into groups based on the six access levels which are defined in Section 3.2.4.2.
Note that the access level of any OpenAIS service resource needs to be specified within the Object Model and that this access level is not determined at runtime. The authorization policy that every OpenAIS device needs to implement is the following:

*Only a CoAP client with authorization level greater than or equal to the category (security level) of the CoAP server resource is allowed to access the resource. It is possible to further implement more fine grained access control for lighting operational resources based on group membership (e.g. a level 2 client is authorized to access level 2 server resources via group communication only if the CoAP client and CoAP server belong to the same group).*

### 3.6.2.3 Access Control Lists and Security Objects

To implement the authorization policy explained in the previous section, we use the Access Control Lists and the Security Object from the LWM2M specification [LWM2M] with some small extensions/modifications for Peer-to-Peer security and for role-based authorization. The extensions are based on credentials object from the OIC standard [OIC] and the ACE Object [Ace]. For Peer-to-Peer communication, one of the peer devices is a CoAP client and the binding Object in the CoAP client will also use the Security Object to authenticate itself with the remote device’s CoAP server.

The access control method is based on two steps: 1) Use the Security (credentials) Object to authenticate the remote device using either the DTLS handshake or (COSE) object security and 2) Use Access Control List to determine if the authenticated remote device is authorized to access the local resources or determine which credentials to use to authenticate itself to the remote device in case the local device is a CoAP client that initiates the access.

### 3.6.3 Secure Bootstrap

Based on LWM2M, the following four Bootstrap modes will be supported:

1. Factory Bootstrap
2. Bootstrap out of band (e.g. smartcard, NFC, Thread network pairing DTLS session)
3. Device Initiated Bootstrap
4. Bootstrap Server Initiated Bootstrap

The first two modes are identical to the LWM2M specification and the reader is referred to the LWM2M specification [LWM2M] for details. The device initiated and bootstrap server initiated bootstrap methods are modifications of the LWM2M specification proposed by OpenAIS because we assume internet connectivity is not guaranteed during the bootstrap process. For any bootstrap process, at the conclusion of the bootstrap process:

1) Any device MUST have at least one security Object (see Section 3.6.4.1) specifying a remote CoAP client with the highest possible access level of Device Owner (i.e. this CoAP client is allowed to access every resource on the system).
2) Any device SHOULD have the bootstrap information as specified in Section 5.1.1 in the LWM2M specification [LWM2M]

The bootstrap processes for both device initiated bootstrap and bootstrap server initiated bootstrap are shown in Figure 23.
The main change from the LWM2M bootstrap process is within the discovery phase of the bootstrap process. Typically, in lighting installations, it is not possible for the device to know the location and credentials of a LWM2M Bootstrap Server which is a software function usually running on installation-site specific server or as a mobile application. Therefore we introduce in OpenAIS the discovery step in the bootstrap process.

3.6.3.1 Bootstrap Server security Object

At manufacturing time, every device is configured with at least one bootstrap server security Object (see Section 3.6.4.1 for security Object details). This bootstrap security Object provides the device with one of the following types of credentials of bootstrap server: 1) Bootstrap Server Public Key Certificate, 2) Bootstrap Server raw public key/ID, 3) Pre-shared Key/ID, 4) Pre-shared password, 5) No security. The bootstrap server security Object SHOULD NOT use the No security mode.

The type of security Object provided is manufacturer specific and methods must be provided by the manufacturer that allows the bootstrap server to obtain the required...
keying material to establish the DTLS session with the Device (LWM2M Client) during the bootstrap process.

3.6.3.2 Discovery

The discovery phases allows a Device and a bootstrap server to discover each other using standard CoAP (multicast) discovery methods as explained further in Section 3.5.8.3. The Device and bootstrap server initiated bootstrap methods allow either the Device or bootstrap server to initiate the discovery process. Both of these discovery processes are done without any security and the corresponding CoAP resources for the discovery process MUST be access level 0 resources.

Once the Device and bootstrap server discover each other, the Device MUST provide its unique identifier to the bootstrap server using the CoAP confirmable POST method on the /bs resource with the Device_UUID as payload. After a device provides its UUID to the bootstrap server, the bootstrap server MAY optionally request the identity or bootstrap server public key of the bootstrap server security Object inside the device and the device in that case returns the security Object with includes key identification information. In the response the Device MUST NOT return any private keys that may be stored within the security Object also.

3.6.3.3 Bootstrap

The bootstrap server MAY use the Device_UUID and the information returned within the security Object to obtain keying material that the bootstrap server uses during the DTLS handshake with the device. The method used by the bootstrap server to obtain this keying material is manufacturer specific and is out of scope for the security architecture. For each mode of the bootstrap server Object we explain what the manufacturer specific process needs to accomplish.

1) Bootstrap Server Public Key Certificate

The bootstrap server must be the owner of the public key in the certificate and the device authenticates the bootstrap server based on establishing a DTLS session using the public key of the Device. The bootstrap server MAY use the device UUID to acquire knowledge of a public key belonging to the device or a pre-shared key or password belonging to the device. This public or pre-shared key or password MAY be used in the DTLS session for the bootstrap server to authenticate the device.

2) Server raw public key/ID

The bootstrap server MUST obtain the private key associated with the raw public key inside the bootstrap server Object inside the Device, using a manufacturer specified method. Examples include talking to a cloud service to get the private key, private key printed on a QR code, NFC tags etc. The device authenticates the Bootstrap server by establishing a DTLS session based on the raw public/private key pair. The bootstrap server MAY use the device UUID to acquire knowledge of a public key belonging to the device or a pre-shared key or password belonging to the device. This public or pre-shared key or password MAY be used in the DTLS session for the bootstrap server to authenticate the device.

3) Pre-shared Key/ID

The bootstrap server MUST obtain the private key associated with the device UUID/key ID in the bootstrap server security Object using a manufacturer specified method. Examples include talking to a cloud service to get pre-shared key, PSK on a QR code, NFC tags etc. The PSK is used in the DTLS session by the device and Bootstrap server to authenticate each other.
4) Pre-shared password
The bootstrap server MUST obtain the password associated with the device UUID/key ID in the bootstrap server security Object. Examples include talking to a cloud service to get the pre-shared key, PSK on a QR code, NFC tags etc. The password is used in the DTLS session (with the [JPAKE] extension) by the device and bootstrap server to authenticate each other.

5) No security
The device does not authenticate the bootstrap server and establishes a DTLS session in the no-sec mode. The bootstrap server MAY use the device UUID to acquire knowledge of a public key belonging to the device or a pre-shared key or password belonging to the device. This public or pre-shared key or password MAY be used in the DTLS session for the bootstrap server to authenticate the device.

After this, the bootstrap process follows the LWM2M recommended method to write the security Object for the device management server into the Device.

The Bootstrap server and the device management server must exchange device specific keying material. The method to achieve this exchange is out of scope of this document. Note that in several instances the bootstrap server and device management server maybe collocated. Note that when the bootstrap server generates keying material for the device management server security Object the bootstrap server MUST have access to randomly generated keys (see e.g. [RFC4086] for guidelines).

3.6.3.4 Registration
The registration process now follows the LWM2M recommended method to register resources with the device management server. Note the device management server may be running on a mobile device and may not always be available after commissioning.

3.6.3.5 Secure removal of device / decommissioning
The secure removal of devices requires the deletion of client credentials on the device management server and the deletion of server credentials on the device. The second part is accomplished by the deletion of the server security Object on the device.

Note that any “access to device without keys”, e.g. by out of band methods requires first secure deletion of stored keys before the access can take place.

3.6.4 Authorization, Authentication, and Key Handling
The authorization policy of the OpenAIS system is described in Section 3.6.2.2. In the present section we explain how the device provides mechanisms to support this authorization policy based on extensions to the LWM2M security Object (which stores credentials of a remote device) and LWM2M ACL Object, which specifies the authorization policy on the devices and links security Objects to specific local resources to which the remote device has access. The security Object may also be used by the binding Object to allow a device to authenticate itself to the remote device. Both the Security Object and the ACL Object will be detailed in [OpenAIS_D2.4]. In the following, we will describe the main modifications to the LWM2M specification and explain how these modifications will be used in the security architecture.
3.6.4.1 Security Object
The security Object allows the storage of the security keys that are used to authenticate the remote device and/or authenticate itself with the remote device. The security Object could be linked to the role of the remote device (e.g. CoAP client role discussed above) or subject based that uniquely identifies the LWM2M server. Also, the security Object could enable group access for group communication. The key components of the security Object are:

1. Security Object instance (Credential ID in [OIC]): Which is a local identifier to reference from other resources.
2. Server ID (in DTLS) / Security-Group-ID (in COSE): This ID identifies the security to be used with a specific server or by a group the credentials are applied to.
3. Role ID: Tells the role of the remote device corresponding to this credential; an access level value between 0 (service detection) and 5 (owner).
4. Security keys: The keys that need to be used for this remote device (could be symmetric key, group key, raw public key, public key certificate based).
5. Transport Layer Security mode: Determines the transport used to secure the communication - DTLS, COSE or SMS.

3.6.4.2 Access Control List
The access control list implements the authorization policy. The LWM2M ACL Object determines the access rights of a remote LWM2M server, identified via the short server ID. Roughly speaking the ACL Object consists of a table with three entries. 1) The short (LWM2M) server ID, 2) Pointer to the local Object instance and 3) Permissions which explain the set of operations the remote device identified via the short (LWM2M) server ID is authorized to perform.

In addition to the ACL Object in LWM2M, an OpenAIS server MUST support a role based access control Object. The role based access control Object will be described in detail in [OpenAIS_D2.4]. The role based access control Object replaces the Short (LWM2M) Server ID with a Role ID. The Role based access control is used in an identical fashion to the LWM2M ACL Object with one exception – the Object determines the access rights of any remote LWM2M server or group member which has been authenticated to be a CoAP client of the corresponding role using the role ID in the security Object.

3.6.5 Secure Communication
3.6.5.1 Link security
The link layer security depends on the choice of the link layer used and therefore details on this topic are out of scope for this document. For wireless links, link layer security is mandatory while for wired links it is optional.

3.6.5.2 E2E security
The method used for end-to-end security depends on the content of the security Object used to secure the communication. For any unicast CoAP request and response for a resource with required access level greater than 2 (Lighting operational), the communication MUST be secured using DTLS. If the security mode specified by the security Object is one of PSK [LWM2M], RPK [LWM2M] or PKI [LWM2M], DTLS MUST use the cipher suites and security procedures specified in
Section 7.1 of the LWM2M specification [LWM2M]. If the security mode specified in the security Object is passwords then the J-PAKE cipher suite MUST be used [JPAKE].

For all multicast communication, CoSE based object security MUST be used and is explained in more detail in the next section. Unicast CoAP requests and responses for resources at access level 2, MAY use either object security or DTLS.

All requests and responses for access level 0 and level 1 resources MAY be performed with no security bindings.

3.6.5.3 Group communication security
An Object Security format for use in multicast communication and for unicast communication within a group for Access Level 2 resources is currently being standardized within the IETF ACE working group (see [Ace]). The object format specified in the final version of this draft MUST be used for all group messages.
4 ARCHITECTURE REALIZATION

This chapter provides first a summary of design requirements and recommendations derived from the architecture. It then provides more details of the functionality and interaction of the core architectural element “Control Object”, and its usage in some typical lighting controls scenarios. Next it provides more insight into how and where the architectural components may actually be used (deployed) to achieve a versatile but still simple to encompass installation. Afterwards, it looks in more detail into the installation and commissioning process and how the architecture allows for a future proof easy life setting in this aspect. The last part elaborates how the flexibility for the various embedding requirements throughout the life cycle of a controls system may be handled.

This collection of examples, useful hints and guidance in architecture interpretation supports creating systems based on the OpenAIS reference architecture. We focus on the intended blueprint for the OpenAIS M33 Demonstrator design, giving this way a hands-on example how the relatively high level description of the reference architecture materializes into a specific system.

4.1 Component Requirements and recommendations

4.1.1 Introduction

The OpenAIS reference architecture description as laid down in Chapter 3 leaves a lot of freedom for future evolution of standards, protocols and available hardware. As a reference architecture, it also leaves quite some freedom to system designers to construct their components and systems as well. Some of these requirements have been mentioned in their relevant Sections in Chapter 3, this section combines and re-iterates these as a service and easy reference for system designers. Furthermore, recommendations and minimum requirements are listed for implementation purposes.

4.1.2 Logical view

Any OpenAIS compliant system needs to follow the decomposition and mandatory Object Model interfaces as defined in this document and the Object Model [OpenAIS_D2.4].

4.1.3 Physical view

4.1.3.1 Minimum requirements device platform and (RT)OS

As described in Section 3.4, the OpenAIS architecture does not define a specific embedded device platform, nor a specific Operating System (OS) or Real-Time OS (RTOS). However, a set of minimum requirements towards a platform and the OS/RTOS for first introduction in the 2018-2020 timeframe is provided in this section:

REQ 01. Platform must support IPv6-stack
REQ 02. Platform must support UDP, DTLS, COAP (Multicast) protocols in the network stack

4.1.3.2 Suggested Platform Selection Criteria
The following suggested criteria for OS/RTOS platform selection were identified, although these are not 'must' level requirements. The more of these criteria are satisfied, the more suitable/attractive the platform.

SUG 01. Multitasking support
SUG 02. Real time, priority based, primitive scheduling > 64 priority levels
SUG 03. Standard intertask communication and synchronization mechanisms: message queues, semaphores, timers
SUG 04. Fast task switching (< 2 µs) and interrupt latency (< 1µs) (@100MHz CPU)
SUG 05. Small footprint for kernel < 10 kB in ROM and < 3 kB RAM
SUG 06. Standby power management of CPU and peripherals
SUG 07. Software and hardware platform must feature a choice of MCU vendors Support for various embedded CPU architecture like ARM (M and R), ARC, MIPS, OpenRISC etc.
SUG 08. Persistent memory management to support lifetimes of 20 years
SUG 09. Flash file access system support
SUG 10. Memory management support
SUG 11. Standard support available for peripherals like I2C, Ethernet, USB, GPIO, etc.
SUG 12. There is support for the platform either from a community or professional support
SUG 13. C++ compiler support
SUG 14. Support for a high-level language (e.g. Lua, Java, Python, JS, …)
SUG 15. Cross development and simulation environment available on both Linux and Windows platforms
SUG 16. Good development suite on Linux and Windows including compiler, build-environment, symbolic multitask debuggers, performance analysis tooling
SUG 17. Good debug support
SUG 18. Target has AES-128 acceleration
SUG 19. Target has SHA-256 acceleration
SUG 20. Target has ECC-256 acceleration
SUG 21. Platform features malware detection and removal capability
SUG 22. If ARM MCU, Cortex M v6M or v7M architecture with MPU with 8 or more regions
SUG 23. Platform supports firmware upgrade over-the-network; including memory support for firmware upgrade: validation of new image, storage of image and recovery to old image.
SUG 24. Minimum Target resources are at least EEMBC CoreMark 110, 32kB RAM, 256kB Flash Ranging up (2018) to CoreMark > 300, 1 Mbyte flash, 128 kByte RAM
SUG 25. Platform support asymmetric and symmetric security functions as required by the OpenAIS security solution (AES128, DSA, ECDH, ECDSA, DTLS 1.2 - 1.3)
SUG 26. Platform has an internal source of entropy e.g. TRNG
SUG 27. Platform supports separate maintenance and update of security protocols and platform code from business logic
SUG 28. C compiler support
4.1.4 Network view

4.1.4.1 Requirements for layer L1-L3 network join functions

This section lists the requirements towards the technology-specific L1-L3 network join functions. Note that “network stack” here refers to a complete L1-L3 IPv6 stack including the network layer and the MAC/PHY layers below it.

REQ 01. The network stack MUST provide a means for a node to start a new network
  • This is how the first device ‘joins’ a network. Typically, this is a border router or access point, but not necessarily.

REQ 02. A wireless network stack SHOULD provide a means for a node to automatically discover wireless networks nearby, select order for joining based on pre-set criteria, and attempt to join one or more networks found.
  • This is desired to allow out-of-the-box operation with wireless networks.
  • The “joining” mechanism may also be simpler; e.g. instead of discovering networks a node could simply start sending and receiving on a default RF channel with default packet formats that will be received by neighbour nodes. This is considered to be equivalent to “joining” a single default wireless network.
  • If this requirement can’t be met, then no out-of-the-box functions are possible prior to commissioning.

REQ 03. At least one global-scope IPv6 address MUST be automatically allocated by the network stack when a device becomes connected to an IP backbone
  • The global-scope IPv6 address may be a global-scoped Unique Local Address (ULA)
  • A global-scope IPv6 address allows a device to communicate beyond its immediate local (link-local, mesh-local) network scope.

REQ 04. After a device reboot, its allocated IPv6 address SHOULD be the same as before reboot if the device attaches to the same IPv6 subnet after reboot.
  • This is needed to prevent that control (binding) relations that are configured with IPv6 addresses become broken after devices reboot.
  • In some cases a change of IPv6 address can’t be avoided. For these cases the OpenAIS architecture will in a later stage define best practices to cope with IPv6 address change.

REQ 05. Once a device has securely joined an IPv6 network, any future re-joining to the same network MUST be fully automatic without involving any manual configuration processes.
  • This holds even if operational network keys are changed from time to time and sleepy devices exist in the network.
  • Re-joining may happen for example after reboot of a device.

REQ 06. The network stack SHOULD support a simultaneous power-up of many devices
  • For example, due to an intentional or accidental power interruption to multiple OpenAIS luminaires.
• If the stack does not meet this requirement, a fall-back solution is to implement a random start-up timer in application code.

REQ 07. The network stack SHOULD have a means to inform a (newly joined) node about the IP address of a DNS server which the node can use for DNS queries.

• For example, if a new node needs to contact a predefined OpenAIS vendor host “myvendor.example.com” or possibly a standard LWM2M Bootstrap Server “srv.openais.org” and needs to resolve this to an IPv6 address.

• The IETF has specified standard mechanisms to distribute DNS server information and addresses, including through DHCPv6, through the Neighbour Discovery (ND) protocol, or by supporting routing of a well-known ‘DNS server’ anycast address towards the nearest DNS server.

• In a fully stand-alone networked system this requirement is not relevant, as there is no DNS server and no way to contact an external host/server.

REQ 08. For wireless networks, a network joining operation MUST be secured such that non-authorized devices can’t join the OpenAIS network. (See some special considerations for the out-of-the-box network joining noted earlier.)

• Out-of-the-box functions, which can be accessed also by non-authorized devices by definition, are considered to use communication outside of the OpenAIS operational network. E.g. by using a standard out-of-the-box unsecured network.

REQ 09. A network key MUST NOT be exposed in the clear at any point during the node’s join process.

REQ 10. For wired networks, the network joining operation MAY be secured.

• Wired connections already provide a form of limited physical security. For this reason, secure joining is optional for wired networks.

• For Ethernet wired networks in smaller installations, the typical expectation is that no security mechanisms or authentication is used on L1-L3. For larger installations, secure joining is more likely to be used for Ethernet.

4.1.4.2 Minimum PHY/MAC Requirements

The OpenAIS architecture will support the use of any well specified (i.e. fully specified, e.g. by IEEE/IETF) wired or wired physical layer that can support IPv6, either directly or via 6LoWPAN. However, such a PHY/MAC must meet the minimum requirements as specified in this section.

Latency of the total chain (‘press button’ to ‘lights on’) is one of the key parameters of a lighting system. As this performance is dependent on both the parameters of the network system and further architecture choices it is not feasible to just impose a simple speed requirement for the physical layer only. Let us assume a minimum requirement of the latency between manipulating a switch and having actual light on is < 400 ms. Whatever physical layer and further configuration are chosen, this requirement must be met.
The following aspects need to be taken into account:

- Bandwidth of the physical layer
- Number of hops for multi-hop networks (which will be influenced by both the deployment of the Control functionality as well as the network choices).
- Duty cycling delay, especially with low-power devices
- Average network load / band access timing (and possible peak band load / access timing)
- For packet loss rate PLR >0% the average packet transit time is multiplied by \((1/(1-PLR))\) and re-transmission time delays need to be accounted for.

And the following aspects need to be taken into account if they differ significantly from the values stated below:

- For security (encryption/decryption) we assume 2 ms average for encryption and 2 ms for decryption for each L2 encrypt/decrypt operation. We assume an additional 2 ms average for overall end-to-end security, in case a secure channel between the two ends has already been set up using e.g. DTLS.
- For packet processing per node (hop) we assume 2 ms.

Some examples:

1. Control allocated on a central device, two hops between sensor and controller and two more hops between controller and luminaire. We take a band-width of 250 kbps and assume no duty cycling. This gives \((4 \times 100 \times 8 /250 + 5 \times 2 + 8 \times 2 + 2) = 40.8\) ms for a packet of 100 bytes.
2. Direct communication between sensor and actuator, single hop no duty cycling. We take a band-width of 16 kbps and assume a single hop. This gives \(100 \times 8/16 + 2 \times 2 + 2 \times 2 + 2 = 60\) ms.

We have verified that the following default set of IPv6 supporting network technologies selected for OpenAIS can meet the requirements: 6LowPan/Thread mesh networks, Ethernet/PoE wired networks, and Wi-Fi wireless networks.

**REQ 01.** The PHY/MAC layer MUST support IPv6 with IPv6 multicast
- This implies a Path MTU of at least 1280 as mandated for IPv6

**REQ 02.** Link-layer (L2) security for wireless PHYs MUST be provided
- Having the properties of data confidentiality (encrypted) and data integrity (tamper-proof and replay-attack-proof)

**REQ 03.** Cryptographic security level MUST be at least equivalent to that of standard AES-128

**REQ 04.** The PHY/MAC layer MUST enable a total latency of the lighting chain (as defined in this section) of < 400 ms.

Note that for wired PHYs no link-layer security is required.

### 4.1.4.3 Recommendations for IP Network Stacks

The OpenAIS architecture does not prescribe specific IP Network Stacks (L1-L3, so including PHY/MAC choices) but rather aims to operate over any IPv6 stack. However, to operate OpenAIS systems correctly the IP Network Stack must adhere to a set of minimum requirements.

Section 4.1.4.1 has provided requirements for network join functions that a stack must provide. Minimum bandwidth and other requirements were given in Section 4.1.4.2
In this section a number of IP Network Stacks are recommended that would fit in an OpenAIS system.

For wired connectivity, Ethernet is the current recommended PHY/MAC. The IP stack used on top of Ethernet is the standard IETF IPv6 Protocol Suite stack. Standard network devices such as hubs, switches and routers are used to build the OpenAIS network or alternatively an existing IT network can be shared between OpenAIS and IT equipment.

For wireless connectivity, we recommend these two network stacks:
- Wi-Fi (802.11b/g/n/ac) for high-speed wireless networks, defined by the Wi-Fi Alliance.
- Thread, the new low-power low-cost IPv6 mesh networking technology defined by the Thread Group [Thread]. This recommendation is tentative, because at the time of writing there is not yet sufficient experience with Thread in actual applications to judge whether it can perform well for professional lighting applications.

4.2 System Design Considerations

This part gives an example that shows in more detail the functionality and interaction of the core architectural element "Control Object", by explaining its typical interface and basic functionalities, followed by some examples of how diverse modes of stacked control interact to achieve typical control requirements of a lighting system. It also explains the handling of scenes as a powerful tool that allows to realize advanced functionality without exceeding the usual bandwidth limitations.

4.2.1 Control Object General Concepts

4.2.1.1 Preamble

The control object is the functional heart of the OpenAIS lighting controls: It forms the bridge from sensor signals and user inputs to the actuator settings. Therefore, it will be available in many different flavours, allowing to cover various requirements that may evolve over time and also create some base for functional differentiation for the vendors.

The main task of the Control object is to set all the actuator values of the group it controls to the correct values, and to make sure the actuators are monitored to stay at the expected values.

4.2.1.2 Control Object Interfacing

The various connections of a control object will be classified into "5 types of interface" to ease understanding. This is a classification ONLY, it is not about implementation structures at all.

Southbound interface:
- This interface connects the Actuators under control with the Control object.
- The Actuators under control of a Control object form one group.

Northbound interface:
• This interface allows other Control objects to operate the group as a single Actuator.
• It provides a management interface that allows the stacking of Control objects.
• It provides combined Sensor data of each sensor type as if it represents a single Sensor.

Eastbound interface:
• Connects the users through an API to the Control object.
• Provides status information and a control interface for the group (as in the northbound interface) for the information of the user.
• Provides status information and a control interface for the individual Actuators and Sensors that are connected to the control object.
• Provides access to the adjustment of scene and transfer pattern settings.
• The eastbound interface provides access per session authorized by an AAA Server.

Westbound Interface:
• Connects the Sensors to the Control object.
• Triggers actions of the Control object regarding the sensor type and the Sensor group the message derives from.

Configuration Interface:
• This interface is providing access per session, using the object access mechanisms of the (LWM2M) framework.

To ease wording, Actuators are depicted as Control object internal structures (in this section) and the actual Actuators status and operation is assumed to be executed “automatically” in the field, whenever the settings change.

• Example: An "off" command is executed. This means that the internal representation of the group and the Actuators are updated and the off is sent out to (all) the Actuators in the group, and the status reports from the Actuators are checked if they correctly reflect the status required.

This is a simple example of a Control object as it might be used for the OpenAIS M33 Demonstrator. It shows the main principles of interaction between the various components the Control object consists of, but does not impose a specific Object Model by showing examples.

The main concepts used in the Control object will be shown, using events to trigger state changes and state changes that trigger events. This is only a possible way of describing it, and was chosen to make things as clear as possible, but is not a prescription of how it actually should be implemented. In an actual implementation the event path is possibly interrupted, incoming events and changed states may lead to internal state machine transfers. The actual action that is derived from a state change of the internal state machine is possibly documented at a different place.

4.2.1.3 Control Object Operation Concepts

ON <—> OFF: in operation / not operating (boolean: ON/OFF)
• There is an ON/OFF state for the full group, and an ON/OFF state for each actuator.
• State change is triggered by sensors, users or superior control objects.
• An ON/OFF state change in the full group imposes a state change at all actuator objects.

**DIM_UP <--> DIM_DOWN**: direction of the next manual dim action (boolean: dim_direction)

- This state determines the dimming direction a non-directional dim-request should take. The change of the state is triggered by the end of a non-directional dimming request (so that the next non-directional dimming request dims the opposite direction), to allow alternating dimming directions from e.g. long pressed buttons.
- On set_intensity executed with an intensity above 80%, the state is set to DIM_DOWN.
- On set_intensity executed with an intensity below 5%, the state is set to DIM_UP.

**PRESENCE <--> NO PRESENCE**: all presence detectors combined (boolean: presence)

- This state combines the possible multiple presence detector states.
- It is set to presence whenever a presence message from a presence detector is received.
- It is set to no_presence by the timeout of the presence timer, which is re-triggered at the reception of every presence or hold_presence signal from a presence detector sensor.
- The state-change to presence sets the ON/OFF to ON.
- The state-change to no_presence issues a slow_off command to the actuator objects and sets the ON/OFF to off (without executing an off command to the actuators, as the slow-off was already issued.)

The Control Object also maintains:

**ACTUATOR SETTING**: This is a vector that contains the actuator set value(s)

- The vector may differ depending on the type of actuator controlled.
- For a light point it contains e.g. intensity, colour, and other information that specify the operational setting.
- The actuator setting will be available for the group as a whole, and (with advanced control objects) also for each actuator under control.

### 4.2.2 Control Object Southbound Interface, handling Actuators

#### 4.2.2.1 Handling Light Point Control

Light point control may be used as general actuator template. Control of light points is done by the following actions: on, off, step_up, step_down, set_intensity, slow_off.

- On sets the state of the actuator to ON using the standard transfer pattern. The other actuator settings do not change, e.g. the last on intensity used will be executed, with the transfer pattern applied (e.g. smooth transfer within 500 ms)
- Off sets the state of the actuator to OFF using the standard transfer pattern. The other actuator settings do not change, e.g. the actual output is turned off but the last intensity used will be preserved as "actuator setting", with the transfer pattern applied (e.g. smooth transfer to off within 500 ms)
• Slow_off sets the state of the actuator to OFF using a specific (long duration) transfer pattern. The state transfer will be immediately, but the smooth transfer to off will change the output slowly over an interval of 50 seconds. This is used with presence detectors to dim unnoticeable to off.
• Step_up increases and step_down decreases intensity smoothly by 26% (from actual) in 1/4 sec. (This is equivalent to 12 steps in DALI, or 17 Steps in DSI) For continuous smooth dimming this command will be issued in 250 ms intervals, a full 1-100% range dimming is taking 5 sec.

4.2.2.2 Minimal Actuator Control
If all actuators are always set to the same value, the actuator control gets relatively simple: all actions are executed in parallel on the (internal) group setting and multicast to the group the actuators consists of.
• The (random) status responses from the actuators are compared to the internal group setting, and the actuator status is corrected using unicast messages if necessary.
• The individual members do not need to be known in minimal operation.

4.2.2.3 Typical Actuator Control
In normal operation the individual actuator settings may differ. Therefore, each actuator has its settings and actual operation mirrored in the Control Object. All commands will be executed using multicast commands in the actuators in parallel to the internal mirror.
• The reported status from the individual actuator is compared to the entries in the control object and corrected using unicast messaging if necessary.
• Besides the actual actuator status (output vector) additional information may be presented by the status message of the Actuator Object:
  o Information about the last valid received commanded value (it may differ from the actual output for various reasons)
  o Information about the reason of the difference between commanded and actual value (transfer pattern, physical limitations, priority conflicts regimes etc.)
• The updating mechanisms used to keep the external actuators in sync are not handled in more detail here.

4.2.3 Control Object Westbound interface handling: sensor signals
Sensors that supply signals to the westbound interface are usually of the type push button, light sensor or presence detect. Other sensor types may be supported, but are not listed here.

4.2.3.1 Push button sensors:
Push button sensors are usually of the type "momentary action switch/dim".
• A short "click" toggles the on/off state of the control object
• A longer "hold" starts a dimming sequence (using the actual dim_direction) that is stopped when a "break" signal is received. The dim_direction is toggled to enable dimming into the opposite direction with the next long hold of the button.
• Push button sensor inputs may be programmed to switch to on also with the first hold signal received.
• Push button inputs may be programmed to recall a scene with a click event.
• The dimming on long press may be disabled with some push button programming.
• Other interpretations of push button events may be used by suppliers depending on the application that needs to be supported.

4.2.3.2 Presence detector sensors:
The presence detector sensors switch the status of the control object to on with a presence signal received. There is no change action if the status has been on already.
• The presence inputs may be programmed to recall a scene if a change to status on is performed. This may be programmed dependent on other conditions to support more complex applications, depending on the vendors choice, e.g. during a fade_to_off process the switch to on is performed without a scene recall to support a "back to last operation" setting immediately after the presence detector triggered off.

4.2.3.3 Light sensors:
• Light sensors provide, depending on the type used, intensity information either from available daylight only or from the sum of day- and artificial light.
• With daylight sensors a linear control is available, using the available daylight as variable x.
• With sensors that pick up the combination of day- and artificial light a loop/back control is available, that changes the intensity to keep the sensor signal at the set value.
• Note that these control algorithms
  o May be subject of differentiation between vendors
  o May be using additional zonal information within the control group to allow differentiated control (possibly of each light point) according to specific application needs.
  o Are therefore not detailed further.

4.2.4 Control Object Eastbound interface: User control
The user control interface provides status and control of the group. (Absolute and relative settings)
• The user control provides the list of available single actuator IDs (and their type) that form the group. (With minimal southbound operation the answer will be "0" as no individual actuators are managed)
• The user control interface provides status and control of single actuators (identified by their IDs) that are part of the group.
• User interfacing is usually done by session based authorization, using the methods provided by the LWM2M framework rather than the OpenAIS-defined Peer-to-Peer interface.
• User interface provides access to automatic control parameters (enable / disable automatic control) for a time specifiable.
• User interface provides access to the group scene management (recall, set, modify) and to the scene settings of each actuator (identified by the actuator ID)

4.2.5 Control Object Northbound Interface: Stacked control
4.2.5.1 Mimicking an actuator
The Northbound interface of the control object provides controls that are identical to the controls of an actuator.
• This allows a superior control object to handle the group of actuators related to this control object together as single light point.
• As long as relative settings are used (e.g. step up/down etc.) the differences between the actuators are preserved, with absolute settings (e.g. goto 75%) all actuators are equally set to the same value.
• With scene recalls the control object will use an internal referenced local scene recall with its diverse settings rather than implement a single value (as the mimicked actual Actuator Object would do).
• The internal state reported to the superior control object is an algorithmic result, combining all actuator values to a single value that represents the local state to its best: intensities are averaged, errors and on/off accumulated, and energy data summed up.

4.2.5.2 Mimicking a sensor
The Northbound interface mimics a sensor interface that provides sensor data of the combined to its best sensors: presence and switching is accumulated, light sensors averaged, energy consumption accumulated, etc.

4.2.5.3 Stacked Control Support
Superior control objects need to control the operational state of the subordinated control objects. This is done by selecting the operational state of the control objects. Please note that this control needs regular update (in a kind of watchdog signal), as the control object will fall back to "normal operation" if the operating mode signal is not available. This ensures a system fall-back operation.

• Normal: The standard operation as outlined later. This could also be called "local operation". The local control is fully operational, a superior control object can override the actual control using the northbound interface, but has no guarantee that a sensor or User Interface override is done without prior warning or negotiation.
• Ghost: No actuator commands are sent out, all operation is kept internal. The northbound interface with the mimicked sensor and status values are provided. This ghost status may be used e.g. as a backup control object that can jump in when the operational control object fails and stops setting the ghost status.
• Remote: No actuator actions related to sensor signals received via the westbound interface are performed. The Remote state is e.g. used to allow the superior control to handle the actuators individually through its southbound interface, using its own algorithms.
• Local: Set to local, a control object will no longer execute requests coming through the northbound interface. The sensor mimicking and the control status setting remains operational. This operating mode isolates the control object from superior control objects, and may be used e.g. to support troubleshooting and error triage handling.

Please note that this list may be extended to control the internal algorithms one by one (presence, daylight compensation, circadian compensation etc.) and alike, depending on the preferences of the vendors: Details of supported features and how they are handled will be found in the product data sheets and their application notes.

4.2.5.4 Using Stacked Control (examples)
An example of simple stacked control behaviour is a corridor-follow function. The corridor lighting stays on at a minimum level, when any light is on in one of the
adjacent rooms, and goes to a higher dim level when presence in the corridor itself is detected. In this example there will be a Control Object for every room, each having an interface indicating if any light in the room is enabled or not. A “corridor” Control Object monitors these “signals” and controls the corridor lights.

In Figure 24 we see this set-up in detail. There are 5 rooms with a corridor in between. Each room has its own “local” Control Object (which e.g. may be allocated in one of the luminaires, such details are not important in this story). These local Control Objects control the settings of the luminaires in the room based on the sensors and switches present there.

On top of these 5 Control Objects which are on the same level, there is one “stacked upon” Control Object that monitors the “status” of all room Control Objects and based on that information, combined with the information of the sensors in the corridor, sets the dim level of the luminaires in the corridor.

Note that this is a simple stacking as the “stacked Control Object” only makes use of the data of the underlying Control Objects and does not execute control over their devices.

Another example of such “layered” control is a central Control Object running somewhere on an IT-device, either on-premises or in the cloud, that receives energy Demand Response requests and instructs all Control Objects in all areas (or a subset of these depending on the configuration) of the system to limit its settings e.g. to 60% of the requested dim level to save energy.

Figure 25 shows this set-up schematically. There is some IT-device in the building on which the energy Control Object runs. When requests come in from the Automatic Demand Response (ADR) system this Control Object may then send a “limit” command to all Control Objects in all rooms of the building (maybe some rooms are excluded, like meeting rooms or CEO’s office). This Control Object may also use other information about the building like time of day, occupancy of the building or current power consumption.
Note that this is a more complex use case in which the “stacked” Control function also influences the settings of the luminaires. Of course also here we have two options; the “stacked” control could directly address the actuators or do this through the Control Object(s) at the lower hierarchical level. In this case the latter solution is chosen which is less complex. Again this is an example solution, not as such a restriction of the OpenAIS architecture.

Let us now look at another scenario in which there is a stacked Control Object that overrules some behaviour initially programmed into the system. Let us look at a standard area where a number of luminaires with integrated presence sensors are positioned and a few “sensors”: several daylight sensors, a switch and a dimmer. In this situation the default “control” deployment is that each luminaire includes a Control Object that “listens” to the commissioned sensors (on/off, dimmer, local presence, daylight - if commissioned for this light) and based on this information implements a Control Object that switches the light on when there is presence, using the actual dimming value and the information of a nearby daylight sensor to determine the new dim level setting. This situation is depicted in Figure 26.
Typically, IP multicast would be used by the “sensors” (dimmer and on/off switch) to send their updates to the multiple Control Objects.

Note that there might also be some built-in fall-back behaviour in this scenario. When IP-connectivity is lost, a luminaire may revert to using its local presence sensor or even switch on to a default dim level. Note that the details of such algorithms are not in the OpenAIS reference architecture scope and those presented here are only examples.

There is no stacked control in the above example yet. However, when an independent vendor wants to add an additional Control Object for somewhat more advanced functionality (e.g. all luminaires do not only have settings based on the embedded sensors but also react to the luminaires which surround them) this could be added in the following fashion. In this new example, a separate (area) controller is added to the network to avoid (software) updates of the installed devices and on this separate controller a “room” Control Object is added. This new stacked room Control Object can now start listening to all sensor events and control all the lights based on the new algorithm, bypassing the original per-luminaire Control Objects. This is depicted in Figure 27.

This set-up requires re-commissioning and a provision in the original Control Object that it can be reprogrammed (at commissioning time) to change behaviour.

Note that, as always, other solutions to this problem are feasible. We could update all Control Objects on the luminaires with updated code that next to its initial functionality listens to the surrounding luminaires states. However, if we have a given set of luminaires which cannot be (easily) modified a stacked Control Object will do the job. As already visible in the above example we must make sure that whatever Control Object is present in a device it can be “disabled” to allow stacking when required.

These examples present real-life cases which explain the essence of stacked behaviour. Of course much more complicated set-ups are possible, such as two Control Objects in parallel modifying two actuators. However, it is not expected these will be widespread as most functionality can be handled in a rather straightforward way. OpenAIS will not dive deep in specifying complex cases. The only aspects that all Objects must follow are:
• An interface/configuration to indicate if it can accept multiple parallel requests from clients. The actual interface could be implemented by having two or more interfaces which route their functionality to a single "physical" object.

• The Control Object then has to implement conflict resolution behaviour that may be specified by the supplier. A default conflict resolving behaviour: e.g. "last come last served".

• Potentially more complex or specific conflict resolving may be implemented in such a controller, which then has to be specified and can be specific for a supplier/luminaire.

4.2.5.5 Using Stacked Control: example sequence diagrams

We will now show the sequence diagrams for the three presented use cases to further clarify the behaviour.

1. Corridor follow

![Sequence diagram for Corridor follow](image)

2. Building level control
3. Overriding behaviour

4.2.6 Handling Scenes

When talking of "Scene" technically a 'Scene Object instance' is referenced. This is to cover the idea that a Scene Object is instantiated with every scene that is made available.

4.2.6.1 Definition

A scene is a set of actuator settings that together create a specific effect (In lighting: a specific (colored) light distribution creating a lighting effect, e.g. in a presentation room when switching to "presentation setting").

The scenes are an integral part of the Control Object, all scenes apply to the group of Actuators the Control object controls. Each Scene object instance manages one scene, and only one scene is active at a single time.

4.2.6.2 Working with scenes

Scenes can be activated by a single action, often expressed as "recall a scene", e.g. by a simple press of a button or a single command from a user interface. The scene settings are realized after the action either "at once, in sync" or with a smooth
transition, the transition starting at once and ending in sync as specified by the transition settings.

Scenes are an ideal measure to reduce the peak bandwidth load: a single multicast command is able to trigger a complex parallel execution at many actuators, creating a nice optical effect, based on cached parameters in the actuators that have been prepared (long time) before the execution is requested.

- A scene definition includes the (usually actuator specific) settings and may include a transition pattern. Transition patterns may also be specific per actuator.
- To support scene settings and transitions in restricted communication environments a multicast based execution with some actuator caching to support multicast requests is required: The scene value and the transition pattern for this scene will be cached in the actuator (under control of the control object) and called into action by a multicast request issued.
  - If the transfer pattern is uniform for the group, it may be part of the multicast call. If it is different for the members it needs to be cached in the devices and called into operation as part of the request.
- Scene actuator values may be given by explicit value or by (algorithmic) reference. (Naming: scene settings by value are named "standard scenes" and scenes using referenced values are part of an "extended scene") The extended scene object instance provides the parameters needed to calculate the actuator values depending on e.g. time or sensor value(s).
  - Example given: The extended scene value definition is defined e.g. by using the linear equation, \( y(x) = mx + b \), by the parameters \( m \) and \( b \) for attenuation and offset respectively. The actual actuator value is calculated at the time of the scene activation using an appropriate sensor value as "\( x \). Please note that the linear equation example is just one of the possibilities for a referenced value definition, different equations for different \( x \) or non-linear equations may also be applied.
- Extended scene definitions that define the value by functional reference are also referenced as "program" (e.g. when related to sensor values like available daylight) or as "show" (e.g. when related to a time clock or time-of-day value).
- In areas where the groups in one room have different scenes active the total effect for the room is a kind of combined scene, combining the effects of the group scenes acting at the same time; this "combined scene" can be seen as a scene by itself that is defined on the level of the room control object, using the group scenes as "values". Therefore also higher level control objects handle scenes, using the group scenes provided by the group control objects instead of the actuator values.

### 4.2.6.3 Scene Object Basics

Any scene object instance is identified by its ID (the scene-ID), which is defined at the control object and covering the group. (The object instance will be called "a scene" usually). The scene-ID is unique only together with the application group-ID.

- A scene object instance with its ID consists of the set of settings of the covered actuators, each actuator (identified by its ID) having its own entry. (NULL entries for some actuators may be acceptable in some cases, pointing to a "do not change on a scene recall" setting)
- For a given set of actuators multiple instances of the scene object may exist, providing the different scenes for different application purposes.
- In addition to the scene settings the scene may contain parameters that
• specify the transition from the actual setting (smooth changeover, immediate action etc.)
• specify exclusion rules if any (in some applications some scene settings or transitions may be applied "only if": this could be seen as part of the scene, but also as part of the control object logic, but anyhow it needs to be connected to the scene this or that way.)

• These scene settings will be executed at a control object that in turn organizes the appropriate execution of the transition in the actuators, by using appropriate caching of parameters in the actuators to allow for synchronized scene change operations through multicast requests.
• If no explicit transition specification is part of the scene object instance, the predefined default transition will prevail.
• In the definition of the operation of the cached values, care needs to be taken that the solution supports the self-similarity that allows to use control objects with their scenes as if they were "actuators". This might end in a "value reference" used instead of an actual value definition at the actuator level, where in a control object the value reference is pointing to a (local) scene object instance.

**Setting a Scene / configuring a Scene Object:**

• Of course the actuator settings may be stored into the scene object instance by some kind of commissioning tool, e.g. retrieved from a pre-programmed database, or controlled through other (cloud based) services.
• Due to the nature of light a possibility to "set what you see" as a scene (object instance) by some user action is something that should be considered. That is where the common terms "to store" and to "recall" a scene are coming from.

**Extended Scene Objects:**

• The extended scene objects replace the explicit actuator value by the parameters needed to calculate the value, and a pointer to the algorithm that is to be applied, together with the parameters that guide the calculation. Parameters will be e.g.: The sensor or timer value to be used together with the parameters that control the algorithmic definition of the output value. (In case of linear equation, y = m.x + b, this would be m and b. Note that a fixed value is identical to the parameter setting m = 0 and giving b the respective actuator setting.)
• Extended scene objects may cover complex structures like guided circadian settings, attractive light shows etc. in a very generic and easy to use way.

**4.2.6.4 Handling restrictions: Actuator caching of Scene Object structures**

• To set a scene for a large group of actuators in a restricted environment (restricted bandwidth and channel access) may take a substantial amount of time, as the series of "goto this output now" messages pile up at the moment the scene is "recalled".
• To avoid this the scene object values are cached in the actuators, each actuator holding a local copy of his relevant settings of the scene object. These local copies need of course to stay in sync with the scene object itself (that is always bound to the control object). (Note: In contrary to this some heritage systems define the actuator to hold the only / the leading copy of the settings information)
• Doing so the recall of the scene is possible by multicasting the recall request to all the group the control object controls at once: <recall scene><scene-ID>: The actuators receiving this request will use their cached values to activate their respective settings, resulting in an "at once" creation of the whole group.
• With extended scenes two ways of executing the algorithmic reference may be considered:
  o For scenes with relative fast change of values and relative slow communication: cache the algorithmic execution in the actuator, and change the scene request to <recall scene><scene-ID><x-value> where the payload "x-value" is the input to the algorithm cached in the actuator.
  o For scenes with relative slow change of values: Regularly update the cached scene settings whenever they change substantially, to prepare for immediate action when the scene is called.

4.2.6.5 Scene handling in the control hierarchy:
• Scene handling with operational local group objects (controlling the control)
  o A higher hierarchy control object will control other control objects instead of actuators, but looks at the other controls objects as if they were actuators.
  o To support this best, the "actuator values" are referenced in a first step. This allows to use a local scene as "value" when dealing with a control object, or a (referenced) value at the actuator.
  o Of course also control objects can use "values" and set the actuators to a common value on a scene recall. The referencing step in between allows to use both choices with a uniform object model.
• Scene Handling with hibernated (stacked) local group objects (direct control)
  o Once the more local control object is hibernated, the control object now in command uses the standard scene handling (and caching) mechanisms, defining its own scene reference cache at the actuators.

4.2.6.6 Additional Considerations:
A generalized vision of the scene object caching structure at the actuator and the subordinate control could look like this:

(Superior) Control Object:
• Scene - ID with its group member settings
• group member cache ID
• group member cache settings

Actuator and (local Control) Object:
• Scene cache ID (= local scene ID for the local control object) with
  • (actuator:) cache setting
  • (control object:) group member cache ID and group member settings (see above)

Portable scenes:
Some OpenAIS requirements talk about "portable personal preferences" and "portable scene settings". The scene object detailed here is not designed to travel with the person (or alike, as required by the user-requirements). The scene object instance is
permanently related to the control object and its group, and the actuator settings forming the scene in this context are strictly related to the actuators (by their ID’s).

- The portability requirements will be handled by additional high level software (apps), that changes the scene object settings (or add specific scene definitions) according to the requirements of portable scenes.
- To achieve portability some meta-data that identify the specific task (or relative placing) of the actuators in the room will be necessary, to understand the relation of the scene settings and e.g. identify the “door side” placed light points.
- Such software enabling portability will most likely be cloud or central management based, and will doing its task by changing the group scene settings appropriate.

4.3 Deployment

This section provides more insight into how and where the architectural components may actually be used (deployed) to achieve a versatile but still simple to encompass product portfolio that covers many needs of actually realized installation.

4.3.1 Introduction

As explained in Section 4.2 there is a lot of flexibility, especially in the allocation and distribution of Control function(s). In combination with the stacked control capabilities described in Section 4.2.5.3 this leads to so many potential combinations in the reference architecture that some guidance is needed, which will be given in this section. Note that this section is descriptive and not prescriptive in nature, the full flexibility of the reference architecture stands as defined.

For any given implementation of the reference architecture in a real system a full system design must be made. In this design, choices have to be made on the application and applicability of all potential variations that the reference architecture allows. Likely, the control approach for the system will be selected and constrained in such a system design.

This section is only intended to illustrate some possibilities of the architecture, as input to a system design process. We will start with some “simple” and “extreme” examples in the next subsection (Section 4.3.2.1- 4.3.2.5), followed by a more in-depth discussion of a realistic example in the subsequent subsection (Section 4.3.2.5).

Note that in the physical view we identified the following physical devices:

- Luminaires, device with one or more (LED) drivers and zero or more sensor devices connected to it.
- Area controllers, device with only networking capabilities.
- Standalone sensors, devices with only one or more sensors connected to it. (They may be either battery or AC powered.)

As already described, two out of the three identified types of functions two are linked to physical hardware directly. A sensor function needs sensor hardware (e.g. light sensor, or hardware-button), while an actuator needs a light source. This means that Actuator functions are deployed to luminaires and Sensor functions will run on standalone sensors and on luminaires that have sensor devices included.
We have seen an example architecture (without the Controller functionality included yet) depicted in Figure 18 in Section 3.4.4. There the physical devices are connected to a restricted network (like 6LoWPAN) and the potential area and building controllers are connected to the LAN of the building.

4.3.2 Examples

We will start with a simple example of deployment to a standard system configuration that we have introduced in Figure 18. The devices here are indicated in schematic way to show the deployment. The two luminaires in the following example system also have a presence sensor integrated. We also see that the Sensor and Actuator functions are “naturally” deployed. For the Control function there is no direct link to hardware (it is software only) and the Control function can therefore be allocated anywhere in the system.

4.3.2.1 Control function deployed to every luminaire

In this first example, a Control function has been deployed to every luminaire where it directly converts the signals from the sensor embedded in the device to the settings of the actuator in this same device. On top of that there is a presence sensor in the room that can switch all lights on and off. Of course this is only a partial (and simplified) behaviour of an entire lighting system. However, it does provide a good insight into the control flow going on in the system. Note that only one control flow is indicated with arrows in Figure 31, for clarity.

This set-up has a number of characteristics:

- As there is no central Control function, this system is very resilient against disturbances. Each luminaire operates independently, only the on/off switch is a single point of failure.
- Since no central Control function exists, there is no control that goes beyond the scope of a single luminaire.

4.3.2.2 Control function deployed to sensors

Of course the allocation can also be fully reversed by allocating a Control function(s) to the Sensor device. As all luminaires have sensors integrated this does change the picture only superficially. However it introduces a stacked control because the Control function in the switch (sensor) now will also control the Control functions in all the luminaires, where the master Control function in this case will also get its inputs from the local luminaire sensors. This is shown in Figure 32.
4.3.2.3 Control function deployed to every sensor

In a situation where there are no sensors in the luminaires the picture changes considerably which is shown in Figure 33.
This works well when there is only one sensor involved in the control of the system. When there are more sensors involved in this configuration which influence the lighting settings, they either have to communicate or we re-introduce Control functions in the luminaires which implement the algorithm to arbitrate between parallel requests.

### 4.3.2.4 Control function deployed to Area Controller

We can also allocate the entire Control function for an area on a single controller device. This may be an Area controller but it can be as well one single luminaire that acts as “controller” for the area. In this alternative example “all” sensors, even those in the luminaires, report their status to the Control function in the Area controller, which on its turn controls all Actuators. In Figure 34 we see this set-up. We have only drawn a part of the control messages for clarity not to crowd the picture too much with many arrows. The characteristics of this solution are:

- All information in one central Control function that can implement integrated behaviour for the entire area (like putting adjacent luminaires to half the light level e.g. when one above a desk luminaire is lit).
- As the central Control function has the input of all sensors it can implement integrated behaviour and advanced behaviour. As an example it may track movement through the room based on presence data and adjust behaviour accordingly.
- This solution has clearly one single point of failure, when the central Control function fails all lighting functionality dies.
- Also the network load is significantly higher, as all messages have to travel back and forth to the central Control function. This may also introduce performance and latency issues.
- Note that allocating the central Control function on one of the luminaires has another disadvantage: not all luminaires would be identical, which is a complication for installation, commissioning and repair.
Of course there can also be a Control function allocated to an even higher level in the control hierarchy, for example on a building server. The various characteristics of the system that were listed above are then further strengthened.

The above examples already show a number of the main trade-offs that have to be made when deploying Control function(s), specifically trade-offs between the following properties:

- Scope of control
- Performance/latency
- Resilience against failures.

The optimum is very much dependent on the actual design of the system; the OpenAIS architecture supports flexible deployment and stacked control to facilitate any necessary system set-up.
4.3.2.5 A stacked control example

In Figure 35 we see a building with two areas where stacked control is introduced, following the concept that control is allocated as low in the hierarchy as possible.

Each luminaire has a Control function that reacts to all sensors relevant to it. It listens to its local sensor and also to stand-alone daylight sensors, presence sensors and switches. The Control function in the luminaire can decide on its own Actuator setting only based on this information. It does not need information from other luminaires in the area to determine its settings.

There is stacked control also in this case, because a higher-level Control function is allocated on the Area controller. This Control function implements behaviour like setting adjacent luminaires, i.e. when one is lit its neighbours are lit up too and tracking person movements through the area. This Control function receives all sensor information it needs and controls the local luminaire controllers in a stacked way.

On top of that there might be even higher-level behaviour like “corridor follow”. As a corridor (not shown in this picture) must be lit during presence of someone in one of the areas (rooms for this discussion) there is a controller above the “area level” that monitors all “areas” and sets the corridor controller accordingly. Also Automatic Demand Response functions can be part of this “building level” Control function. Following this approach, resilience is guaranteed as failure of a controller still leaves all smaller scope behaviour operational.
4.4 Installation and Commissioning workflows

This part looks into the installation and commissioning process. The efficiency and reliability of this process determines the acceptability of the system(s) at the electrical contractor, being one of the major deficits of today's lighting control offer. A whole set of workflows is depicted as the external preconditions vary a lot from site to site.

4.4.1 Introduction

Before a lighting controls system can be operated, some steps need to be taken. The devices need to be mounted, connected, and integrated into the controls structure. All these steps are accompanied by commercial activities (bidding, offering, ordering, billing, etc.), that substantially vary regarding system size and type of contract. The commercial activities will be not elaborated further; instead the technical workflow from manufacturing to operation of the devices is described in this section.

The actual functionality of the system, the steps to determine the logical grouping of luminaries and other devices etc. is not part of this section. Please refer to Section 0 for more details on possible layouts. Once the layout is clear (this is part of the preparation process that is usually handled by consultants), the actual grouping and binding is managed by a tool (Commissioning Tool) that uses device management interfaces to execute this predefined structure. Device management sets the resources of the devices and the Object instances that are used for binding and addressing. The details will be found in the device and Object instance API definitions ([OpenAIS_D2.4]).

This section elaborates how the devices can be identified/located after mounting, to do the parametrization. To locate, reliably identify, and finally administer some 10.000 lighting devices installed in a larger office building is not trivial. A lot of the “ease of life” requirements are connected to this, as residual misidentifications errors are rather difficult to handle. Therefore this section concentrates on how a light-point can be identified/located after mounting, and how wrongly located or missing devices in lists can be identified.

Please note that before any system integration is done on site, devices may be integrated into luminaries. The product that is mounted inside a luminary is called “OEM product”. We use the term “device” for the final product that is mounted and connected on site. Wherever necessary to distinguish between the actual unique device and the product, we use the term “article” to avoid misinterpretation. (The article number is used to buy another device that replaces the broken one).

Manufacturing:

• At manufacturing time, the device is pre-programmed for the out-of-the-box functionality and receives (at least) two identifiers: an OEM_Product_ID and a unique Device_ID (MAC).
• Devices are transferred to the luminaire manufacturer for integration into the luminaire if the devices are integrated into luminaries.

Implementation into the luminaire:
• At luminaire implementation time, the device receives the Final_Product_ID, that fully specifies the article, and possibly some pre-programmed parameter settings that support the application of that specific luminaire best.

4.4.2 Commissioning Flow(s)

The commissioning workflows depend very much on the timing of information (e.g. the availability of already elaborated grouping information and the grouping document) together with the available capabilities of the executing workforce. (Specialists are rare and often not available when needed.)

This section shows a set of 3 basic workflows, namely pre-programming, documenting and “direct install” that may be supported by vendors. The direct install workflow needs to be supported by all vendors, as it allows recovery from mistakes and triage at a late stage.

A main goal of OpenAIS is to use as little special knowledge as possible on site, i.e. if possible, the installer should be the only person that needs to be on site. After the installation job is done all other commissioning work, including the programming of the devices, should be available as much as possible “through the wire” without physical presence.

4.4.2.1 Pre-programming workflow

In the previous section we have seen the pre-programming of functionality and IDs. Here the workflow using pre-programmed devices is described as follows:

- Before mounting, the following site documentation needs to be available:
  - Location Identification (IDs that can be processed by the installers)
  - All bindings and groupings
  - All parametrization
  - System credentials and access regime

- The devices are pre-programmed for the locations in which they will be mounted. Two flavours are possible.
  - The pre-programming is done in factory, and the devices (luminaires, sensors etc.) get labelled with the location where they need to be placed.
  - The programming of the device is done according to the mounting location at mounting time. This is performed by the commissioner on site. Various techniques may be used, e.g. NFC programming, Configuration plugs, etc. The installer needs to be supplied with pre-labelled configuration plugs or with an NFC programmer that does the complete configuration job with only the location entered.

- Devices are finally mounted and connected, and do their intended work immediately.

The pre-programming path works quickly and reliably, but comes with the disadvantage that it is not simple to cover site issues such as:

- Last-minute changed layouts,
- Mistakes like mounting of luminaries on (systematically) wrong or (even worse) random locations due to communication troubles on site,
- Replacement of broken or missing devices.

The pre-programming path works without any out-of-the-box functionality, as the devices are already pre-programmed for their target operation.
4.4.2.2 Documenting workflow

There are some flavours of documenting the location of a specific device, which will be shown in this section. All of them follow the same goal: To establish a relationship between the actual location of the device (and the location ID) and the device (UU)ID. Once this relation is established, an offsite part of commissioning may be performed, most likely by some algorithm that executes the “Grouping and Controls’ structural” information given by the consultants.

- Before mounting devices, the following site documentation needs to be prepared:
  - Location Identification Scheme (prepared in a way that the mounting and connection personnel can handle it.).

- Devices locations are documented when mounted.
  - Labels containing device IDs are detached from the device / luminaire and attached to planned location where the device is mounted. This creates the relationship between device-ID and location.
  - Labels of devices are scanned together with a location tag on the plan. This creates the relationship between the device-ID and the location.
  - Once indoor positioning is available devices may sense their location and provide the relationship between device ID and location electronically.

- Device installation is verified / possible handover to commissioners.
  - All devices are checked whether they perform the out-of-the-box functionality correctly.
  - A list of all connected devices (and the luminaire articles they are part of) is compared to the contract list / the BIM database.

- Before commissioning the site the Controls structure needs to be prepared
  - Grouping and Binding
  - System credentials and access regime
  - Controls Parameterization

- Commissioning is performed
  - The device - ID relationship to the location is used to program the devices with the correct set of groupings, bindings and parameters after establishing the network connections.

The documenting flow works fine as long as the location scheme and the (final electronic) documentation of the relation of the device ID to the location is working fine. If indoor positioning is available this provides a perfect and very flexible way of commissioning a system with only little site presence.

4.4.2.3 Direct install workflow

- Before mounting and connecting no specific site documentation needs to be available.
- Devices are mounted and connected.
- Device installation is verified / possible handover to commissioners.
  - All devices are checked whether they perform the out-of-the-box functionality well
  - A list of all connected devices (and the luminaire articles they are part of) is compared with the contract list / the BIM database.
- Before commissioning the following site documentation need to be available:
  - Grouping and binding
  - Parametrization
System credentials and access regime

(A location ID scheme that allows to identify the location of each device is helpful, but not mandatory)

- Devices are selected and commissioning is performed.
  - Selection of already mounted devices is done by (human) interaction:
    - Sensors are selected by manual interaction, e.g. pressing a button or alike, catching the signal, and documenting its location (and possibly in the same instance doing the programming of the selected sensor with the prepared operating parameters is performed).
    - Actuators are selected by identifying the location of the (visible) actuation. Two concepts may be used:
      - Binary search: (advisable if only a small part of the devices are in sight): on a specific location a device is selected by actuating half the devices, checking which half the device is in, and repeating this until the last decision selects the single device (and possibly in the same instance the programming of the actuator with the prepared operating parameters is performed).
      - Linear search: (advisable if a large fraction the actuators are in sight): out of a set of actuators (e.g. link-local) only one is actuated. If this one is in sight, its location is documented (and possibly in the same instance the programming of the actuator with the prepared operating parameters is performed).
  - Selection of already mounted devices is done by distant reading the ID electronically:
    - Light points may modulate their device ID in the emitted light, other devices may have distant-readable QR, NFC or alike, or may be asked to provide for a short while a pulsed status LED that modulates the device ID, and is read out this way, and its location is documented (and possibly in the same instance the programming of the device with the prepared operating parameters is performed).
    - This is repeated until all devices are commissioned.

The direct install requires more effort, as the mapping of the devices to the location needs to be done manually. On the other hand this is the most flexible approach, and the one that requires the least competence and education of installers. In case the other ways do not deliver correct results this workflow is always available and very robust.

4.4.2.4 Functional verification and handover to operation

- Check for completeness
  - All Final_product_IDs of commissioned devices are retrieved and compared to the following:
    - The BIM database if available,
    - Other site documentation.
  - The Final_product_ID of devices that are in the out-of-the-box operation are retrieved and their location is checked. (There should not be any device left that is in the out-of-the-box operation!)
- Check for local operation (correct binding)
- Room by room (and in large rooms group by group) the local operation of the lights is checked using the local switches and local presence detectors: Check if all luminaries in that group go to ON and OFF. In parallel to that the status change of the rooms and groups is logged and the log is compared to the sequence of the walk-through.

4.4.3 Further commissioning aspects

This part elaborates the following commissioning aspects which are not covered earlier: i) How can adjustments be made in operation or late stages of commissioning? ii) How do cloud services come in? iii) How are mobile devices configured? iv) What kind of access will “third parties” (companies that have not been part of the fulfilment process in the first place) get?

4.4.3.1 Additional adjustments

With established correlation of the device ID, IP address, and Object IDs to the location and function of the devices, any additional adjustment (parameter settings, daylight harvesting algorithms, presence timing considerations, etc.) can be made online. These adjustments can be made with appropriate access rights.

4.4.3.2 Cloud services

To enable cloud services, the devices need to be connected to the cloud service. This can be achieved by configuring the standard binding mechanisms to (also) operate with the cloud service. The security encryption methods and keys and the details of the binding delivered by the cloud service are out of the scope of OpenAIS. The access rights need to be at least of level 4 “organize” to configure such connections.

4.4.3.3 Mobile access

Access from mobile devices can be twofold: Devices that become part of the system permanently, and devices that get temporary access to a part of the system.

- Permanent devices: These are configured in the same way as any OpenAIS device is configured: It is permanently bound to other devices that get access to a part of the system binding. This way apps that reside on that device can work directly with the system. To achieve this, access level 4 “organize” rights need to be available.

- Temporary devices: Users of temporary devices apply for access to the system using an authentication server. After operational access is granted the mobile device is able to interact with the (westbound) API provided by Control functions (and data collector functions). There is no direct connection to sensors or actuators from mobile devices. The access rights of the temporary devices is organized through the authentication server of the system.

4.4.3.4 Third party access and interaction

- Third party Control functions will simply be commissioned to operate with the devices and their interfaces, and will utilize stacking control to the already existing Control Objects.

- Third party software may be applied to devices only if it is accepted by the device software (electronic signature of the hardware vendor necessary) and allowed by site management (software update access right level needed).
4.4.4 Other Considerations

4.4.4.1 Security and site credentials
All content carrying OpenAIS communication is encrypted to ensure sufficient privacy. All access to OpenAIS devices needs sufficient authorization.

When starting up the commissioning, the devices run through the bootstrapping process and get an initial secured connection to the device management server (see Section 3.6.3 for further reference). This server is either part of the system itself, or part of the commissioning tool. (If the system already provides a device management server, the commissioning tool is advised not to use the local one.)

4.4.4.2 Using credentials
Access to the device management server, that handles all parametrization of devices, needs authentication. Depending on the authentication an access level to the devices will be granted. For commissioning a minimum level 4 access is mandatory. (See Section 3.6.4 for further details)

Please note that commissioning activities in larger systems may run in parallel, and due to the connected world ideally a single authorization server and a single device management server that are already permanently part of the system support this process. If this is not possible, the device management server and authorization server can be separated entities (e.g. residing in the commissioning tool), and be merged at a later state. (The device management keys need to be adapted for the new setup.)

When using the pre-programming path of commissioning the later system device management server relation should be known and set up in the preparation process. If this is not possible the device management keys from the pre-programming tool need to be shifted to the central device management server.

There may well be multiple device management servers in a system, e.g. one per floor, if this is advisable. Devices may also be programmed to accept parametrization from more than one device management server. (The primary device management server can authenticate other device management servers).

Without an authorized device management server that is known to the device no higher level access to a device is possible. (Operation does not rely on the presence of a device management server)

It is good practice to use higher access levels only when needed. The commissioning engineers therefore should have regular level 4 access only.

Operational protection:
After commissioning all access will be protected by the local system credentials (site specific), and all data transport will be protected by unique local keys.

Recovering from lost keys:
If device management keys are lost (either on the side of the device(s) or by losing the device management server), a device- and a manufacturer specific ways to reset the device or to grant a new access needs to be followed.
4.4.4.3 Commissioning tools

Commissioning tools may be vendor specific to support specific applications best or in a unique way. However, commissioning is done using the standardized device management interface available in all devices. Therefore all tools will be able to commission devices from all vendors.

Future systems might have no or only little need for physical presence during commissioning. OpenAIS commissioning does not really require physical presence for commissioning, but some steps rely on physical presence today due to lacking technology. Once the mounting and connecting workflow creates a reliable relationship between location-ID and device-ID, and all the databases are accurate and tended well, a fully automated commissioning will be available.

4.5 Integration with external components

This part elaborates how the flexibility for the various embedding and connection requirements throughout the life cycle of a lighting control system may be handled: User connections e.g. are of course key, but security and integrity of the system is also key. This part also elaborates how the trade-off can be managed.

4.5.1 Ad Hoc Devices

4.5.1.1 Introduction

Devices (even mobile devices like smartphones or tablets) which are permanently part of the OpenAIS system are commissioned after installation. However, there is also need for so called ad hoc devices, usually mobile devices which are coupled temporarily to the lighting system. The most common use case for this is the use of mobile devices for personal control of the lights in a room.

In a personal control situation, a user enters a room (or area) with a personal mobile device which is not normally part of the OpenAIS lighting network in the building. Through that mobile device the user can obtain control over a (pre-)defined part of the functionality of the lighting system, usually restricted to an area like a room. Standard functionality that can be controlled in this way is dimming the lights to adjust to personal tastes and needs. There are various elements in a personal control solution: localization, authorization, network joining, the restricted lifetime of user control (when he leaves the room), available Controls and UI and potential conflicts with other devices.

There are also situations where the ad hoc devices are used, notably in the commissioning use case, where an engineer attaches the device to the system to start commissioning. Although these use cases differ there is a lot of commonality in the joining of the ad hoc devices, which has been treated in this section.

4.5.1.2 Physical network joining

It is not to be expected that standard mobile devices will be capable of joining an OpenAIS RF-network directly. The default proposed wireless connectivity technique for OpenAIS (6LoWPAN) is not available on mobile devices. We do not expect this to change in the near future. Wired solutions like PoE are clearly fully inaccessible. We envisage two solutions to communicate to the system:
• Wi-Fi: The mobile device could login on the Wi-Fi network of the building. Of course this assumes that there is such a network. Although this can be seen as an easy and standard solution it is not hassle free. Especially for infrequent visitors it is not always easy to access a network in a building. Network keys are normally needed and connectivity problems are quite common. Nevertheless this is one of the options that can be used in OpenAIS.

• Using the telecom network of the mobile device. A mobile phone will usually have a (IP-)network connection over 3G which it may use to contact the login page of the building. This removes the problem of Wi-Fi network joining but of course relies on availability of a telecom/IP connection in the building. Users without ubiquitous internet access cannot be served in this way. Also tablets without a 3G-connection are excluded by this method.

It is not necessary for OpenAIS to prescribe the use of any of these methods. The OpenAIS solution only expects that the mobile device is capable of creating an IP-link to a server containing authorization/login service.

Note that this implies that personal control requires a server with an IP-connection. This does not necessarily mean that the system needs to be connected to internet. It would be possible to have wireless access points and a server on site fulfilling the role.

4.5.1.3 Actual joining and leaving the network

As indicated above all communication will go through a standard IT-infrastructure (or telecom infrastructure if the IT is not yet available) and a back-end server. Whenever a user enters a room with a mobile device in the building for the first time and he/she wants to control the lights and activates an app (“light control”) or loads a “webpage”. In the remainder we will only discuss the “app”; a web-interface follows a similar flow. Note that the user will have to download this app (or configure the webpage) for first usage. This “app” will be somewhat specific as the flow depends on the authorisation and identification policies in a building.

This app contacts the “back-end server” and in coordination with the back-end takes care of the authorisation of this user (for the first time). This may include filing in just a user and password but it may also integrate single sign-on for a user known in the building (an employee of the company), so that in a company various applications can be used with the same login credentials. For commissioning a higher level of authorization is needed of course.

For personal control a second important step is localization, this may be more or less automatically with beacons or coded light or the user may be guided to scan e.g. a code in the room. After the “app” has sent this information to the “back-end”, the back-end authorizes the user based on user, location and status information for the lights in his area and presents the user its interface.

The “log-off” of the user is a point of concern especially in the personal control use case. There clearly must be a timeout, avoiding users to control the lights in the office when they have returned home. After this time the app needs to re-initiate the connection. The setting of the timeout has to be configured in real operation. Also when the user walks into another room and reopens the app, the app should always check the location again. If this is not identifiable or changed the app should guide the user again to identify the location and connect to the back-end again for authorization and getting access to the right user interface/area.
Note that there will always be situations to get access in one room, quickly move to another room and then still control the original room. Only with a very solid check-in/check-out this could be avoided.

### 4.5.1.4 Authorization

The authorization service is a “web-based” IP-accessible service that grants (or refuses) access to an identified mobile device for a certain location/set of devices which will vary from site to site and is out of scope of this document. This authorization service will have to rely on a policy which will vary from site to site. Who is allowed to log in: only known devices/users, the use of passwords or not, potential single sign in requirements, only for own personnel or free access for everybody, it is all up to the building owner to decide. The same applies to defining what parts of the system users can access. Furthermore one probably wants to allow only one personal control device per area at the time. Finally this policy must decide on the lifetime of access, re-entry of regular users, etc. All of this is not defined in the OpenAIS architecture. However when the authorization service provides the mobile phone with the credentials needed for the control and sets a lifetime for these, the interaction with the OpenAIS system starts.

### 4.5.2 Personal Control

#### 4.5.2.1 Localization

Ad hoc devices are never allowed to access a full lighting network. Access is always restricted to the devices in an area or room. Some devices may not be controllable at all by personal control like e.g. the devices in a corridor. Therefore the localization function is an essential part of a personal control system. The mobile device has to obtain information about its location in one way or another and communicate this to the “login-page” of the system. The OpenAIS architecture does not prescribe the localization mechanism as this may be viewed as a differentiating element or even standard elements of a mobile phone may be used. Some examples which can be found in practice are:

- Use of coded light to determine the luminaire(s) closest to the mobile phone
- QR-codes in the room which can be scanned when entering.
- (i)beacons

#### 4.5.2.2 Controls

A mobile device can only use already pre-programmed control functionality in the system. It is not expected that a Control Object instance will run on the mobile device. An additional Control Object instance requires development and commissioning of the system and is specific to any room/area/building. As such there is no use in developing a Control Object instance that runs on a mobile phone although it would be technically feasible (see previous section for some commissioning aspects here).

So there will an existing and already commissioned Control Object instance in the system that will provide the functionality offered to the mobile devices. The ad-hoc device that is authorized uses the API of the area Control Object instance to control the lights of that area. So there will be a one-to-one relation between the mobile phone and the Control Object instance it uses. All multicast issues and related security will be treated within the lighting network and are out of scope for the mobile device. The mobile device only sets up a secure one-to-one communication with the Control Object instance of which it has received an address at login. This greatly simplifies communication and security. After localization and authorization one simply gets the UI
presented through which one can send commands to the Control Object instance like up and down.

The Control Object API may (depending on the vendor) provide APIs for all light points the Control has in its group. In such a case the user of the mobile device will be allowed to select which device he wants to control through the UI, e.g. by pressing on a button, or icon on a small map that is displayed on his phone.

4.5.2.3 User Interface
Personal control on the mobile device now consists mainly of an UI to invoke the control interface(s) described in the previous section. We do have two options to realize this UI:

- Build an app that provides the UI and communicates with the Control function (using the keys supplied). The major drawback is that building and maintaining apps is a quite costly business. There are multiple environments (Apple, Android, etc.) that need to be supported and every introduced phone may require some testing and development work.
- Use a generic UI like HTML5 and use the phone only as a display. Whenever someone logs into the system and is authorized and localized he receives the link to the related UI that can be used to connect to and control its area. This UI is part of the system and has already been developed in conjunction with the Control function it controls.

The OpenAIS architecture does not mandate one of these solutions. Both are allowed in the design for a system.

4.5.2.4 Extensions to the system
User interfaces may be easily provided by the system vendors for the control devices they deliver. Whenever someone develops new functionality (like a new Control function) to be loaded in an OpenAIS system he/she has to decide if this functionality is controllable. If so a UI has to be provided and installed with the Control functionality.

As only a minimum API for the Control will be mandatory the UI’s may change with specific controller features, but always can rely on the minimum functionality, no matter what kind of device mix is combined.

4.5.3 Integration into Building Automation Systems

4.5.3.1 Introduction
Building Automation is a complex trade in itself, with diverse and complex tasks, and a substantial variety of system topologies and architectures used by the vendors. The main concepts used are:

- PLC-based control (as of today approximately 40% of the building automation market, diverse variants and programming languages)
- BACnet-based control (as of today approximately 60% of the building automation market)

It has to be noted that many (especially larger) Building Automation Systems are heterogeneous; they often consist of a mixture of diverse technologies to adapt to some requirements.

- The field level of BACnet type of Automation Control often consists of various communication and controls systems, like “KNX”, “LON”, “MODBUS”, or “DALI” bus systems (there are some 30 different such systems in regular use),
whereas the management level often uses BACnet architecture based concepts, in addition to diverse (SQL) database founded solutions.

Given the diversity of Building Automation and the freedom for vendors to create OpenAIS compatible systems there is no “single path of integration” mandated. This section will show the concepts available within the OpenAIS (reference architecture) that may be used for such integration.

4.5.3.2 Device-level integration

There are diverse HVAC/building automation strategies available. This part focuses on the integration of single devices into the building management system. The example use case for this is: A lighting presence detector should be re-used by HVAC automation to control the airflow in a room. The integration warning: Lighting needs “immediate on” with a very short signal integration time, whereas HVAC should respond to persistent presence only, so the signal integration time should be long enough. (Airflow needs not to change on a short visit payed to a room to e.g. fetch a paper, but should support longer presence in a room). Within the OpenAIS architecture the following solution is foreseen:

OpenAIS sensor devices may be equipped with multiple bindings for a single resource. This means, the sensor can send its status change information to multiple receivers, with the following options:

1. The Building Automation Control function is commissioned to be part of the multicast group the sensor reports to (same sensor Object instance, including all the security involved).
2. The sensor is commissioned (using an additional Object Instance) to send its sensor event to a HVAC / building automation resource compatible IoT binding and transportation (using the same protocol stack)
3. A plugin / software extension is loaded onto the sensor firmware that creates an additional Object Instance that is commissioned to communicate through an HVAC system compatible stack directly to the HVAC system.

In summary: OpenAIS devices are internet connected at the node and can be integrated using different high level protocols. Therefore, the “single gateway” solutions of today’s system integrations will be history, once the Building Automation Systems switch to IoT technology also.

4.5.3.3 Control / DataCollect Object based integration

Out of the diverse HVAC/building automation integration strategies this part focuses on the integration using the Control Objects as point of integration.

In building automation the smallest structural entity is usually “a room”. Using the controller (or the data collector) Object as a base for the integration allows matching this kind of structural setup with the lighting control. Lighting typically has a higher granularity (many light points per room) than the building automation (HVAC) needs: Using area Control functions as the integration point resolves this issue.

1. The Control Object may expose any HVAC protocol for integration purposes (e.g. a Control Object may have a (configurable) BACnet interface embedded
2. The Control Object may expose a simple “group control” to the building automation system that allows the Building Automation System to use its
internal calendar functions and alike to switch the lights to on or off at specific times etc.

3. The Control function may create extended or summarized failure reports and hand them to building automation.

4. There may be specific Control Objects that are stacked on the area Control Objects that provide the integration into the building automation. This is structurally (and within the OpenAIS concepts) identical to the interface that is integrated into an area Control Object.

In summary: there are many “may have” statements in that integration text: that is due to the fact that integration has been made easy, but not mandatory with OpenAIS.

Please note that there is no “out-of-the-box” integration specified in OpenAIS. Such integration is not completely impossible, but system designers that provide this, need to be aware of all the privacy and security implications that may arise if the commissioning step that is used in OpenAIS to secure communication is avoided.

4.5.4 Integration into “Smart” Cloud Services

Besides the integration of the lighting controls system into the Building Automation System(s), the request to integrate lighting controls into cloud services as envisioned in “smart buildings” will come up soon. This section shows the principles for such integration.

OpenAIS is fully based on IoT technologies, (LWM2M is even especially designed to connect to the cloud) and that makes it relatively simple to provide cloud service based integration also. This part is considered to be out of scope for OpenAIS, it is completely up to the vendors to provide such interfaces either using separate Object Instances or providing additional bindings for specific resources.

Please note that with some cloud service interfaces the OpenAIS node Object API’s may work without any change: The nodes are IoT compatible in a specific way, and this can be simply used (after the authorization issues are handled in a commissioning step) by the cloud services.

- To reduce complexity and total bandwidth OpenAIS allows for “compressed” data access for the cloud services by using the data collect and the Control Objects. These two Objects expose “group control” and “group data” interfaces, that allow transferring data to the cloud services in a more controlled way, e.g. providing 1/4 hourly reports or room reports rather than single light point reports.

- But any way this is commissioned, a change to a more or less detailed integration base can be commissioned at any time, as the functional blocks and their interfaces are available, and the way they are used is “only” a commissioning / parameterization issue.
5 ARCHITECTURAL ANALYSIS

Architecture analysis in the design phase is a cost-effective approach to identify deficiencies and incorrect decisions. Without a thorough architectural analysis, there is a high probability of omissions/errors in the architecture and misunderstanding between stakeholders. Solving these problems in the later stages of development is very expensive, so it is important to evaluate the architecture as early as possible.

We have seen in Section 2.5 how we used the decision matrix to evaluate the effectiveness of the architectural decisions we made in the design stage and to evaluate the validity of the final candidate by comparing it with one the best state-of-the-art solutions. In this chapter we focus our architectural analysis on the risk assessment aspect.

Risk is defined as the effect of uncertainty on objectives [ISO 31000]. The effects are the deviation from expected outcome and can be negative or positive. Uncertainties are caused by ambiguity, events or lack of information. Risk assessment is the task of identifying and mitigating risks. In OpenAIS it is a part of the Task 2.3 Architectural risk assessment. This chapter reports the activities carried out to identify the risks in OpenAIS reference architecture.

Two workshops have been organised for risk assessment. The first workshop was architecture-centric where an internal assessment of the architecture has been carried out by the architects with the goal of improving the understanding of the architecture and obtaining agreement among the architects. The second workshop was stakeholder-centric and concentrated on eliciting stakeholder points of view and bridging the gap between the architects and the customers. Hence in addition to the architects of OpenAIS (WP2 team), customer representatives (WP1 team) were present in the latter workshop.

5.1 ATAM - Architecture Tradeoff Analysis Method

We used the Architecture Tradeoff Analysis Method (ATAM) [Kazman00] for our evaluation. ATAM, developed by the Software Engineering Institute at the Carnegie Mellon University is a leading method to evaluate software architectures [Kazman00]. ATAM evaluations reveal how well the architecture satisfies particular quality goals and expose architectural risks that potentially inhibit the achievement of an organization's business goals. This risk-mitigation process is most beneficial when done early in the software development life cycle.

5.1.1 Goals of ATAM

The major goal of ATAM is to:

- Evaluate the architectural design decisions to determine if they satisfactorily address the quality requirements.

A prerequisite of an evaluation is to have a statement of quality attribute requirements and a specification of the architecture. However, in the early stage of design they can be vague and ambiguous. Hence, two other important goals of ATAM are to:

- Elicit and refine a precise statement of the architecture’s driving quality attribute requirements
- Elicit and refine a precise statement of the architectural design decisions
5.1.2 Steps of ATAM

The ATAM process consists of nine steps [Kazman00]. We have slightly adapted the steps, namely 4, 6 and 8 by changing the architectural approaches to architectural decisions. ATAM considers architectural approaches and styles as the means to ensure quality attributes. However, the key objective is to identify how the architectural decisions embodied by these architectural approaches and styles affects the achievement of a quality attribute. In the early phase of design of a complex system like OpenAIS, it is easier to directly weigh the architectural decisions. Figure 36 shows the adapted steps of the method used for our analysis.

![Figure 36: The steps of the ATAM](image)

The first stage involves a presentation where the method is explained to the stakeholders and the motivating business goals and key architectural drivers are presented. The architect then presents the architecture, focusing on how it addresses the business drivers. The next stage is the investigation and analysis stage where the key architectural decisions are identified and then the quality factors that comprise system utility are elicited, and prioritized. The results are captured in a utility tree, a hierarchic model of the driving architectural requirements. A sample utility tree is given in Figure 37.

To create a utility tree, we identify the key quality attributes. We then put specific concerns under each of the quality attributes as the next level in the tree. Concerns are then characterized by scenarios which are the leaf nodes of the tree. A scenario is a short description of the interaction of one of the stakeholders with the system, e.g. user’s scenario, developer’s scenario, customer’s scenario, maintainer’s scenario, etc. A scenario represents a use, or modification, of the system and can help in determining if the architecture meets a functional requirement and more significantly the system qualities. The scenarios are prioritized along two dimensions:

- Importance to the system to the stakeholders
- Perceived risk in achieving the particular goal by the architects
The last step in the investigation and analysis stage is eliciting and analysing the architectural decisions that address the quality factors in the utility tree. During this step architectural risks, sensitivity points, and trade-off points are identified. After this phase, there is a testing stage where use case scenarios are elicited and prioritized and architectural decisions to support the highly ranked scenarios are identified. The last stage is the reporting stage where the findings are presented to the stakeholders and report is written detailing this information.

5.2 Risk Assessment Workshop 1

In the first workshop, the Steps 1-6 were executed. For Step 2, the key business drivers were extracted from WP1 D1.1 “Selected Scenarios and Use Cases” [OpenAIS_D1.1]. To identify the most common quality attributes and put them in the utility tree in Step 5, we considered the ISO/IEC 25010:2011 Systems and software quality attribute list given in Table 4.

Table 4: ISO/IEC 25010:2011 Systems and software quality attributes

<table>
<thead>
<tr>
<th>Functional suitability</th>
<th>Performance efficiency</th>
<th>Portability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional completeness</td>
<td>Time behaviour</td>
<td>Adaptability</td>
</tr>
<tr>
<td>Functional correctness</td>
<td>Resource utilization</td>
<td>Installability</td>
</tr>
<tr>
<td>Functional appropriateness</td>
<td>Capacity</td>
<td>Replaceability</td>
</tr>
</tbody>
</table>

Usability
- Appropriateness
  - Recognisability
- Learnability
- Operability
- User error protection
- User interface aesthetics
- Accessibility

Security
- Confidentiality
- Integrity
- Non-repudiation
- Accountability
- Authenticity

Maintainability
- Modularity
- Reusability
- Analysability
- Modifiability
- Testability

Reliability
- Maturity
- Availability
- Fault tolerance
- Recoverability

Compatibility
- Co-existence
- Interoperability
The main attributes in Table 4 are indeed the most relevant attributes to OpenAIS. We put specific concerns and scenarios after referring to the Decision Matrix [OpenAIS_D2.2]. Architects then evaluated the scenarios in subgroups and scored their importance and risks in coarse levels viz. high, medium and low. Figure 38 shows a part of the utility tree created during the workshop. A few scenarios with high risks are shown in the leaf nodes of the utility tree in Figure 38. The complete list included ten quality attributes and forty five scenarios.

![Utility Tree](image)

**Scenario 1.1.1:** Time to Light (TTL) when a user presses a switch < 500 ms [H,H]
**Scenario 3.1.1:** Support for extensions so that a vendors can deliver and use device functionality up and above the standard features [H,H]
**Scenario 6.1.1:** Uses service integration technologies familiar to IT group staffs [M, H]

<table>
<thead>
<tr>
<th>Identified risks</th>
<th>Quality attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowance of 6lowPAN IP networks raises high risks to the real-time/latencies and synchronicity requirements</td>
<td>Performance</td>
</tr>
<tr>
<td>Missing extension concept in object models and definition for HW extension interface affecting the capability to accommodate new/other technologies</td>
<td>Extensibility</td>
</tr>
<tr>
<td>Support for easy integration of emergency lighting is not possible as IP-systems are not rated for that and secure emergency installations will not be integrable</td>
<td>Extensibility</td>
</tr>
<tr>
<td>Non-selection of standard management tools and protocols to reuse from IT domain and feasibility of integration in the existing security systems have not been addressed</td>
<td>Reusability</td>
</tr>
<tr>
<td>New device with new UID may introduce a new IP address and the controls do not work anymore when based only on IP address</td>
<td>Availability</td>
</tr>
<tr>
<td>Architectural decisions are not made explicit or their documentation is missing, e.g.</td>
<td>Availability</td>
</tr>
<tr>
<td>• No control devices in the design (with the exception of routers) have single point of failure not documented</td>
<td></td>
</tr>
<tr>
<td>• Fallback behaviour/function of components needs to be specified and mandated</td>
<td></td>
</tr>
<tr>
<td>• Recommended PHY specifics are missing</td>
<td></td>
</tr>
</tbody>
</table>

The architects then took the most important scenarios with high or medium risk for identifying the architectural decision taken or missing to support the quality attributes.
Table 5 lists a sample set of identified risks from the workshop. The complete list of architectural decisions and the corresponding identified risks are given in Appendix A.

5.3 Risk Assessment Workshop 2

In the second workshop, we continued with the Step 7 of ATAM to come with scenarios from the customers’ perspective. We reiterated through the Steps 1 – 6 and analysed the architectural decisions (Step 8) of ATAM more in a CAFCR style [Muller04] to bridge the gap between the architects and the customers.

One of the risks of the reference architecture is that it is inherently abstract where the quality attributes such as scalability, extensibility and generality are there. However, when we try to instantiate the architecture, we often identify that it is way too complicated or cannot configure easily and hence difficult to ensure the qualities. As we are creating a reference architecture in OpenAIS we wanted to mitigate this risk. For this, we begin our analysis by considering one of the systems based on OpenAIS reference architecture which will be used in 2020. As an example we picked a bank which is refurbishing/renewing its four floors in 2022.

In the workshop, we tried to understand how we are going to design this refurbishment from a lighting perspective and what steps we are going to make. The WP1 representatives came up with the specification that the bank has Building Management Systems, uses three IPv4 networks (very secure for bank transactions, for normal IT operation, and for building services operations) and then worked out for the requirements. They came up with the assumptions of different spaces in the bank (e.g. Atrium, Trading Floor, Stairs, Corridor, Meeting rooms, Archive, Datacentre etc.).

In the second part of the workshop, we tried to identify the steps in installation and commissioning of the system and assigned a timeline to these steps. One of the key requirements from the WP1 members was to reduce the on-site time of expensive personnel in the commissioning process. Some of the dominant concerns/questions came up in the discussion are:

- When is IT available and live (power and network configuration) during installation and commissioning?
  - Is the internal network in the building working at this stage?
  - Is the external communication to internet also available (e.g. cloud service support during commissioning)?
- The phasing of the steps in installation and commissioning is very important – today’s phasing is adversarial to OpenAIS system, as IT is available only after lighting installations.
- Which one - wired or wireless to use in our scenario?
  - Considering security, performance and maintenance for 25 years wired is preferred in the considered scenario.
- Can we simplify the commissioning process or even remove the need of the commissioner?
- Installation and commissioning period of OpenAIS can take around 12 weeks in the particular scenario considered, if we follow today’s phasing, whereas current lighting systems need only 11 weeks. There is a need for a smart commissioning tool that allows easy installation and commissioning, but it needs to be developed.
- What happens if IT issues arise during installation and commissioning?
- Common network service for lighting and building management is a requirement for the future. How do we address this?
In the last part of the workshop, we selected three key topics from the list of important aspects identified from the first ATAM workshop and list of topics identified by WP1 (listed in D1.2 User Requirements [OpenAIS_D1.2]. The topics are: 1) Network availability, 2) Diagnosis, triage and 3) Scalability. The topics are discussed in groups and the section below highlights the exemplary findings from the workshop:

1. Network availability:
   - During installation and commissioning network availability issue arises due to phasing issue, i.e. IT availability during commissioning. To solve this:
     - Contractual agreement can be made to ensure that IT is available at the commissioning stage.
     - Organizational preliminary network that is available after installing lighting specific network components can be made independent from standard IT network by using temporary devices (e.g. preliminary switches/routers, tablet with SIM that can directly connect to cloud, etc.)
   - During operation time network availability (24x7) issue arises due to i) operational throughput loss, ii) part Failures, iii) IT misconfiguration and iv) RF interference. These issues can be resolved by:
     - Operational throughput loss
       - It is a contractual issue: use either a separate lighting network or ensure minimum BW available for lighting purpose.
       - Use IPv6 priority flags to receive higher priority for lighting packets (but it may not be available always).
     - Part Failures
       - Limit spreading of failures and use the fallback controller provision.
       - Redundant network components and connections to circumvent failing parts.
     - IT misconfiguration
       - Triage is possible by using a tool that asks components to respond and if response is different, it can detect issues.
     - RF interference
       - Enable channel agility in PHY or have a tool (in the devices) that can detect issues and change channels.
       - Limit the hop count to 1. This introduces the need for additional border routers. However, it is cost effective when compared to the cost for debugging (say with two staffs), as it solves several issues due to larger hop counts.

2. Diagnosis and triage
Diagnosis is needed when i) user complains, ii) system generates alarms or iii) analytic alarms are raised. From diagnostics perspective, the severity of user complaints is higher when compared to most system generated alarms which in turn is higher than analytic alarms. Some examples of such instances are listed below:
   - User complaints when
     - Devices break
     - Strange behaviours occur
     - Unexpected responses occurs (remark: System works according to specification but unexpected to users)
   - System alarms are generated when
     - Error message
     - Online status/missing devices
• Analytic alarms are raised when
  o Trends identify drifting or degrading paths (e.g. battering life time degrades, too much run time needed for blinds to open, etc.)
  o Statistics/history shows probability of failures

For triage, the possibilities are:
  • Checking the message flow
    o Logging provision should be there (Check if a logging standard is needed in OpenAIS)
    o Replay of events
    o Timestamping events (Check if there is common notion of time in OpenAIS systems)
  • Simulation can identify problems. It also allows to playback logged events and detect/analyse the problems.

3. Scalability
  • In the typical office building, limits are imposed by PHY medium. To solve it, networks can be segmented. Guidelines are needed how to do it effectively!
  • When we have outside display on the façade of the whole building, multi-hop issues and synchronicity of messages become more important.
  • In wired network, though rather rare bad protocols could cause scalability issues more often than expected. The combination of many devices, many multicast groups, a flat network topology (w.r.t. broadcast/multicast scope) and poor filtering of multicast traffic could result in serious network loads. Some network technologies (like VLANs or subnets per floor) may help.
6 CONCLUSIONS

This deliverable described the final architecture of OpenAIS - an open service oriented IP-based lighting architecture. The architecture provided in the document is a reference architecture that can support a wide range of deployment scenarios and use cases of future office buildings.

The system requirements that are used as the base of the OpenAIS Architecture and the process followed to arrive at the final candidate architecture are described in this document. Three candidates were shortlisted in this process and using the Decision Matrix, a tool we developed, the candidates were compared. The IoT-centric architecture emerged as the winner in this process. A comparison of the final candidate architecture with a recent heritage system reveals that the final candidate not only matches the robustness, reliability and real-time performance of today’s dedicated lighting systems but can excel them in many aspects such as security, interoperability, ease of installation in addition to being open, allow vendor differentiation and reuse from IT domain.

An extensive overview of the proposed reference architecture is given in this document covering four different architectural views, namely logical, physical, networking, and security view. Logical model of interaction, interfacing, communication, interoperability (with Building Automation Systems), extensibility and commissioning aspects are also covered in this document. These are accompanied by some high-level examples and illustrations. Various recommendations for system design on the choices of PHY, network stack and operating systems are also provided.

In addition to using decision matrix to evaluate the effectiveness of the architectural decisions made in the design stage and to evaluate the validity of the final candidate, further architectural analysis, focussing on risk assessment has been employed to detect the potential risks in the early stage of the project. The outcome of the two workshops conducted and potential risk identified are also explained in this report.
7 REFERENCES


[OpenAIS_D2.1] E. Dijk, Identify potential architectures for OpenAIS system, including state of the art in SSL, June 2015.

[OpenAIS_D2.2] T. Ozcelebi, Implementation & Verification Guidelines for the Architecture: Selection of the architecture. Publically available at:


8 APPENDIX

8.1 Analysis of Architectural Decisions

This section describes the analysis of the key architectural decisions in the light of nine most important (with high or medium risk) scenarios selected from a set of forty five scenarios. The format of the content is as follow:

For each selected scenario

• The corresponding the quality attribute, concern and scoring on importance and perceived risks identified during the creation of utility tree are listed
• Additional explanations to interpret the scenarios follows
• Finally, a table with architectural decisions taken or missing, rationale for this and identified risks

Legends used in the table are:
⊕ - helps, ⊖ - doesn’t help/against, ?⊕ - Missing in OpenAIS, but if added would help

Attribute 1: Performance
Concern 1.1: Real time requirements/Latencies & Synchronicity

Scenario 1.1.1: Time to Light (TTL) when a user presses a switch < 250 ms [H, H]
  a) High risk to achieve < 250 ms for a large group of luminaries together
  b) Instead of 250 ms, 500 ms might be sufficient
  c) High importance because of user experience, expectation, risk from wireless uncertainty

Scenario 1.1.2: Time to Light (TTL) when a user’s presence is detected user becomes present < 250 ms [H, M]
  a) As in Scenario 1.1.1
  b) 750 ms might be sufficient
  c) User expectation little bit more relaxed (enter a dark room and lights turn-on)

No: Architectural decisions | Rationale & Identified risks
---|---
1 ⊖ | Allow 6lowPAN IP networks [contradiction]
2 ⊘ | Use of IPv6 “traffic class” in IP packet Risks: Not implemented in Thread/BR
3 ⊘ | Local control is allowed Distributed architecture i.e. segmentation
4 ⊘ | IPv6 multicast is supported (even secure)
5 1.1.2 – Guideline for sensor detection time (presence sensor) < 500ms (750ms since user enters room)
6 1.1.2 – Combined sensor + luminaire possible in one physical device

Attribute 2: Availability
Concern 2.1: Component failures

Scenario 2.1.1: The system is available 364/365 [H, M]
  a) Reliable power
  b) Expected uptime much higher (99.9)
  c) Seems achievable with normal engineering practice

No: Architectural decisions | Rationale & Identified risks
---|---
1 ⊕ No architectural decisions yet

2 ⊗ No single point of failure devices in the design (with the exception of routers) Not yet documented*

3 ⊗ For more central devices (e.g. area control) the architecture allows for backup functions Not yet documented*

4 ⊗ Fallback to "out of the box/addressed" to be made mandatory for sensors and luminaires

4 ⊗ "Last action served" allows for easy switch over

*Note: At the time of the risk assessment workshop 1

**Concern 2.2: Communication failures (Too many message losses, too much delay/out of order, and too much interference in wireless channel)**

**Scenario 2.2.1:** System gives the minimum performance to users (e.g. user presses a switch TTL < 250 ms) even in case of (incidental) network failures [M, M]

a) Wireless is the potential risk
b) Rare case – can be accepted

<table>
<thead>
<tr>
<th>No</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Not much work on this</td>
<td>PHY specifics</td>
</tr>
<tr>
<td>2</td>
<td>Depends on PHY e.g. TISCH, special, time, frequency, diversity – (99.99%)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>There is no backup for a completely failed network</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Fallback behaviour of components shall be specified</td>
<td>Detection, timeout, etc. need to be specified</td>
</tr>
<tr>
<td>5</td>
<td>Do we need QOS- detection?</td>
<td></td>
</tr>
</tbody>
</table>

**Concern 2.3: Availability issue due to components from single/different vendors**

**Scenario 2.3.1:** Replacement of sensors/luminaries does not have impact on lighting system availability (only local influence) [H, M]

a) Assume repair in running systems, roll back configurations
b) Interoperability is projects goal
c) Single vendor - less risk

<table>
<thead>
<tr>
<th>No</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IPv6 choice – Issue: new device with new UID – may cause new IP address. So existing controls don’t work anymore when based only on IP address;</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Well defined commissioning process: Replacement of device and putting the old device setting into the new device May copy IP also</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>For sensor/control functions we need a default operation specification what to do in such case of ‘sensor device out' and more</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Possible: use of DNS or mDNS for indirect addressing</td>
<td></td>
</tr>
</tbody>
</table>

**Attribute 3: Extensibility – Modifiability**

**Concern 3.1: Capability to accommodate new/other technologies**
Scenario 3.1.1: Support for easy integration of emergency lighting [H, M]
  a) Depends on integration level
  b) Strong legislation
  c) Separate systems to be integrated

<table>
<thead>
<tr>
<th>No</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Do not use OpenAIS for “life-saving operations”</td>
<td>IP-systems are not rated for that Secure emergency installations will not be integrable</td>
</tr>
<tr>
<td>2</td>
<td>Reporting &amp; testing</td>
<td>Need to be secured using the standard IP mechanisms</td>
</tr>
</tbody>
</table>

Concern 3.2: Capability to accommodate new/other technologies
Scenario 3.2.1: Support for extensions so that vendors can deliver and use device functionality up and above the standard features [H, H]
  a) Project requires
  b) High uncertainty

<table>
<thead>
<tr>
<th>No</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Object models</td>
<td>Missing extension concept!</td>
</tr>
<tr>
<td>2</td>
<td>Plug-in model available</td>
<td>Where and when do you need it? How does it work?</td>
</tr>
<tr>
<td>3</td>
<td>Stacked control functionality</td>
<td>Allows addition of functionality</td>
</tr>
<tr>
<td>4</td>
<td>No HW extension interface defined</td>
<td></td>
</tr>
</tbody>
</table>

Concern 3.3: Capability to accommodate new/other technologies
Scenario 3.3.1: Support for extensions so that vendors can deploy additional system functionality to all networked devices [H, M]
  a) Large scope
  b) Even increase in uncertainty

<table>
<thead>
<tr>
<th>No</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Firmware update</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Plug-ins allowed</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Stacked control functionality</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Webservers coupling</td>
<td>Details missing!</td>
</tr>
</tbody>
</table>

Concern 3.4: Capability to accommodate new/other technologies
Scenario 3.4.1: Support for extensions so that vendors can detect and use additional functionality supplied by a foreign device [M, H]
  a) For standard functionality

<table>
<thead>
<tr>
<th>No</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Standardized object models will be used</td>
<td>Control functions can rely on data formats and meaning and on methods provided</td>
</tr>
<tr>
<td>2</td>
<td>Scheduled device management/service detection will be used</td>
<td>Services may be detected during operation/commissioning</td>
</tr>
<tr>
<td>3</td>
<td>Controller function to operate on new changed services will be commissioned/set to do so</td>
<td>Decoupling of sensors and actuators helps with new functionalities</td>
</tr>
</tbody>
</table>

Attribute 4: Reusability
Concern 4.1: Reuse from IT domain
Scenario 4.1.1: Uses service integration technologies familiar to IT group staffs [M, H]
a) IPv4 is the standard. IPv6 and 6LowPAN are still “not familiar”

<table>
<thead>
<tr>
<th>No.</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No selection of standard management tools + protocols yet (SNMP)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>⊕ It is IP after all</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>⊕ IoT standards to be developed. Any time on the market</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>⊕ Integration in existing security systems – not discussed</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>⊕ What are the usual IT questions? How many IP addresses, bandwidth requirements, border routers?</td>
<td>Not addressed</td>
</tr>
</tbody>
</table>

**Attribute 5: Security**

**Concern 5.1: Authenticate and authorize users**

**Scenario 5.1.1:** User (installers, developers and end user) wants to log in using an approved authentication means during operations, and system recognizes the user and accords him appropriate access permissions. This process should be easy for legitimate users. [H, M]

a) Users and “1234” passwords are a security risk or a difficult issue

<table>
<thead>
<tr>
<th>No.</th>
<th>Architectural decisions</th>
<th>Rationale &amp; Identified risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>⊕ User access is through controller function only</td>
<td>Only one access to control one area needs to be encouraged</td>
</tr>
<tr>
<td>2</td>
<td>⊕ Controller functions to provide aggregated data from sensors</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>⊕ Controller functions to route enhanced access to “its” devices</td>
<td>No temporary users in actuators or sensors!</td>
</tr>
</tbody>
</table>
9 GLOSSARY

9.1 Abbreviations and Acronyms

Many of the capitalized terms used in the ‘Description’ column below are further explained in the ‘Definitions’ table in Section 9.2 or otherwise in the reference or website listed in within square brackets.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6LoWPAN</td>
<td>IPv6 over Low-power lossy Wireless Personal Area Networks [RFC4944]</td>
</tr>
<tr>
<td>6LoBAC</td>
<td>6LoWPAN over BACnet’s MS/TP protocol</td>
</tr>
<tr>
<td>6LBR</td>
<td>6LoWPAN Border Router [RFC6775]</td>
</tr>
<tr>
<td>6LN</td>
<td>6LoWPAN Node [RFC6775]</td>
</tr>
<tr>
<td>6LR</td>
<td>6LoWPAN Router [RFC6775]</td>
</tr>
<tr>
<td>6ND</td>
<td>6LoWPAN Neighbor Discovery [RFC6775]</td>
</tr>
<tr>
<td>6TiSCH</td>
<td>IPv6 over the TSCH mode of IEEE 802.15.4e [<a href="http://www.ietf.org">www.ietf.org</a>]</td>
</tr>
<tr>
<td>802.3</td>
<td>The IEEE 802.3 standards on wired communication (Ethernet)</td>
</tr>
<tr>
<td>802.11</td>
<td>The IEEE 802.11 standards on wireless communication (Wi-Fi)</td>
</tr>
<tr>
<td>802.15.4</td>
<td>The IEEE 802.15.4 standards on low-power low data rate wireless communication protocols [<a href="http://www.ieee802.org/15/pub/TG4.html">www.ieee802.org/15/pub/TG4.html</a>]</td>
</tr>
<tr>
<td>AAA</td>
<td>Authentication, Authorization and Accounting</td>
</tr>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>ADR</td>
<td>Automatic Demand Response (smart grid domain term)</td>
</tr>
<tr>
<td>AP</td>
<td>Access Point (Wi-Fi specific term)</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AS</td>
<td>Authorization Server</td>
</tr>
<tr>
<td>ASN</td>
<td>Absolute Slot Number</td>
</tr>
<tr>
<td>ATAM</td>
<td>Architecture Trade-off Analysis Method</td>
</tr>
<tr>
<td>BACnet</td>
<td>Building Automation and Control Networks [<a href="http://www.BACnet.org">www.BACnet.org</a>]</td>
</tr>
<tr>
<td>BAS</td>
<td>Building Automation Systems</td>
</tr>
<tr>
<td>BMS</td>
<td>Building Management System</td>
</tr>
<tr>
<td>BTLE</td>
<td>Bluetooth Low Energy (also known as Bluetooth Smart)</td>
</tr>
<tr>
<td>CBOR</td>
<td>Concise Binary Object Representation [RFC7049]</td>
</tr>
<tr>
<td>CoAP</td>
<td>Constrained Application Protocol [RFC7252]</td>
</tr>
<tr>
<td>COSE</td>
<td>CBOR Object Signing and Encryption</td>
</tr>
<tr>
<td>CT</td>
<td>Commissioning Tool</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
</tbody>
</table>
### Abbreviation Description

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS-SD</td>
<td>DNS Service Discovery [RFC6763]</td>
</tr>
<tr>
<td>DTLS</td>
<td>Datagram Transport Layer Security [RFC6347]</td>
</tr>
<tr>
<td>ER</td>
<td>Edge Router</td>
</tr>
<tr>
<td>EUI</td>
<td>Extended Unique Identifier (defined by IEEE)</td>
</tr>
<tr>
<td>ETH</td>
<td>Ethernet</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force [<a href="http://www.ietf.org">www.ietf.org</a>]</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission [<a href="http://www.iec.ch">www.iec.ch</a>]</td>
</tr>
<tr>
<td>ISA</td>
<td>International Society of Automation [<a href="http://www.isa.org">www.isa.org</a>]</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPv4</td>
<td>Internet Protocol version 4</td>
</tr>
<tr>
<td>IPv6</td>
<td>Internet Protocol version 6</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>JN</td>
<td>Joining Node</td>
</tr>
<tr>
<td>KDC</td>
<td>Key Distribution Centre</td>
</tr>
<tr>
<td>L1, L2, L3, ...</td>
<td>Layer 1, Layer 2, Layer 3, ... OSI layers reference.</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LLN</td>
<td>Low-power Lossy Network (typically, wireless low-bandwidth)</td>
</tr>
<tr>
<td>LWM2M</td>
<td>Lightweight M2M [openmobilealliance.org]</td>
</tr>
<tr>
<td>M2M</td>
<td>Machine-to-Machine communications</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control (layer)</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base [RFC1155]</td>
</tr>
<tr>
<td>MIC</td>
<td>Message Integrity Code</td>
</tr>
<tr>
<td>mDNS</td>
<td>Multicast DNS (protocol) [RFC6762]</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base [RFC1155]</td>
</tr>
<tr>
<td>MPL</td>
<td>Multicast Protocol for Low-power lossy networks</td>
</tr>
<tr>
<td>MS/TP</td>
<td>Master Slave / Token Passing – a protocol defined in BACnet</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>ND</td>
<td>Neighbor Discovery</td>
</tr>
<tr>
<td>NFC</td>
<td>Near Field Communication</td>
</tr>
<tr>
<td>ODM</td>
<td>Object Data Model</td>
</tr>
<tr>
<td>OIC</td>
<td>Open Interconnect Consortium [openconnectivity.org]</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OM</td>
<td>Object Model</td>
</tr>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
<tr>
<td>OpenAIS</td>
<td>Open Architectures for Intelligent Solid-state lighting systems</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection (project at [<a href="http://www.iso.org">www.iso.org</a>])</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>PD</td>
<td>Presence Detector</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical layer</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PLC</td>
<td>Power Line Communication</td>
</tr>
<tr>
<td>PoE</td>
<td>Power over Ethernet</td>
</tr>
<tr>
<td>PSK</td>
<td>Pre-Shared Key</td>
</tr>
<tr>
<td>REST</td>
<td>REpresentational State Transfer</td>
</tr>
<tr>
<td>RIP</td>
<td>Routing Information Protocol [RFC2080]</td>
</tr>
<tr>
<td>RN</td>
<td>Routing Node</td>
</tr>
<tr>
<td>RPL</td>
<td>Routing Protocol for Low-power lossy networks [RFC6550]</td>
</tr>
<tr>
<td>RPK</td>
<td>Raw Public Key</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real-Time Operating System</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service (SMS transport method defined in [LWM2M])</td>
</tr>
<tr>
<td>SLIP</td>
<td>Serial Line Internet Protocol [RFC1055]</td>
</tr>
<tr>
<td>SCA</td>
<td>Sense-Control-Actuate (OpenAIS-specific)</td>
</tr>
<tr>
<td>SSL</td>
<td>Solid State Lighting</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol [RFC793]</td>
</tr>
<tr>
<td>TRNG</td>
<td>True Random Number Generator</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol [RFC768]</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URN</td>
<td>Uniform Resource Name</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual Local Area Network [link]</td>
</tr>
<tr>
<td>VLC</td>
<td>Visible Light Communication</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Wireless Fidelity / the Wi-Fi standard based on IEEE 802.11</td>
</tr>
</tbody>
</table>
## 9.2 Definitions

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA Server</td>
<td>A software function for AAA (Authentication, Authorization and Accounting) typically residing on a server located on premises or in the cloud. In OpenAIS this server is used to secure group communication.</td>
</tr>
<tr>
<td>Access Point (AP)</td>
<td>An access point according to the Wi-Fi (IEEE 802.11) standard, to which Wi-Fi clients can connect.</td>
</tr>
<tr>
<td>Actuate-Function</td>
<td>A Function that enables an actuator to change state; for example a Light-Function. An Actuate-Function is under control of one Control-Function. See also: SCA model</td>
</tr>
<tr>
<td>Application (Layer)</td>
<td>Synonyms: Gateway</td>
</tr>
<tr>
<td>Gateway</td>
<td>A group of Object instances that is defined during Commissioning for an application-specific purpose, for example a group of Light Points to control. The members of this group may reside on multiple Devices. See also: Multicast Group</td>
</tr>
<tr>
<td>Application Server</td>
<td>Server that executes software applications; the server typically residing somewhere in a Backbone Network or in the Cloud. The Application Server typically communicates with Nodes using IP. Such IP connections traverse other networks for example an Intranet, VPN, Backbone Network or the public Internet.</td>
</tr>
<tr>
<td>Area Controller</td>
<td>A connected device that includes one or more Control-Functions and does not function as a Luminaire or sensor itself. It is connected somewhere in the Field Network or Backbone Network. Note: this device may be integrated with a Border Router function</td>
</tr>
<tr>
<td>Room Controller / Floor Controller</td>
<td>Note: not to be confused with Area Control-Function</td>
</tr>
<tr>
<td>Floor Controller</td>
<td>A Control-Function that controls a specified area, which may be a room or floor. A Function is not a physical device; it could be allocated to any Device including Luminaires or Sensors.</td>
</tr>
<tr>
<td>Backbone Network</td>
<td>A high-speed, reliable IP data Network that exists in a building and connects multiple network Segments with each other. Example: switched Ethernet segment in an office building, that connects all the Wi-Fi Access Points in a building See also: Subnet</td>
</tr>
<tr>
<td>Back-end Server</td>
<td>A server that is not located in the Field Network, but rather on the Backbone Network or even more remote: a Cloud Server. See: Cloud Server</td>
</tr>
<tr>
<td>Binding</td>
<td>A configurable relation between a producer of information and a receiver of this information, where the receiver typically resides on a different Node. Example: a presence-sensing function is bound to the group G1 of luminaires Example: a Commissioning Tool configures a binding from a room controller to a floor controller, such that the floor controller is permanently kept up to date about the occupancy and lights status of the room.</td>
</tr>
<tr>
<td><strong>Definition</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Bootstrap Server</td>
<td>A software function defined in Lightweight M2M [LWM2M] that enables bootstrapping an OpenAIS Device including joining a network securely for the first time. The bootstrap server authenticates the new device and provisions keying material to the device to enable it to connect to a (Lightweight M2M) device management server. This function may be located on a Commissioning Tool, or on a (Cloud) Server.</td>
</tr>
<tr>
<td>Border Router (BR)</td>
<td>A Router residing at the border (boundary) of a PHY Segment, realizing the routing of IP packets between the two distinct PHY+MAC technologies. There may be zero, one, or multiple Border Routers present in a single PHY Segment. A Border Router typically has additional functions to manage Nodes in a PHY Segment or to allocate addresses to Nodes. Example: 6LoWPAN Border Router (6LBR) which routes packets between an IPv6 network (typically Wi-Fi or Ethernet) and a 6LoWPAN network (typically an RF network). Note: a Border Router as such does not use application-specific software to translate application level protocols in data packets traversing the Border Router, like a Gateway does. But a Gateway function can be integrated in a BR.</td>
</tr>
<tr>
<td>Bridge</td>
<td>Synonym: Switch</td>
</tr>
</tbody>
</table>
| Client              | Multiple meanings are used in this document:  
  - Client: An entity that uses an interface that an OpenAIS system exposes  
  - CoAP Client: the client role of the client/server CoAP protocol as defined by [RFC7252]  
  - LWM2M Client: the client role of a Node in the Lightweight M2M [LWM2M] protocol | |
<p>| Client/Server       | The commonly used client/server communication pattern. Example: CoAP is a client/server protocol. Note: Peer-to-Peer communication can be implemented using a Client/Server protocol, where usually each peer (Node) acts in both of (or alternating between) a client role and a server role. See also: Peer-to-Peer |
| Cloud Server        | Concept of software services running transparently on server(s) connected to the Internet; of which the location is not pre-defined and computing resources can be automatically scaled up or down depending on resource demands. This can be in a private data centre, shared data centre, or even on-site. |
| Commissioner        | The person or group of persons that performs the Commissioning process. See also: Commissioning |</p>
<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
</table>
| Commissioning              | The process of configuring the Devices that are part of a (lighting) system in such a way that the system will afterwards operate according to the intended design objectives (i.e. as the customer would like to have it); including any required verification activities. Often this process is performed on-site after the individual Devices that are to be part of a (lighting) system are already installed in the ceiling. In OpenAIS the following phases of Commissioning are distinguished (in brackets the person competences needed to perform the work):  
  - Phase 1: Installation, localization and test (electrical contractor, on site)  
  - Phase 2: Binding; and if needed finalize localization (Commissioning Engineer, IT person, possibly off site with some low tier support on site)  
  - Phase 3: Functional Parameterization (lighting specialist)  
  - Phase 4: Fault finding missions (Commissioning Engineer, IT person, electrical contractor) |
<p>| Commissioning Engineer     | See: Commissioner                                                                                                                                                                                           |
| Commissioning Tool (CT)    | A device brought into the building premises with the specific purpose to perform commissioning operations. It may connect to a Subnet, to a Backbone Network, or only to the luminaire/sensor directly while doing these operations. The Commissioner uses a Commissioning Tool. Example: a tablet with specific software to perform Commissioning  |
| Control-Function           | A Function that can control one or more Actuator-Functions, or other Control-Functions, possibly based on events and information reported by Sensor-Functions.                                                                 |
| Converter                  | A Device or module (part of a Device) that controls one or more Light Points. The Converter generates the specific voltage and current needed to activate a Light Source. Typically a higher voltage is converted to the low voltage needed to safely activate LEDs. Synonym: Driver  |
| Datagram                   | Synonym: Packet                                                                                                                                                                                             |
| Device                     | A physical device; a box with physical interfaces within a casing/package; installed typically as one entity which cannot operate when disassembled in parts. A device may offer one or more Services in OpenAIS context. A Node is a networked Device. See also: Node  |
| Device-to-Cloud            | A communication pattern where a Device in the Field Network contacts a remote Node or service or Cloud Server, or vice versa a Device is the Field Network is contacted by such remote entity to perform communication. See also: Peer-to-Peer, Client/Server  |
| Device Management (function)| Specific functions in an OpenAIS system to register a Device the first time it is used; to keep track of its status including ID and hardware/software version; manage software updates; manage configuration settings; keeping track of dynamic status items such as errors that occurred, (network) diagnostics and (self-)test results. Note: configuration settings includes those settings needed for the lighting functions to operate as planned, such as application group definitions, sensor/control thresholds, time-outs, algorithm selection, control function parameters, etc. This is applied during the process of Commissioning during which Device Management functions of Devices are used. |
| Driver / LED Driver        | Synonym: Converter                                                                                                                                                                                         |</p>
<table>
<thead>
<tr>
<th><strong>Definition</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| Edge Router (ER)    | Synonym: Border Router  
Note: ‘Border Router’ is used within OpenAIS context.                                                                                                                                                       |
| End Node            | A Node that never ‘relays’ data packets destined to other nodes because it is unable to do so by design. An End Node can be only the original source or the final destination of a data packet.  
See also: Gateway Node, Mesh Network, Routing Node |
| Endpoint            | A Node that is either IP Source, IP Destination, or both.  
Note: this term can also be used to describe an application on a Node which is the IP Source or IP Destination. In this case there may be multiple Endpoints per Node, referring to the individual applications.  
Example: a luminaire that accepts UDP packets and processes the commands enclosed in these packets |
| Event               | A message containing information that is (potentially) of importance to be received by an interested entity in an OpenAIS system. The interested entity that receives an Event is called consumer, receiver or Event receiver. The entity that generates an Event can be called the Event producer, although we also refer to it by other names (such as Sensor).  
Events may be sent over the network between OpenAIS Nodes. |
| Field Network       | The collective name for all network Segments in a building that provide the last-meters connectivity to the building (IoT) devices like luminaires, buttons, sensors or HVAC actuators. It can be implemented using one type of physical medium (PHY, L1) or multiple PHYs.  
Note: a Field Network may include Stand-alone Networks.  
See also: Backbone Network |
| Firewall            | A network security function that controls the incoming and outgoing network traffic on a Node, based on an applied rule set.  
See also: Network Address Translation |
| Frame               | A single unit of multiple data bits at Link Layer (L2) that is transmitted over a single Link by a Node and possibly received by other Node(s)  
Synonyms: packet |
| Function            | In the WP2 architecture ‘Function’ is used to describe a capability allocated to a Device e.g. Sense, Control or Actuate Functions. A Function may be realized by a Service or by a set of Services that works in unison.  
See also: SCA model |
| Gateway             | A Node that translates an incoming data Packet from one format into another format and sends out the translated packet again. The translation modifies the contents of a Packet in at least one or more of the higher protocol layers (above L3).  
Note: a Gateway can also translate IP-based messages into non-IP messages or vice versa. |
| Hub                 | Node that simply listens on each port and indiscriminately regenerates the signal at all other ports. A hub (L1) operates at physical layer.  
Synonym: repeater |
| Hop                 | Traversal of a single Link by a data Packet.  
See also: Link, Mesh Network |
| Installer           | A person or group of persons that installs OpenAIS devices in a building; mounting these in the ceiling and connecting the wires. |

© OpenAIS Consortium – Public Document
<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Destination</td>
<td>Any Node which is the destination of an IP data packet. In other words, the Node hosts an application that is a consumer of an IP packet.</td>
</tr>
<tr>
<td>IP Node</td>
<td>A Node that uses the Internet Protocol (IP) for data communication</td>
</tr>
<tr>
<td>IP Router</td>
<td>A Routing Node that routes IP packets</td>
</tr>
<tr>
<td>IP Source</td>
<td>Any Node which is the source of an IP data packet. In other words, the Node hosts an application that is a generator of this IP packet.</td>
</tr>
<tr>
<td>IP Subnet</td>
<td>A logically visible subdivision of an IP network. Example: In a 6LoWPAN Mesh Network all Nodes together form a single IPv6 Subnet with each Node using an IPv6 address containing the same IPv6 Prefix.</td>
</tr>
<tr>
<td>IPv6 Prefix</td>
<td>The most significant bits of an IPv6 address, indicating the type of IPv6 address and the IP Subnet that a Node belongs to. [RFC4291]</td>
</tr>
<tr>
<td>Joining Node (JN)</td>
<td>A Node that needs to, or is executing a process to, join a Subnet in order to become part of the building Network. The joining process typically includes security protocols to ensure the Node can legitimately participate in the building Network.</td>
</tr>
<tr>
<td>Key Distribution Centre (KDC)</td>
<td>A software function that is used to provide keys for secure group communication to authorized group members.</td>
</tr>
<tr>
<td>L2 (Ethernet) Switch</td>
<td>A Switch that has Ethernet ports. See also: Switch</td>
</tr>
<tr>
<td>Light Point</td>
<td>A single light output that can be controlled. A Light Point may allow modifying its intensity, colour, beam focus, direction etc., but can’t be split in two separately controlled light outputs itself. A Luminaire integrates at least one Light Point. See also: Luminaire, Light Source</td>
</tr>
<tr>
<td>Light Source</td>
<td>A source of light, which in OpenAIS context is typically a module that uses LEDs. A Light Point integrates typically one Light Source. See also: Light Point, Luminaire</td>
</tr>
<tr>
<td>Lightweight M2M</td>
<td>A standard for device management over IP, defined by the OMA [LWM2M]. See also: Section 3.5.2.</td>
</tr>
<tr>
<td>Link</td>
<td>A unidirectional or bidirectional data communication capability over a wired or wireless medium, directly between two or more Nodes, without any modification, filtering or L3 routing of the data Frames by other Nodes in-between.</td>
</tr>
<tr>
<td>Localization</td>
<td>Measuring or determining the location of a Device in a building/room.</td>
</tr>
<tr>
<td>Luminaire</td>
<td>A Device that incorporates one or more Light Points. A Luminaire may also incorporate sensors or Converters. See also: Light Point</td>
</tr>
<tr>
<td>Machine-to-Machine (M2M) communication</td>
<td>Autonomous data communication between a Node and Application Server(s), which doesn’t require human intervention during normal operation. Normally an M2M device takes initiative to report data or events to an Application Server. Also the Server could take initiative to configure parameters on the device/Node or to request a report about something. Example: a sensor Node reporting sensor data periodically to a Cloud server</td>
</tr>
<tr>
<td>Mesh Network</td>
<td>A Network having a mesh topology. In a mesh topology, Routing Nodes are typically active to enable data connectivity between all Nodes in the Mesh Network even in case a Link fails.</td>
</tr>
<tr>
<td>Mesh Link</td>
<td>A Link in a Mesh Network See also: Link, Mesh Network</td>
</tr>
<tr>
<td>Definition</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Multicast Group</td>
<td>A group of Nodes identified by an IPv6 multicast address. See also: Node Group</td>
</tr>
<tr>
<td>Neighbour Node</td>
<td>Another Node that is one Hop separated from a Node</td>
</tr>
<tr>
<td>Network</td>
<td>Synonym: Segment</td>
</tr>
<tr>
<td>Network Address</td>
<td>Network function of remapping one IP address space into another by modifying network address information in Internet Protocol (IP) datagram packet headers while they are in transit across a Router. Note: NAT function is often combined with a Firewall. See also: Firewall</td>
</tr>
<tr>
<td>Node</td>
<td>A data communication function (i.e. abstraction) of a Device, used to send and/or receive data packets over Links. Alternative definition: device that is attached to a Network, and is capable of sending, receiving, or forwarding information over a Link See also: Routing Node, Border Router, End Node, Node Group</td>
</tr>
<tr>
<td>Node Group</td>
<td>A logical grouping of Nodes See also: Multicast Group Example: An IPv6 multicast group that certain Nodes have joined.</td>
</tr>
<tr>
<td>Object</td>
<td>A LWM2M Object as defined in the LWM2M specification [LWM2M], which is a type of Service.</td>
</tr>
<tr>
<td>Object Model (OM)</td>
<td>The OpenAIS Object Data Model as defined in Section 3.3.7.</td>
</tr>
<tr>
<td>Object ID</td>
<td>A (numeric) type identifier for a type of Object. Object IDs syntax conforms to the LWM2M specification [LWM2M].</td>
</tr>
<tr>
<td>Object instance</td>
<td>An instance of a LWM2M Object, which is a concrete instantiation of a Service.</td>
</tr>
<tr>
<td>Out-of-the-box</td>
<td>A specific behaviour of a lighting system or OpenAIS components which is factory-default, prior to the Commissioning phase.</td>
</tr>
<tr>
<td>Packet</td>
<td>A unit of data (i.e. group of bytes) which is transferred over a Network at L3 over one or more Links. Example: unicast IPv6 packet Synonym: Datagram</td>
</tr>
<tr>
<td>Peer-to-Peer</td>
<td>A communication pattern where a Node sends a message, or communicates with, either a single Node or a Node Group in the same Subnet or in another Subnet. Note: when peers (Nodes) communicate with each other using CoAP as the application layer protocol, then one of the peers initiating the communication using a request acts as a CoAP client and the other(s) receiving the request act as a CoAP server. Typically, an OpenAIS Device can act in a CoAP client and CoAP server role at the same time, for different purposes. Note: Node Group members may span multiple Subnets See also: Client/Server</td>
</tr>
<tr>
<td>PHY Segment</td>
<td>A Segment within the Field Network that is implemented using one and the same physical medium (PHY, L1) providing mutual network connectivity between all Nodes in the Segment. Note: a PHY Segment can contain multiple Links Synonyms: capillary network, local network, local segment</td>
</tr>
<tr>
<td>Prefix</td>
<td>Synonym: IPv6 Prefix</td>
</tr>
<tr>
<td>Definition</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Router</td>
<td>Node that ‘relays’ (receives and then re-sends) data packets to other Node(s) without altering packet content. A Router operates at Network Layer (L3). Note: the Node may maintain routing tables in order to implement the routing i.e. ‘relaying’ of packets, but this is not necessarily the case. Note: packet headers may be altered in the routing process. See also: End Node, Mesh Network</td>
</tr>
<tr>
<td>Security Group</td>
<td>A group of entities that shares the same security context, which allows group members to send messages securely to other group members and to receive messages from other group members. See also: Application Group</td>
</tr>
<tr>
<td>Sense-Control-Actuate (SCA) model</td>
<td>The basic architecture proposed for OpenAIS by WP2. The name refers to the functions Sense-Function, Control-Function and Actuate-Function.</td>
</tr>
<tr>
<td>Segment</td>
<td>Portion of a computer network that is separated from the rest of the network by a device such as a repeater, Hub, Bridge, Switch or Router. Each Segment contains one or more Nodes. Synonyms: Network</td>
</tr>
<tr>
<td>Sense-Function</td>
<td>A Function that is able to get sensor data and events from the local Device on which this function is deployed. The Sense-Function has the ability to communicate sensor data and events to interested subscribers, including Control-Functions that need this sensor data. See also: Actuate-Function</td>
</tr>
<tr>
<td>Server</td>
<td>Used in multiple meanings in this document:</td>
</tr>
<tr>
<td></td>
<td>• Server: a typically powerful networked computer/Device that provides a set of functions over the network</td>
</tr>
<tr>
<td></td>
<td>• CoAP Server: the server role of the CoAP client/server protocol [RFC 7252]</td>
</tr>
<tr>
<td></td>
<td>• LWM2M Server: the server role of a Lightweight M2M compliant device as defined in [LWM2M].</td>
</tr>
<tr>
<td>Service</td>
<td>Functionality offered by a Device for use by other entities (e.g. other Devices or other Services). A Service has a specified interface or API that can be used to gain access to it and use it. A service in the OpenAIS architecture is implemented through a LWM2M Object.</td>
</tr>
<tr>
<td>Stand-alone Network</td>
<td>A Network, part of the Field Network, that is stand-alone i.e. not connected to a Backbone Network nor to the Internet Synonyms: Stand-alone Segment, Isolated Network</td>
</tr>
<tr>
<td>Subnet</td>
<td>Synonyms: IP Subnet</td>
</tr>
<tr>
<td>Switch</td>
<td>Node that forwards Frames to the correct port based on the Data Link address (MAC address in case of Ethernet). In case the Data Link address is a broadcast address, the switch (L2) forwards the Frame to all ports (except the receiving port). A Switch (L2) operates at Data Link layer Synonyms: Bridge</td>
</tr>
<tr>
<td>Thread</td>
<td>An IPv6 (6LoWPAN) networking technology L1-L4 based on IEEE 802.15.4 mesh networking. Defined by [threadgroup.org].</td>
</tr>
<tr>
<td>Wireless Access Point</td>
<td>Synonyms: Access Point</td>
</tr>
</tbody>
</table>
## 9.3 Icons

<table>
<thead>
<tr>
<th>Wired luminaire</th>
<th>Wireless luminaire</th>
<th>Sensor</th>
<th>Light sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presence sensor</td>
<td>Dimmer</td>
<td>Network Router</td>
<td></td>
</tr>
</tbody>
</table>