Adaptive Gateways for dIverse muLtiple Environments

D2.1

Core requirements specification and IoT protocol integration

Project Acronym	AGILE	
Project Title	Adaptive Gateways for Diverse Multiple Environments	
Project Number	688088	
Work Package	WP 2 Core Module Integration & Gateway SW framework development	
Lead Beneficiary	RN	
Editor	Georgios Michalakidis	Operations Director, Resinio Ltd
Reviewer 1	Paolo Azzoni	Research Programme Manager, Eurotech
Reviewer 2	Philippe Krief	Research Relations Director, Eclipse
Reviewer 3	Prof Alexander Felfernig	Graz University of Technology
Dissemination Level	PU	
Contractual Delivery Date	30/06/2016	
Actual Delivery Date	30/06/2016 – *Resubmission 26/10/2016*	
Version	V1.1	

Abstract

This report forms part of the contractual deliverables of the AGILE H2020 *EC-funded* project. It documents the results from the analysis of available IoT Standards and protocols (for IoT and Machine-to-Machine communication) that will be integrated within AGILE for maximising the connected device support and interoperability. Support of the selected protocols (for integration in various layers such as the operating system, device management and the user-space) is assessed for features such as device discovery, communication and, additionally, utilisation of external protocols (e.g. for messaging).

In addition, the deliverable illustrates a process for auto-configuring hardware modules within the gateway, focusing on the configuration steps required during the deployment of the gateway and on the resource optimisation at runtime. This will allow the AGILE gateway to be auto-configured based on the application context and user preferences. A preliminary mechanism to implement the configurator service is documented herein. A more detailed documentation of the configuration, will be presented in deliverable D2.2 (initial version of Gateway Self-configuration, IoT Device discovery & Remote gateway management).

Document History

Version	Date	Comments
V0.1	01/06/2016	First Draft
V0.2	10/06/2016	Amendments – Section inclusion
V0.3	20/06/2016	Amendments – Section inclusion
V0.4	24/06/2016	Review and suggestions
V0.9	28/06/2016	Final draft after reviewers' comments
V1.0	30/06/2016	Deliverable ready for submission
V1.0.1	24/10/2016	Details are inserted under Section-1 for review comment "D2.1 is to be revised to include a brief introduction of the AGILE Configurator specifically addressing its adaptation to the overall architecture."
V1.0.2	25/10/2016	Review
V1.1	25/10/2016	Deliverable ready for resubmission

Table of Contents

List of Figures

List of Tables

Acronyms

Acronym	Meaning
H2020	Horizon 2020
EC	European Commission
AGILE	Adaptive Gateways for dIverse multiple Environments
IoT	Internet-of-Things
WP	Work Package
RN	Resinio Limited
PU	Public Dissemination
OS	Open Source

1 AGILE Configurator

The AGILE configuration services support other AGILE services and AGILE users in determining consistent settings (configurations) that make the overall gateway environment operable. Examples of configuration scenarios are discussed in the following paragraphs.

Differentiation between the terms "recommender" and "configurator":
Recommendation functionalities are used to recommend specific items (e.g., apps, devices, and cloud services) to users of the AGILE environment (e.g., a user of a smarthome environment based on AGILE can receive recommendations regarding interesting apps that could be additionally installed, a user of health tracking devices can receive recommendations regarding to step tracker or pulse/blood pressure monitoring applications). The determination of recommendations is based on recommendation algorithms such as collaborative and content-based filtering. Configuration functionalities are used to determine configurations (i.e., combinations of items that satisfy a given set of constraints - e.g., gateway hardware configurations for a specific application domain or gateway profile configurations – some related examples will be provided in the following paragraphs).

Architecture and Integration of AGILE Configurators:
AGILE will have two configurators which are (1) a "Resource Optimisation Configurator" and (2) a "Ramp-up Configurator" (see the Figure.1).

Figure.1 gives an overview of all the recommender and configurator services which are developed for AGILE. There are four recommendation services (app/workflow/device/cloud service) and one configuration service (ramp-up) on the server side and one configuration service (resource optimisation) on the gateway side. **Figure.2** shows AGILE configuration and recommendation services on the server side as part of the overall AGILE architecture. **Figure.3** shows the integration of the Resource Optimisation Configurator in the overall AGILE architecture. It provides a restful API on the AGILE gateway's static server that can be accessed by the other AGILE services.

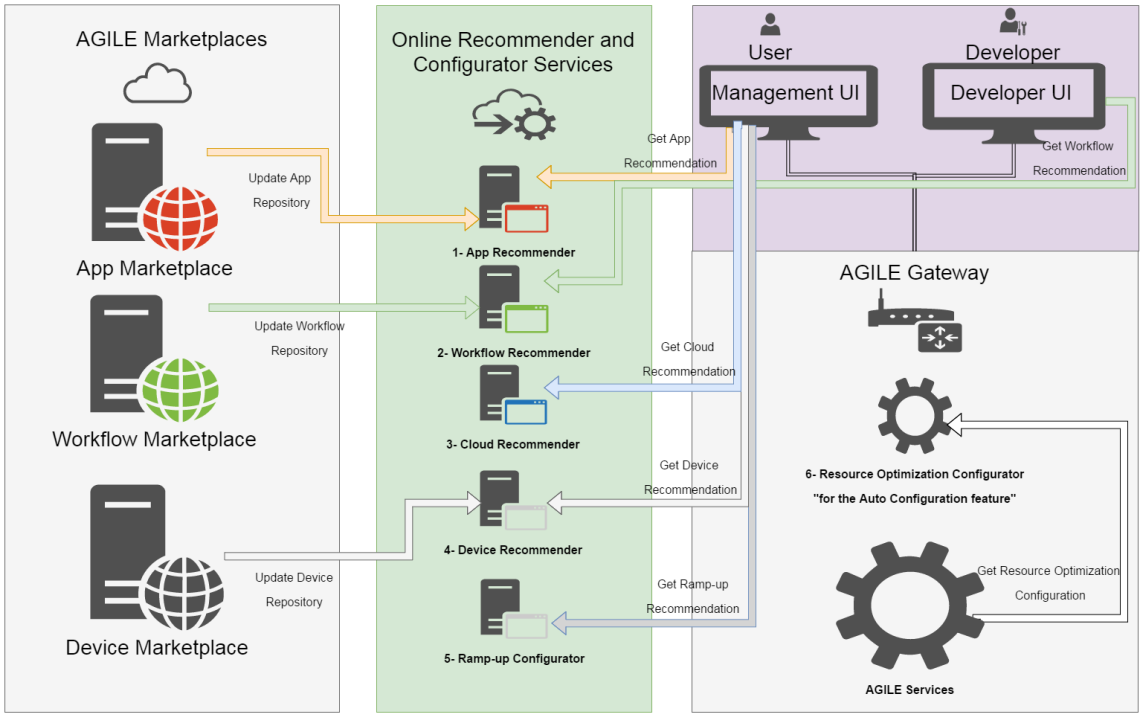

Figure 1 Recommender and Configurator Services

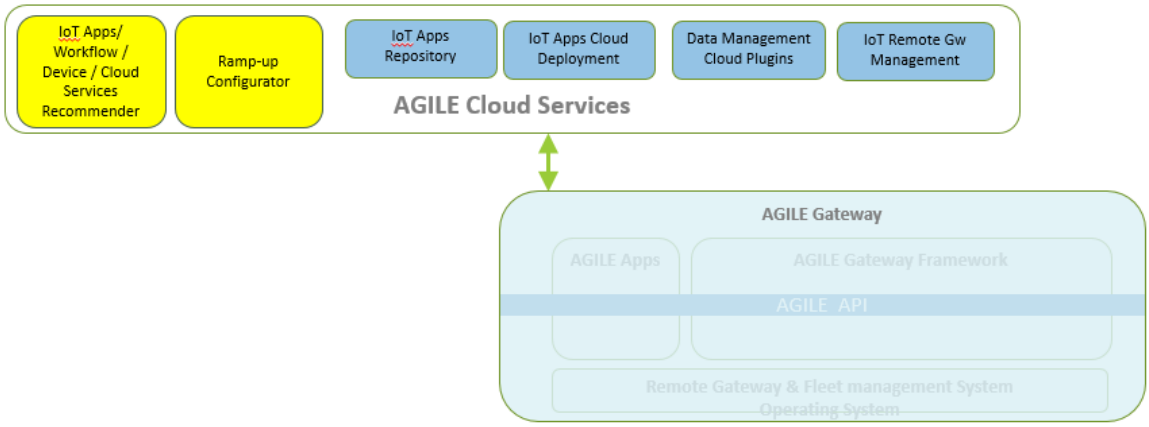

Figure 2 Online Recommender and Configurator Services are shown in yellow boxes in the AGILE Cloud Services Architecture

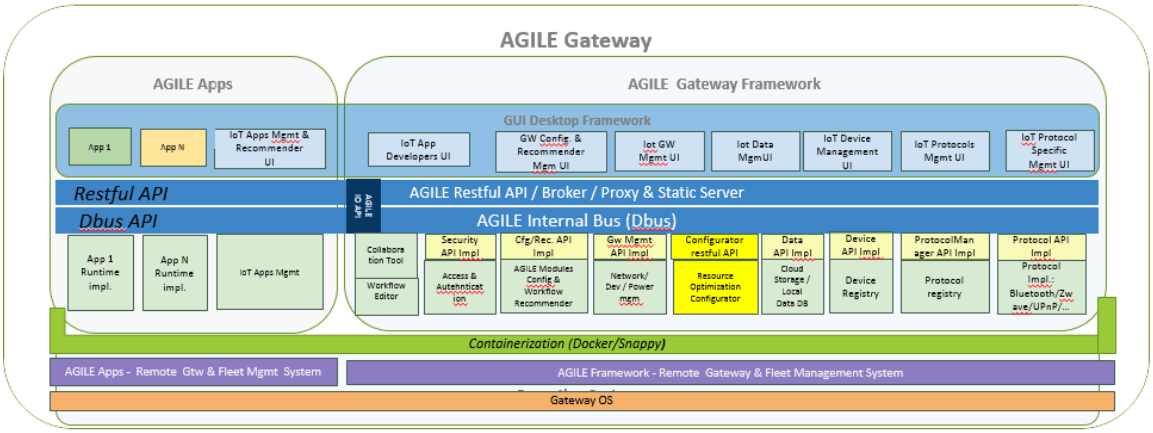

Figure 3 Configurator Service is shown as the yellow box in the AGILE Gateway Architecture

The ramp-up configurator will run on the server side as a web application (see Figure.2) to support AGILE users with regard to environmental settings (e.g., which and how many sensors should be used and where to locate these and which apps should be used). The resource optimisation configurator ("auto configuration") will run on the gateway as an AGILE service on the restful API (see Figure.3). It can be activated on the basis of information provided by the AGILE Device Manager or Component Manager. According to predefined constraints and resources (apps and devices) on the gateway, it will determine (re-) configurations (e.g., enabling, disabling devices and changing the network protocol etc.). The CSP (Constraint Satisfaction Problem) Solver "Choco" (choco-solver.org) will be used by AGILE Services to determine needed adaptations (reconfigurations) in the gateway profile. The Resource Optimisation Configurator itself is not in charge of executing reconfigurations. This will be done by the AGILE Services that activate the Resource Optimisation Configurator.

AGILE Configurator use cases:

Resource Optimisation Configurator is triggered by AGILE services such as Device Manager or Component Manager. Potential triggers include the following: "New App is installed/uninstalled", "New IoT Device is plugged/unplugged", "New Network Enabler Device is plugged/unplugged", "New driver is installed/uninstalled", "Memory Consumption is increased/decreased 10%", "CPU Consumption is increased/decreased 10%" and "Power Mode is changed". In such cases, the Resource Optimisation Configurator can be triggered by other AGILE Services to recalculate the best possible (re-)configuration. Depending on the defined knowledge base (represented as a constraint satisfaction problem describing properties of network and data encoding protocols), some or all of the mentioned scenarios can be supported by the configurator. The main aim of the configurator is to optimise the resource allocation on the AGILE gateway taking into account the preferences related to "Memory/CPU/Power Consumption". Example input data for the configurator includes:

- Gateway Profile Information: for example, current Memory/CPU/Power consumption, installed apps, connected IoT devices, and network enabler devices
- User Preferences: for example, power saving mode should be enabled/disabled.

Ramp-up Configurator is a web application which is used by AGILE users. It will be running as standalone on TU Graz server. Ramp-up Configurator can support users by providing the configuration recommendations such as which sensors to use, how many sensors to use, where to place them, etc. We will support Pilot-C with this configurator. Later on, other pilots also can generate their own constraints for the Ramp-up Configurator to be able to support their customers as well. Pilot application owners will have an admin access right to manage their constraints on the system.

The **Ramp-up Configurator** is a standalone web application which is used by AGILE users (see Figure.1 and Figure.2). This Configurator can support users by providing the configurations such as “which sensors to use”, “how many sensorsensors to use”, or “where to place them”. We selected Pilot-C (Air Quality & Pollution Monitoring) as a use case for this configurator. Later on, other AGILE-based scenarios can exploit the configurator for supporting the ramp-up process.

1.1 Configurator Engine

The Configurator Engine is an AGILE module responsible for the calculation of the constraints and inconsistencies of the AGILE Gateway configuration and for providing a possible solution accordingly. The Configurator Engine of AGILE will be responsible for two main functionalities: the "Ramp-up Configuration" and the "Resource Optimisation".

The **Ramp-up Configuration**, provides the optimum set-up configuration of the AGILE Gateway depending on the environment, the application and the user preferences. A very simple scenario that illustrates this role is the Smart Home Configuration:

The configurator takes into account the specific features of the living environment a user is planning the smart home system for. To support users initial installation of smart home devices, Configurator interactively asks to the user about his preferences. First of all, the user enters the dimensions and features of his house (floors, garden, size, # of rooms, etc.) Then the Configurator also asks some specific questions and get user preferences.

According to all the information collected about the home and user preferences, the configurator calculates conflicts and finds the best suitable option for the user requirements and provides the results. For example: "According to your preferences and home features, wireless systems would be less expensive. Wired system would cost around 3000 dollars. "

The **Resource Optimisation**, provides the optimum setting of the gateway according to the gateway profile (available Memory, CPU, power source, etc.). For example, it may enable/disable the network modules according to the optimisation strategy.

This deliverable illustrates the technologies that will be adopted for the design and development of the configurator, the definition of the configuration process and the description on how it applies to the pilots. The requirements identified during the pilot analysis (WP8) has been considered during the technologies study and during the design of the configuration process:

this approach ensures that the configuration module satisfies the practical requirements of vertical application.

In the following sections, the “Smart Home Configuration” is used as an example to easily and clearly explain the functionalities and capabilities of the ramp-up configurator.

1.1.1 Ramp-up Configuration

AGILE gateways will be deployed in different domains such as health monitoring, animal monitoring in wildlife areas, air quality and pollution monitoring, enhanced retail services, smart homes, and port area monitoring. Each application scenario requires a pre-configuration which estimates the needed hardware and software components/devices to be deployed in the ramp-up phase of the system. We denote this type of configuration ramp-up configuration since each scenario requires a specific set of hardware components and software components (including apps) to ”ramp-up” the system.

Pilot Scenario: Air Quality and Pollution Monitoring

Environmental pollution has become an issue of serious international concern and is increasingly stimulating the development and adoption of solutions to monitor and reduce the effects of pollution. This is an interesting and challenging market, with both potential economical outcomes and a strong societal impact.

Environmental monitoring has a long history of methodologies and technological solutions, unfortunately characterised by high development and maintenance costs, low territorial coverage and complex certifications: these barriers have confined the diffusion of high-end monitoring solutions to a limited set of vertical context, typically managed by the public authorities.

The convergence of hardware integration, reduction of sensor costs, IoT and M2M technologies introduces a new panorama where it is really possible to deliver low cost, high quality monitoring systems with a capillary coverage of the territory. This convergence leads to a new era of solution for environmental pollution monitoring.

The importance of the pollution monitoring domain, inspired AGILE project to focused one of the pilots on this topic. The pilot is responsible to deliver an AGILE based solution for air quality and pollution monitoring in industrial and public environments. The pilot is based on a network of monitoring stations, based on the AGILE platform, that provides multimodal, multisource and pervasive monitoring of air quality and pollution. The pervasive nature of this solution is based on EDC, a cloud integration platform from Eurotech, that cooperates with the AGILE platform to provide final user and B2B services.

Air quality and pollution monitoring stations are complex systems that, depending on the application context, deliver added value pollution monitoring services based on a delicate

equilibrium between the adoption of the most appropriate sensors, their correlation, the selection of the correct algorithms and the configuration of their hardware and software parameters. A wrong or imprecise selection and configuration of these elements leads to misleading, wrong and completely useless results and services.

The configuration process is fundamental for the entire lifecycle of the monitoring station, from the design and assembly phases, to the deployment of the system, to the day by day operation, to the maintenance. Depending on the specific deployment context, it is fundamental to ensure the correct configuration of the hardware and software components of the monitoring station in each of these lifecycle phases.

Furthermore, If the configuration process is assisted or automatic, it enables a real pervasive territorial monitoring that, with the current technologies and costs, will remain for a long time a mirage: the massive diffusion of a low cost-high quality pollution monitoring solution, with a pervasive and intelligent nature, will have an important environmental and societal impact.

During the design and assembly phase the configurator provides support for the dimensioning of the monitoring station, selection of sensors, pre-processing algorithms, environmental positioning and installation procedure. In this phases, it simplifies the cooperation between sales and customer during the requirements analysis, the identification of the final design and the definition of the commercial offer. R&D engineers are supported in the validation of the final design of the monitoring station. Also the company that will be responsible for the maintenance activities has an important role in this phase, because it provides to the configurator fundamental information related the maintenance plan: the maintenance costs are a critical aspect that is currently responsible for the limited diffusion of the pollution monitoring technologies. In these phases the configurator collects information about the deployment environment, the type of installation, the available power source, the monitoring profile, the application domain, etc.. Analysing this information, the configurator automatically defines the configuration of the final design of the monitoring station. Furthermore, during the assembly phase, it supports the operator in assembling the monitoring station.

During the deployment phase, the configurator assists the operator in the installation and configuration process. In this phase, the configurator is responsible for the identification of the correct configuration of the networking and sensing modules of the monitoring station, of the sensors calibration, of the configuration of the pre-processing algorithms, of the operating system modules and of the AGILE platform configuration.

Finally, at runtime, during day by day operation, the configurator is fundamental to support and ensure the adaptive behaviour of the monitoring station. During this phase of the lifecycle, the configurator significantly contributes to the autonomy of the monitoring station providing the configuration information required for periodical runtime calibration, dynamic configuration required by context changes, operation monitoring and maintenance support. During day by day operation, the configurator collects the required information autonomously from the live knowledge based stored in the monitoring station.

Other AGILE Scenarios

The AGILE project explores the adoption of the AGILE platform in other challenging scenarios: health monitoring, wild life monitoring, enhanced retail services and port area monitoring.

The basic task of a configurator in health monitoring is to figure out which measuring devices are needed (including their parametrisation) in order to monitor and analyse specific body functions.

In the wild life monitoring scenario it is important to figure out which infrastructure can be used to complete predefined data collection tasks for the monitored animals. In such scenarios, reachability of animals (and corresponding sensors) plays a major role in order to be able to complete data collection. Reachability depends on the selected drone types but also on the selected communication protocols which have different degrees of power consumption.

Enhanced retail services, that allow a personalised shopping experience in physical stores, requires configuration functionality that provides an indication of the amount and positioning of sensors (e.g., for indoor position detection) and of displays, devices that are needed to successfully support customers in their shopping experience.

In the port area monitoring scenario, configuration technologies are needed to drive the selection of relevant sensors (e.g., gas, radioactivity, and water quality sensors) that are able to provide the required data.

Knowledge Acquisition & Representation

The configuration process relies on the knowledge acquired from the user to suggest a configuration. Therefore, knowledge acquisition and representation represents two important factors for this AGILE module. In AGILE, we will evaluate the applicability of different types of configuration knowledge representations such as answer set programs (ASP) and constraint-based representations. Our aim is to identify a knowledge representation language that can be applied for each of the different application scenarios in order to provide a basic technology for supporting IoT ramp-up configuration tasks. The applicability of these languages will be primarily evaluated with regard to expressiveness and reasoning efficiency. Especially, ASP-based configuration approaches will be evaluated with regard to their applicability in typical gateway ramp-up scenarios.

When configuring, for example, smart homes, the configuration model includes information about the relationships between building properties and corresponding sensors (e.g., if a room is a kitchen and includes an oven, then a corresponding temperature sensor has to be included for the room) or between user preferences and the corresponding technical infrastructure (e.g., if a user wants to save money, wireless communication is preferred). A configuration for a given

configuration task includes information about which components, devices, and drivers are part of the initial gateway installation.

Consistency Management of the Knowledge Base

A configuration Knowledge Base can become inconsistent, i.e., the defined component types and constraints lead to the problem that no solution can be identified. Such a situation can occur in the context of regression testing but also in situations where the conflict is induced by the configuration knowledge base itself. In such scenarios, configuration technologies in combination with model-based diagnosis can be exploited to automatically identify the sources (e.g., constraints) of a given inconsistency. Such functionalities will be included in a development environment for IoT configuration knowledge bases.

In the context of AGILE, we focus on the development of techniques that help to improve the efficiency of configuration knowledge engineering processes. Although automated debugging is a useful means to reduce time efforts related knowledge base development and maintenance, the development and maintenance of related test cases is still costly. We will analyse the applicability of different testing approaches from software engineering and will especially focus on the development of mutation testing approaches for knowledge bases. In this context, a mutation will serve as a basis for generating tests that are, for example, accepted by the original knowledge base but should not.

Consistency Management of User Requirements

Consistency management not only plays a role in the context of knowledge base development and maintenance but also within the scope of a configuration process. A user of an AGILE configurator could articulate a set of requirements in such a way that no solution can be identified.

Also in such a situation, model-based diagnosis approaches can be exploited to indicate sets of user requirements that have to be adapted in order to identify at least one solution. A similar situation occurs in the context of reconfiguration, i.e., in a situation where hardware and software components of an IoT gateway have to be adapted. In this context, minimal changes have to be proposed that indicate how the existing configuration has to be adapted such that a consistent configuration can be determined, taking into account all reconfiguration requirements.

In AGILE, we focus on the development of personalisation techniques that help to improve the diagnosis prediction quality, i.e., to identify those diagnoses that will be accepted by the user. Such personalised diagnoses will be determined on the basis of an analysis of the interaction behaviour of users of similar gateway installations (available in gateway profile repositories). In this context we will develop learning-based approaches that help to calibrate search heuristics in order to improve efficiency and prediction quality of configuration and reconfiguration.

A simple example of our envisioned approach is the following. Let us assume the existence of a configuration log as the one shown in Table 1. The parameters reqi indicate user requirements and xi indicate technical product parameter settings (consistent with the user requirements) accepted by the user UI. The overall goal is to optimise the configurator search heuristics (e.g., variable and domain orderings) in such a way that the prediction quality for the technical parameter settings is maximised. More precisely, we want to identify search heuristics that guide to solutions (configurations) that will be accepted by the current user. User interactions (see, e.g., Table 1) serve as a basis for learning. Prediction quality can be measured, for example, in terms of the user acceptance degree of parameter settings (configurations) proposed by the configurator. In this context we will evaluate different clustering techniques, i.e., to learn heuristics not on a global level, but depending on a specific cluster derived, for example, from the user requirements.

$user$	req_1	req_2	x_1	x_2	x_3	x_4
u_1	1	2	3	4	4	2
u_2	2	2	8	3	4	2
u_3	1	2	3	4	5	2
$current$	1	1	?	?	?	?

Table 1 – Example of configuration log

1.1.2 Resource Optimisation

Modern embedded systems included in IoT scenarios support a rich set of connectivity solutions (e.g., 3G, LTE, TD-LTE, FDD-LTD, WIMAX, and Lora). In this context, configuration technologies play an important role in terms of suggesting optimal connectivity configurations.

In AGILE, runtime configuration must be performed on the gateway – in contrast, ramp-up configuration can also take place in the cloud. On the one hand we will evaluate different types of reasoning engines, for example, the CHOCO constraint solver and the Sat4j boolean satisfaction library. We will also take into account the application of rule engines, optimisation libraries, and knowledge compression techniques to assure efficiency of problem solving on the gateway level.

A simplified example of the application of a utility-based approach is the following. Table 2 includes an evaluation of connectivity protocol configurations conf (confa and confb) to be used on the gateway, for example, for different types of data exchange. The three evaluation dimensions used in this example are performance, reliability, and costs. Furthermore, Table 3 includes the personal preferences of two different gateway users (u1 and u2).

configuration	perfor-mance	reliability	costs
$conf_a$	9	5	2
$conf_b$	5	8	3

Table 2 – Utility table: evaluation of configurations with regard to the interest dimensions performance, reliability, and costs

user	performance	reliability	costs
u_1	10	3	1
u_2	5	7	10

Table 3 – Example of user preferences w.r.t. interest dimensions performance, reliability, and costs

In order to determine the configuration that should be chosen for a specific user, we can apply a utility function (see, e.g., the following formula).

$$utility(conf, u) = \Sigma_{d \in dim} interest(u, d) \times value(conf, d) \quad (1)$$

In this context, utility(conf,u) denotes the utility of the configuration conf for the user u, interest(u,d) denotes the interest of user u in evaluation dimension d, and value(conf,u) denotes the contribution of configuration conf to the interest dimension d. In the example, configuration confa has a higher utility for user u1 (107:0) whereas configuration confb has a higher utility for u2 (111:0). Note that for simplicity we omitted to sketch the determination of the evaluations depicted in Table 2. In order to increase the efficiency of runtime configuration, we will evaluate knowledge compression techniques that help to reduce search efforts as much as possible. For example, we will apply decision diagram techniques to pre-calculate possible configurations and re-configurations.

Table 4 provides a summary of the configuration-related research objectives in AGILE. Within the context of ramp-up configuration scenarios we will identify knowledge representation mechanisms that allow an easy representation of the AGILE IoT domains introduced.

Furthermore, we will develop test case generation techniques that will help to make the development and management of test cases more efficient. For AGILE scenarios, we will develop concepts that support the learning of search heuristics to optimise configuration and reconfiguration processes. Furthermore, we will work on knowledge compression techniques that help to make solution search on the gateway level as efficient as possible.

configuration topic	research objective
appropriate knowledge representations	knowledge representations for easy modeling and efficient configuration search
efficiency of knowledge base development and maintenance	automated test case generation and mutation testing
personalized consistency management	personalized configuration based on learning search heuristics and knowledge compression techniques

Table 4 – Overview of AGILE research objectives

2 Analysis and Integration of IoT Standards

This chapter aims at giving an overview of the IoT/M2M standards and protocols that will be integrated and supported within the AGILE Platform during the project or by the developers community.

Protocols and standards have been classified in 6 different categories:

- Connectivity Protocols
- Messaging Protocols
- Data Encoding Protocols
- Remote management Protocols
- Security Protocols and Standards
- Full-stack Standards

For each category, matching criteria have been identified to help identify the strengths and weaknesses of each protocol/standard.

A detailed comparative study of each protocol/standard with regard to the identified criteria is documented in the Appendix.

2.1 Connectivity Protocols

The main goal of AGILE is to enable various types of devices (wearables, home appliances, sensors, actuators, etc.) to be connected with each other and to the Internet. Therefore, it is critical that the partners select the **connectivity protocols** that will maximise the number of assets that can be part of the AGILE ecosystem.

2.1.1 Evaluation Criteria

The following criteria have been adopted for the selection of the connectivity protocols:

- **Wired or wireless**

- **Topology / LAN vs. WAN**: Some connectivity standards may be more suited for implementing the communication between an AGILE gateway and a cloud infrastructure, while others are more appropriate for inter-asset or inter-gateway communication. AGILE will need to support both kind of scenarios.

- **Max range**: There are cases, in particular in the “Open field & cattle monitoring” pilot, where an AGILE gateway will be located very far away from sensors. Therefore it’s important to get a good understanding of the typical range for each connectivity protocol. For wireless

protocols, a difference should be made between range for indoors scenario vs. “line of sight”/outdoors situations.

- **Max throughput**: The maximum throughput of a connectivity protocol is often correlated to the maximum range. A throughput in the range of megabytes per second might be desirable for transferring video streams, while a few kilobytes per second might be ample for simple telemetry use cases.

- **Security**: Some connectivity protocols (typically the ones corresponding to industrial protocols that have been retrofitted for IoT) have no security mechanism at the transport level. It may not be an issue for some use cases (wired, with the operator being in full physical control of the solution), but we believe that it is important to favour those protocols that make it more difficult to eavesdrop, impersonate assets, etc.

2.1.2 Protocol Selection

Driven by the previous criteria, the selection process allowed the identification of the following connectivity protocols:

6LoWPAN

Link: https://en.wikipedia.org/wiki/6LoWPAN

6LoWPAN is an acronym of IPv6 over Low-Power Wireless Personal Area Networks.

The 6LoWPAN group has defined encapsulation and header compression mechanisms that allow IPv6 packets to be sent and received over IEEE 802.15.4 based networks.

The target for IP networking for low-power radio communication are applications that need wireless internet connectivity at lower data rates for devices with very limited form factor. An example is automation and entertainment applications in home, office and factory environments. The header compression mechanisms standardised in RFC6282 can be used to provide header compression of IPv6 packets over such networks.

IPv6 is also in use on the smart grid enabling smart meters and other devices to build a micro mesh network before sending the data back to the billing system using the IPv6 backbone.

802.15.4

Link: https://en.wikipedia.org/wiki/IEEE_802.15.4

IEEE 802.15.4 is a standard created and maintained by consultants which specifies the physical layer and media access control for low-rate wireless personal area networks (LR-WPANs).

It is the basis for the ZigBee, ISA100.11a, WirelessHART, MiWi, and Thread specifications, each of which further extends the standard by developing the upper layers which are not defined in IEEE 802.15.4. Alternatively, it can be used with 6LoWPAN as Network Adaptation Layer and

standard Internet protocols and/or IETF RFCs defining the upper layers with proper granularity to build a wireless embedded Internet.

IEEE standard 802.15.4 intends to offer the fundamental lower network layers of a type of wireless personal area network (WPAN) which focuses on low-cost, low-speed ubiquitous communication between devices. It can be contrasted with other approaches, such as Wi-Fi, which offer more bandwidth and require more power. The emphasis is on very low cost communication of nearby devices with little to no underlying infrastructure, intending to exploit this to lower power consumption even more.

The basic framework conceives a 10-meter communications range with a transfer rate of 250 kbit/s. Tradeoffs are possible to favor more radically embedded devices with even lower power requirements, through the definition of not one, but several physical layers. Lower transfer rates of 20 and 40 kbit/s were initially defined, with the 100 kbit/s rate being added in the current revision.

IEEE 802.15.4-conformant devices may use one of three possible frequency bands for operation (868/915/2450 MHz).

Bluetooth 4.0

Link: https://en.wikipedia.org/wiki/Bluetooth#Bluetooth_v4.0

The Bluetooth SIG completed the Bluetooth Core Specification version 4.0 (called Bluetooth Smart) and has been adopted as of 30 June 2010. It includes Classic Bluetooth, Bluetooth high speed and Bluetooth low energy protocols. Bluetooth high speed is based on Wi-Fi, and Classic Bluetooth consists of legacy Bluetooth protocols.

The key new feature of Bluetooth 4.0 is its low-energy technology. This lets device manufacturers replace proprietary sensor technology with Bluetooth, which is a more widely adopted standard. An obvious example is in the health and fitness category. Most pedometers, heart rate straps, and blood glucose monitors are designed to only talk to a specific wristwatch or control unit. If these same devices had Bluetooth 4.0, they could speak to any Bluetooth 4.0 device, be it phone or computer, without requiring an intermediary.

Bluetooth Smart (BLE)

Link: https://en.wikipedia.org/wiki/Bluetooth_low_energy

Bluetooth low energy (Bluetooth LE, BLE, marketed as Bluetooth Smart) is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries. Compared to Classic Bluetooth, Bluetooth Smart is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range.

Bluetooth Smart was originally introduced under the name Wibree by Nokia in 2006. It was merged into the main Bluetooth standard in 2010 with the adoption of the Bluetooth Core Specification Version 4.0.

Mobile operating systems including iOS, Android, Windows Phone and BlackBerry, as well as OS X, Linux, and Windows 8, natively support Bluetooth Smart. The Bluetooth SIG predicts that by 2018 more than 90 percent of Bluetooth-enabled smartphones will support Bluetooth Smart.

enOcean

Link: https://en.wikipedia.org/wiki/EnOcean

The EnOcean technology is an energy harvesting wireless technology used primarily in building automation systems, and is also applied to other applications in industry, transportation, logistics and smart homes. Modules based on EnOcean technology combine micro energy converters with ultra low power electronics, and enable wireless communications between battery-less wireless sensors, switches, controllers and gateways.

The energy harvesting wireless modules are manufactured and marketed by the company EnOcean which is based in Oberhaching, Germany. EnOcean offers its technology and licenses for the patented features within the EnOcean Alliance framework.

EnOcean-based products (such as sensors and light switches) perform without batteries and are engineered to operate maintenance-free. The radio signals from these sensors and switches can be transmitted wirelessly over a distance of up to 300 meters in the open and up to 30 meters inside buildings. Early designs from the company used piezo generators, but were later replaced with electromagnetic energy sources to reduce the operating force (3.5 newtons), and increase the service life to 100 operations a day for more than 25 years.

EnOcean wireless data packets are relatively small, with the packet being only 14 bytes long and are transmitted at 125 kbit/s. RF energy is only transmitted for the 1's of the binary data, reducing the amount of power required. Three packets are sent at pseudo-random intervals reducing the possibility of RF packet collisions. Modules optimised for switching applications transmit additional data packets on release of push-button switches, enabling other features such as light dimming to be implemented.[2] The transmission frequencies used for the devices are 902 MHz, 928.35 MHz, 868.3 MHz and 315 MHz.

Ethernet

Link: https://en.wikipedia.org/wiki/Ethernet

Ethernet is a family of computer networking technologies commonly used in local area networks (LANs) and metropolitan area networks (MANs). It was commercially introduced in 1980 and first standardised in 1983 as IEEE 802.3, and has since been refined to support higher bit rates and longer link distances. Over time, Ethernet has largely replaced competing wired LAN technologies such as token ring, FDDI and ARCNET.

The original 10BASE5 Ethernet uses coaxial cable as a shared medium, while the newer Ethernet variants use twisted pair and fibre optic links in conjunction with hubs or switches. Over the course of its history, Ethernet data transfer rates have been increased from the original 2.94 megabits per second (Mbit/s) to the latest 100 gigabits per second (Gbit/s), with 400 Gbit/s expected by late 2017. The Ethernet standards comprise several wiring and signalling variants of the OSI physical layer in use with Ethernet.

Systems communicating over Ethernet divide a stream of data into shorter pieces called frames. Each frame contains source and destination addresses, and error-checking data so that damaged frames can be detected and discarded; most often, higher-layer protocols trigger retransmission of lost frames. As per the OSI model, Ethernet provides services up to and including the data link layer.

KNX

Link: https://en.wikipedia.org/wiki/KNX_%28standard%29

KNX is a standardised (EN 50090, ISO/IEC 14543), OSI-based network communications protocol for building automation. KNX is the successor to, and convergence of, three previous standards: the European Home Systems Protocol (EHS), BatiBUS, and the European Installation Bus (EIB or Instabus).

The standard is based on the communication stack of EIB but enlarged with the physical layers, configuration modes and application experience of BatiBUS and EHS.

KNX defines several physical communication media:

- Twisted pair wiring (inherited from the BatiBUS and EIB Instabus standards)
- Powerline networking (inherited from EIB and EHS - similar to that used by X10)
- Radio (KNX-RF)
- Infrared
- Ethernet (also known as EIBnet/IP or KNXnet/IP)

KNX is designed to be independent of any particular hardware platform. A KNX Device Network can be controlled by anything from an 8-bit microcontroller to a PC, according to the needs of a particular implementation. The most common form of installation is over twisted pair medium.

LoRa

Link: https://en.wikipedia.org/wiki/LPWAN

LoRa, proprietary, CSS modulation technology used for LPWAN patented by Semtech by LoRa Alliance used by LoRaWAN and Symphony Link.

Low-Power Wide-Area Network (LPWAN) or Low-Power Network (LPN) is a type of wireless telecommunication network designed to allow long range communications at a low bit rate among things (connected objects), such as sensors operated on a battery.

NB-IoT

Narrow-Band IOT (NB-IOT) is a technology being standardised by the 3GPP standards body. This technology is a narrowband radio technology specially designed for the Internet of Things (IoT), hence its name. Special focus of this standard are on indoor coverage, low cost, long battery life and large number of devices. This technology can be deployed in GSM and LTE spectrum.

Near Field Communication (NFC)

Link: https://en.wikipedia.org/wiki/Near_field_communication

Near field communication (NFC) is a set of communication protocols that enable two electronic devices, one of which is usually a portable device such as a smartphone, to establish communication by bringing them within 4 cm (2 in) of each other.

NFC-enabled portable devices can be provided with apps, for example to read electronic tags or make payments when connected to an NFC-compliant apparatus. Earlier close-range communication used technology that was proprietary to the manufacturer, for applications such as stock ticket, access control and payment readers.

"Raw" RF

The Industrial, Scientific and Medical (ISM) unlicensed frequency bands below 1 GHz are widely used by wireless communication systems. So-called *sub-gigahertz* RF wireless solutions are usually very low cost, and allow to setup simple unicast communication scenarios.

RS-232

Link: https://en.wikipedia.org/wiki/RS-232

In telecommunications, RS-232 is a standard for serial communication transmission of data. It formally defines the signals connecting between a DTE (data terminal equipment) such as a computer terminal, and a DCE (data circuit-terminating equipment or data communication equipment), such as a modem. The RS-232 standard is commonly used in computer serial ports. The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors.

RS-485

Link: https://en.wikipedia.org/wiki/RS-485

RS-485 enables the configuration of inexpensive local networks and multidrop communications links. It offers data transmission speeds of 35 Mbit/s up to 10 m and 100 kbit/s at 1200 m. Since it uses a differential balanced line over twisted pair (like RS-422), it can span relatively large distances up to 1,200 m (4,000 ft). A rule of thumb is that the speed in bit/s multiplied by the length in meters should not exceed 108. Thus a 50 meter cable should not signal faster than 2 Mbit/s.

In contrast to RS-422, which has a single driver circuit which cannot be switched off, RS-485 drivers need to be put in transmit mode explicitly by asserting a signal to the driver. This allows RS-485 to implement linear bus topologies using only two wires. The equipment located along a set of RS-485 wires are interchangeably called nodes, stations or devices.

Sigfox

Sigfox is a wireless communication protocol that uses frequencies in the unlicensed ISM band (915MHz in the US, 868 MHz in Europe) to send very small amounts of data (12 bytes), very slowly (300 baud) using standard radio transmission methods (phase-shift keying).

Wi-Fi

Link: https://en.wikipedia.org/wiki/Wi-Fi

Wi-Fi or WiFi is a technology that allows electronic devices to connect to a wireless LAN (WLAN) network, mainly using the 2.4 gigahertz (12 cm) UHF and 5 gigahertz (6 cm) SHF ISM radio bands. Access to a WLAN is usually password protected, but may be open, which allows any device within its range to access the resources of the WLAN network.

Devices which can use Wi-Fi technology include personal computers, video-game consoles, smartphones, digital cameras, tablet computers and digital audio players. Wi-Fi compatible devices can connect to the Internet via a WLAN network and a wireless access point. Such an access point (or hotspot) has a range of about 20 meters (66 feet) indoors and a greater range outdoors. Hotspot coverage can be as small as a single room with walls that block radio waves, or as large as many square kilometres achieved by using multiple overlapping access points.

Zigbee

Link: https://en.wikipedia.org/wiki/ZigBee

ZigBee is an IEEE 802.15.4-based specification for a suite of high-level communication protocols used to create personal area networks with small, low-power digital radios.

The technology defined by the ZigBee specification is intended to be simpler and less expensive than other wireless personal area networks (WPANs), such as Bluetooth or Wi-Fi. Applications include wireless light switches, electrical meters with in-home-displays, traffic management systems, and other consumer and industrial equipment that requires short-range low-rate wireless data transfer.

Its low power consumption limits transmission distances to 10–100 meters line-of-sight, depending on power output and environmental characteristics. ZigBee devices can transmit data over long distances by passing data through a mesh network of intermediate devices to reach more distant ones. ZigBee is typically used in low data rate applications that require long battery life and secure networking (ZigBee networks are secured by 128 bit symmetric encryption keys.) ZigBee has a defined rate of 250 kbit/s, best suited for intermittent data transmissions from a sensor or input device.

2.2 Messaging Protocols

On top of the connectivity protocols are **messaging protocols** that aim at enabling unicast, multicast and broadcast kind of communications between things/devices. In AGILE, we want to rely on messaging protocols that are broadly adopted (ecosystem of users and developers), and that are open.

2.2.1 Evaluation Criteria

The following criteria have been adopted for the selection of the messaging protocols:

- **Discovery support**: Does the messaging protocol include discovery mechanisms (e.g. for an asset/sensor to discover its gateway, or for an asset/sensor to discover its "neighbours").
- **Security**
- **Open source community**: The messaging protocol(s) used in an IoT solution control how easy it is for third parties to integrate with it. It is important to evaluate the messaging protocols supported in AGILE according to the number (and activity) of open source implementations available for these protocols.
- **Open standard**: Is the standard open and royalty-free.
- **Whether or not they are used in existing commercial solutions**

2.2.2 Protocol Selection

Driven by the previous criteria, the selection process allowed the identification of the following messaging protocols.

CoAP

Link: https://en.wikipedia.org/wiki/Constrained_Application_Protocol

Constrained Application Protocol (CoAP) is a software protocol intended to be used in very simple electronics devices that allows them to communicate interactively over the Internet. It is particularly targeted for small low power sensors, switches, valves and similar components that need to be controlled or supervised remotely, through standard Internet networks. CoAP is an application layer protocol that is intended for use in resource-constrained internet devices, such as WSN nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialised requirements such as multicast support, very low overhead, and simplicity. Multicast, low overhead, and simplicity are extremely important for Internet of Things (IoT) and Machine-to-Machine (M2M) devices, which tend to be deeply embedded and have much less memory and power supply than traditional internet devices have. Therefore, efficiency is very important. CoAP can run on most devices that support UDP or a UDP analogue.

The CoRE group has designed CoAP with the following features in mind:

- Overhead and parsing complexity.
- URI and content-type support.
- Support for the discovery of resources provided by known CoAP services.
- Simple subscription for a resource, and resulting push notifications.
- Simple caching based on max-age.

The mapping of CoAP with HTTP is also defined, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way.

With the introduction of CoAP, a complete networking stack of open standard protocols that are suitable for constrained devices and environments, becomes available.

Modbus

Link: https://en.wikipedia.org/wiki/Modbus

Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become a de facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices. The main reasons for the use of Modbus in the industrial environment are:

- developed with industrial applications in mind
- openly published and royalty-free
- easy to deploy and maintain
- moves raw bits or words without placing many restrictions on vendors

Modbus enables communication among many devices connected to the same network, for example a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from its use in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact.

MQTT

Link: https://en.wikipedia.org/wiki/MQTT

MQTT (formerly MQ Telemetry Transport) is an ISO standard (ISO/IEC PRF 20922) publish-subscribe based "light weight" messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker. The broker is responsible for distributing messages to interested clients based on the

topic of a message. Andy Stanford-Clark and Arlen Nipper of Cirrus Link Solutions authored the first version of the protocol in 1999.

There are several MQTT brokers available such as ActiveMQ, Apollo, HiveMQ, IBM MessageSight, JoramMQ, Mosquitto, RabbitMQ, Solace Message Routers, and VerneMQ. They vary in their feature set and some of them implement additional features on top of the standard MQTT functionality.

MQTT defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

- Connect: Waits for a connection to be established with the server.
- Disconnect: Waits for the MQTT client to finish any work it must do, and for the TCP/IP session to disconnect.
- Subscribe: Waits for completion of the Subscribe or UnSubscribe method.
- UnSubscribe: Requests the server unsubscribe the client from one or more topics.
- Publish: Returns immediately to the application thread after passing the request to the MQTT client.

MQTT-SN

Link: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf

MQTT-SN is designed to be as close as possible to MQTT, but is adapted to the peculiarities of a wireless communication environment such as low bandwidth, high link failures, short message length, etc. It is also optimised for the implementation on low-cost, battery-operated devices with limited processing and storage resources.

WebSockets

Link: https://en.wikipedia.org/wiki/WebSocket

WebSocket is a protocol providing full-duplex communication channels over a single TCP connection.

WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. The WebSocket protocol makes more interaction between a browser and a website possible, facilitating the real-time data transfer from and to the server. This is made possible by providing a standardised way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. The communications are done over TCP port

number 80, which is of benefit for those environments which block non-web Internet connections using a firewall. Similar two-way browser-server communications have been achieved in non-standardised ways using stopgap technologies such as Comet.

The WebSocket protocol is currently supported in most major browsers including Google Chrome, Internet Explorer, Firefox, Safari and Opera. WebSocket also requires web applications on the server to support it.

2.3 Data Encoding Protocols

If we put aside "full-stack" standards like OIC/OPC or AllJoyn which come with their own way of encoding messages, it's desirable that AGILE has "raw" support for **data encoding protocols** that developers may use to transport data/control information using the messaging protocols listed in the previous section. There are several encoding protocols specifically targeting IoT that are meant to be particularly efficient at saving bandwidth and/or processing power (and therefore energy) required.

2.3.1 Evaluation Criteria

The following criteria have been adopted for the selection of the data encoding protocols:

- **Optimised for bandwidth**
- **Optimised for processing power**
- **Self-described**

2.3.2 Protocol Selection

Driven by the previous criteria, the selection process allowed the identification of the following data encoding protocols.

CBOR

Link: http://tools.ietf.org/html/rfc7049

The Concise Binary Object Representation (CBOR) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation. These design goals make it different from earlier binary serialisations such as ASN.1 and MessagePack.

XML

Link: https://en.wikipedia.org/wiki/XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specification and by several other related specifications, all of which are free open standards.

The design goals of XML emphasise simplicity, generality and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services.

JSON

Link: https://en.wikipedia.org/wiki/JSON

JSON (JavaScript Object Notation) is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is the most common data format used for asynchronous browser/server communication (AJAJ), largely replacing XML which is used by AJAX.

JSON is a language-independent data format. It derives from JavaScript, but as of 2016 code to generate and parse JSON-format data is available in many programming languages. The official Internet media type for JSON is application/json. The JSON filename extension is .json.

A typical mashup fetches JSON-format data from several different web servers using an Open API.

JSON's basic data types are:

- Number: a signed decimal number that may contain a fractional part and may use exponential E notation, but cannot include non-numbers like NaN. The format makes no distinction between integer and floating-point. JavaScript uses a double-precision floating-point format for all its numeric values, but other languages implementing JSON may encode numbers differently.
- String: a sequence of zero or more Unicode characters. Strings are delimited with double-quotation marks and support a backslash escaping syntax.
- Boolean: either of the values true or false
- Array: an ordered list of zero or more values, each of which may be of any type. Arrays use square bracket notation with elements being comma-separated.
- Object: an unordered collection of name/value pairs where the names (also called keys) are strings. Since objects are intended to represent associative arrays,[11] it is recommended, though not required,[12] that each key is unique within an object.

Objects are delimited with curly brackets and use commas to separate each pair, while within each pair the colon ':' character separates the key or name from its value.

- null: An empty value, using the word null

Whitespace is allowed and ignored around or between syntactic elements (values and punctuation, but not within a string value). Four specific characters are considered whitespace for this purpose: space, horizontal tab, line feed, and carriage return. JSON does not provide any syntax for comments.

OGC SensorML

Link: https://en.wikipedia.org/wiki/SensorML

SensorML is an approved Open Geospatial Consortium standard. SensorML provides standard models and an XML encoding for describing sensors and measurement processes. SensorML can be used to describe a wide range of sensors, including both dynamic and stationary platforms and both in-situ and remote sensors.

Functions supported include

- sensor discovery
- sensor geolocation
- processing of sensor observations
- a sensor programming mechanism
- subscription to sensor alerts

Examples of supported sensors are

- stationary, in-situ – chemical “sniffer”, thermometer, gravity meter
- stationary, remote – stream velocity profiler, atmospheric profiler, Doppler radar
- dynamic, in-situ – aircraft mounted ozone “sniffer”, GPS unit, dropsonde

dynamic, remote – satellite radiometer, airborne camera, soldier-mounted video

Protocol Buffers

Link: https://en.wikipedia.org/wiki/Protocol_Buffers

Protocol Buffers is a method of serialising structured data. It is useful in developing programs to communicate with each other over a wire or for storing data. The method involves an interface description language that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes that represents the structured data.

Google developed Protocol Buffers for use internally and has made protocol compilers for C++, Java and Python available to the public under a free software, open source license. Various other

language implementations are also available, including C#, JavaScript, Go, Perl, PHP, Ruby, Scala and Julia.

The design goals for Protocol Buffers emphasised simplicity and performance. In particular, it was designed to be smaller and faster than XML.

Protocol Buffers is widely used at Google for storing and interchanging all kinds of structured information. The method serves as a basis for a custom remote procedure call (RPC) system that is used for nearly all inter-machine communication at Google.

Protocol Buffers are very similar to the Apache Thrift protocol (used by Facebook for example), except that the public Protocol Buffers implementation does not include a concrete RPC protocol stack to use for defined services.

Though the primary purpose of Protocol Buffers is to facilitate network communication, its simplicity and speed make Protocol Buffers an alternative to data-centric C++ classes and structs, especially where interoperability with other languages or systems might be needed in the future.

2.4 Remote Management Protocols

In this section we evaluate the protocols that enable **remote management** of a communicating device, in our case an AGILE gateway, or the assets it is in control of. The main evaluation criteria are around “standardness” and “openness”, as we want AGILE to support management protocols ensuring seamless management of a gateway over its lifespan.

2.4.1 Evaluation Criteria

Same as per messaging protocols.

2.4.2 Protocol Selection

LWM2M

Link: https://en.wikipedia.org/wiki/OMA_LWM2M

OMA Lightweight M2M is a protocol from the Open Mobile Alliance for M2M or IoT device management. Lightweight M2M enabler defines the application layer communication protocol between a LWM2M Server and a LWM2M Client, which is located in a LWM2M Device. The OMA Lightweight M2M enabler includes device management and service enablement for LWM2M Devices. The target LWM2M Devices for this enabler are mainly resource constrained devices.

Therefore, this enabler makes use of a light and compact protocol as well as an efficient resource data model. It provides a choice for the M2M Service Provider to deploy a M2M system to provide service to the M2M User. It is frequently used with CoAP.

OMA Lightweight M2M is designed to:

- Provide Device Management functionality over sensor or cellular networks
- Transfer service data from the network to devices
- Extend to meet the requirements of most any application

Lightweight M2M 1.0 enabler introduces the following features below for the initial release.

- Simple Object based resource model
- Resource operations of creation/retrieval/update/deletion/configuration of attribute
- Resource observation/notification
- TLV/JSON/Plain Text/Opaque data format support
- UDP and SMS transport layer support
- DTLS based security
- Queue mode for NAT/Firewall environment
- Multiple LWM2M Server support

Basic M2M functionalities: LWM2M Server, Access Control, Device, Connectivity, Firmware Update, Location, Connectivity Statistics

OMA-DM

Link: https://en.wikipedia.org/wiki/OMA_Device_Management

OMA DM specification is designed for management of mobile devices such as mobile phones, PDAs, and tablet computers. Device management is intended to support the following uses:

- Provisioning – Configuration of the device (including first time use), enabling and disabling features
- Device Configuration – Allow changes to settings and parameters of the device
- Software Upgrades – Provide for new software and/or bug fixes to be loaded on the device, including applications and system software
- Fault Management – Report errors from the device, query about status of device

All of the above functions are supported by the OMA DM specification, and a device may optionally implement all or a subset of these features. Since OMA DM specification is aimed at mobile devices, it is designed with sensitivity to the following:

- small foot-print devices, where memory and storage space may be limited
- constraint on bandwidth of communication, such as in wireless connectivity
- tight security, as the devices are vulnerable to software attacks; authentication and challenges are made part of the specifications

OneM2M

Link: http://www.onem2m.org/

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide. A critical objective of oneM2M is to attract and actively involve organisations from M2M-related business domains such as: telematics and intelligent transportation, healthcare, utilities, industrial automation, smart homes, etc. Initially, oneM2M shall prepare, approve and maintain the necessary set of Technical Specifications and Technical Reports for:

- Use cases and requirements for a common set of Service Layer capabilities;
- Service Layer aspects with high level and detailed service architecture, in light of an access independent view of end-to-end services;
- Protocols/APIs/standard objects based on this architecture (open interfaces & protocols);
- Security and privacy aspects (authentication, encryption, integrity verification);
- Reachability and discovery of applications;
- Interoperability, including test and conformance specifications;
- Collection of data for charging records (to be used for billing and statistical purposes);
- Identification and naming of devices and applications;
- Information models and data management (including store and subscribe/notify functionality);
- Management aspects (including remote management of entities); and
- Common use cases, terminal/module aspects, including Service Layer interfaces/APIs between:
    - Application and Service Layers;
    - Service Layer and communication functions

OpenFlow

Link: https://en.wikipedia.org/wiki/OpenFlow

OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network.

OpenFlow enables network controllers to determine the path of network packets across a network of switches. The controllers are distinct from the switches. This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols. Also, OpenFlow allows switches from different vendors — often each with their own proprietary interfaces and scripting languages — to be managed remotely using a single, open protocol. The protocol's inventors consider OpenFlow an enabler of software defined networking (SDN).

OpenFlow allows remote administration of a layer 3 switch's packet forwarding tables, by adding, modifying and removing packet matching rules and actions. This way, routing decisions can be made periodically or ad hoc by the controller and translated into rules and actions with

a configurable lifespan, which are then deployed to a switch's flow table, leaving the actual forwarding of matched packets to the switch at wire speed for the duration of those rules. Packets which are unmatched by the switch can be forwarded to the controller. The controller can then decide to modify existing flow table rules on one or more switches or to deploy new rules, to prevent a structural flow of traffic between switch and controller. It could even decide to forward the traffic itself, provided that it has told the switch to forward entire packets instead of just their header.

The OpenFlow protocol is layered on top of the Transmission Control Protocol (TCP), and prescribes the use of Transport Layer Security (TLS). Controllers should listen on TCP port 6653 for switches that want to set up a connection. Earlier versions of the OpenFlow protocol unofficially used port 6633.

SNMP

Link: https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

Simple Network Management Protocol (SNMP) is an Internet-standard protocol for collecting and organising information about managed devices on IP networks and for modifying that information to change device behaviour. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more.

SNMP is widely used in network management systems to monitor network-attached devices for conditions that warrant administrative attention. SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications.

SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects.

In typical uses of SNMP one or more administrative computers, called managers, have the task of monitoring or managing a group of hosts or devices on a computer network. Each managed system executes, at all times, a software component called an agent which reports information via SNMP to the manager.

An SNMP-managed network consists of three key components:

- Managed device
- Agent — software which runs on managed devices
- Network management station (NMS) — software which runs on the manager

A managed device is a network node that implements an SNMP interface that allows unidirectional (read-only) or bidirectional (read and write) access to node-specific information. Managed devices exchange node-specific information with the NMSs. Sometimes called network elements, the managed devices can be any type of device, including, but not limited to,

routers, access servers, switches, cable modems, bridges, hubs, IP telephones, IP video cameras, computer hosts, and printers.

An agent is a network-management software module that resides on a managed device. An agent has local knowledge of management information and translates that information to or from an SNMP-specific form.

A network management station (NMS) executes applications that monitor and control managed devices. NMSes provide the bulk of the processing and memory resources required for network management. One or more NMSes may exist on any managed network.

2.5 Security Protocols and Standards

In this section, we list the **security protocols** that need to be supported in AGILE. Security is an important aspect of an IoT solution, and we will favour security mechanisms based state-of-the-art cryptography techniques, and that ease the integration with existing systems (e.g. Web-based solutions).

2.5.1 Evaluation Criteria

Driven by previous criteria set, as well as per the Confidentiality, Integrity and Availability triad. This component is further analysed as part of WP5 (Gateway Security, Data Provenance & Access Control).

2.5.2 Protocol and Standard Selection

DNS-SEC

Link: https://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

The Domain Name System Security Extensions (DNSSEC) is a suite of Internet Engineering Task Force (IETF) specifications for securing certain kinds of information provided by the Domain Name System (DNS) as used on Internet Protocol (IP) networks. It is a set of extensions to DNS which provide to DNS clients (resolvers) origin authentication of DNS data, authenticated denial of existence, and data integrity, but not availability or confidentiality.

The original design of the Domain Name System (DNS) did not include security; instead, it was designed to be a scalable distributed system. The Domain Name System Security Extensions (DNSSEC) attempts to add security, while maintaining backward compatibility. RFC 3833 documents some of the known threats to the DNS and how DNSSEC responds to those threats.

DNSSEC was designed to protect applications (and caching resolvers serving those applications) from using forged or manipulated DNS data, such as that created by DNS cache poisoning. All answers from DNSSEC protected zones are digitally signed. By checking the digital signature, a DNS resolver is able to check if the information is identical (i.e. unmodified and complete) to the information published by the zone owner and served on an authoritative DNS server. While protecting IP addresses is the immediate concern for many users, DNSSEC can protect any data published in the DNS, including text records (TXT), mail exchange records (MX), and can be used to bootstrap other security systems that publish references to cryptographic certificates stored in the DNS such as Certificate Records (CERT records, RFC 4398), SSH fingerprints (SSHFP, RFC 4255), IPSec public keys (IPSECKEY, RFC 4025), and TLS Trust Anchors (TLSA, RFC 6698).

DNSSEC does not provide confidentiality of data; in particular, all DNSSEC responses are authenticated but not encrypted. DNSSEC does not protect against DoS attacks directly, though it indirectly provides some benefit (because signature checking allows the use of potentially untrustworthy parties; this is true only if the DNS server is using a self-signed certificate, not recommended for Internet-facing DNS servers).

DTLS

Link: https://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security

In information technology, the Datagram Transport Layer Security (DTLS) communications protocol provides communications security for datagram protocols. DTLS allows datagram-based applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. The DTLS protocol is based on the stream-oriented Transport Layer Security (TLS) protocol and is intended to provide similar security guarantees. The DTLS protocol datagram preserves the semantics of the underlying transport — the application does not suffer from the delays associated with stream protocols, but has to deal with packet reordering, loss of datagram and data larger than the size of a datagram network packet.

oAuth

Link: https://en.wikipedia.org/wiki/OAuth

OAuth is an open standard for authorisation, commonly used as a way for Internet users to log into third party websites using their Microsoft, Google, Facebook, Twitter, One Network etc. accounts without exposing their password. Generally, OAuth provides to clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorise third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorisation server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server.

TLS

Link: https://en.wikipedia.org/wiki/Transport_Layer_Security

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), both of which are frequently referred to as 'SSL', are cryptographic protocols designed to provide communications security over a computer network. Several versions of the protocols are in widespread use in applications such as web browsing, email, Internet faxing, instant messaging, and voice-over-IP (VoIP). Major web sites use TLS to secure all communications between their servers and web browsers.

The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating computer applications. When secured by TLS, connections between a client (e.g. a web browser) and a server (e.g. wikipedia.org) will have one or more of the following properties:

The connection is private because symmetric cryptography is used to encrypt the data transmitted. The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated at the start of the session (see TLS handshake protocol). The server and client negotiate the details of which encryption algorithm and cryptographic keys to use before the first byte of data is transmitted (see Algorithm). The negotiation of a shared secret is both secure (the negotiated secret is unavailable to eavesdroppers and cannot be obtained, even by an attacker who places himself in the middle of the connection) and reliable (no attacker can modify the communications during the negotiation without being detected).

The identity of the communicating parties can be authenticated using public-key cryptography. This authentication can be made optional, but is generally required for at least one of the parties (typically the server).

The connection is reliable because each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission.

In addition to the properties above, careful configuration of TLS can provide additional privacy-related properties such as forward secrecy, ensuring that any future disclosure of encryption keys cannot be used to decrypt any TLS communications recorded in the past.[2]

TLS supports many different methods for exchanging keys, encrypting data, and authenticating message integrity (see Algorithm). As a result, secure configuration of TLS involves many configurable parameters, and not all choices provide all of the privacy-related properties described in the list above (see authentication and key exchange table, cipher security table, and data integrity table).

Attempts have been made to subvert aspects of the communications security that TLS seeks to provide and the protocol has been revised several times to address these security threats (see Security). Web browsers have also been revised by their developers to defend against potential security weaknesses after these were discovered (see TLS/SSL support history of web browsers).

2.6 Full-Stack Standards

Over the last few years, several SDOs have worked on providing integrated solutions that simplify end-to-end integration of IoT solutions, all the way from the low-level transport layers, to networking functions, to defining the semantics of the IoT data streams.

2.6.1 Evaluation Criteria

Driven by all criteria defined for protocols as well as security, in the respective sections.

2.6.2 Standard Selection

OIC/OCF

Link: http://openconnectivity.org

Billions of connected devices (devices, phones, computers and sensors) should be able to communicate with one another regardless of manufacturer, operating system, chipset or physical transport. The Open Connectivity Foundation (OCF) is creating a specification and sponsoring an open source project to make this possible. OCF will unlock the massive opportunity in the IoT market, accelerate industry innovation and help developers and companies create solutions that map to a single open specification. OCF will help ensure secure interoperability for consumers, business, and industry.

Alljoyn

Link: https://en.wikipedia.org/wiki/AllJoyn

AllJoyn is a system that allows devices to communicate with other devices around them. A simple example would be a motion sensor letting a light bulb know no one is in the room it is lighting, so it can shut itself off.

The system itself is an open source project which provides a universal software framework and core set of system services that enable interoperability among connected products and software applications across manufacturers to create dynamic proximal networks using a D-Bus message bus. Qualcomm has led development of this open source project, and first presented it at the Mobile World Congress 2011. Unity Technologies has provided the 'AllJoyn Unity Extension' packaged with the AllJoyn SDK release 2.3.6 and above. Major OEM and ODM partners includes Foxconn, Technicolor, LG-Innotek, LeTV and Xiaomi.

The AllJoyn software framework and core system services let compatible devices and applications find each other, communicate and collaborate across the boundaries of product category, platform, brand, and connection type. Target devices include those in the fields of Connected Home, Smart TV, Smart Audio, Broadband Gateways, and Automotive. Qualcomm is

working on providing OEM solutions. Currently, the communication layer (and thus hardware requirements) is limited to wi-fi.

Though the protocol started at Qualcomm, they have signed over the source code to the Linux Foundation. The AllSeen Alliance has been created to promote some type of interoperability for the internet of things, and a number of consumer brands have signed on including LG, Sharp, Haier, Panasonic, Sony, Electrolux, Sears and Arçelik. Other members include Silicon Image, Cisco, TP-Link, Canary, Two Bulls, doubleTwist, FON, Harman, HTC, LIFX, Liteon, Muzzley, Sproutling, Microsoft and Wilocity.

Zigbee

Link: https://en.wikipedia.org/wiki/ZigBee

ZigBee is an IEEE 802.15.4-based specification for a suite of high-level communication protocols used to create personal area networks with small, low-power digital radios.

The technology defined by the ZigBee specification is intended to be simpler and less expensive than other wireless personal area networks (WPANs), such as Bluetooth or Wi-Fi. Applications include wireless light switches, electrical meters with in-home-displays, traffic management systems, and other consumer and industrial equipment that requires short-range low-rate wireless data transfer.

Its low power consumption limits transmission distances to 10–100 metres line-of-sight, depending on power output and environmental characteristics. ZigBee devices can transmit data over long distances by passing data through a mesh network of intermediate devices to reach more distant ones. ZigBee is typically used in low data rate applications that require long battery life and secure networking (ZigBee networks are secured by 128-bit symmetric encryption keys). ZigBee has a defined rate of 250 kbit/s, best suited for intermittent data transmissions from a sensor or input device.

Z-Wave

Link: https://en.wikipedia.org/wiki/Z-Wave

Z-Wave is a wireless communications protocol for home automation. It is oriented to the residential control and automation market and is intended to provide a simple and reliable method to wirelessly control lighting, HVAC, security systems, home cinema, automated window treatments, swimming pool and spa controls, and garage and home access controls. There are hundreds of interoperable Z-Wave products marketed under different brands, and over 35 million have been sold since 2005. Z-Wave was developed by a Danish start-up called Zen-Sys that was acquired by Sigma Designs in 2008.

A Z-Wave automation system can be remote controlled via the Internet, using a Z-Wave gateway or central control device which serves as both the Z-Wave hub controller and portal to the outside.

With regards to security, Z-Wave is based on a proprietary design and a sole chip vendor. Although there have been a number of academic and practical security researches on home automation systems based on ZigBee and X10 protocols, research is still in its infancy to analyse the Z-Wave protocol stack layers, requiring the design of a radio packet capture device and related software to intercept Z-Wave communications. An early vulnerability was uncovered in AES-encrypted Z-Wave door locks that could be remotely exploited to unlock doors without the knowledge of the encryption keys, and due to the changed keys, subsequent network messages, as in "door is open", would be ignored by the network. This vulnerability was not due to a flaw in the Z-Wave protocol specification but instead was an implementation error.

Finally, on the hardware side, the chip for Z-Wave nodes is the ZW0201, built around an Intel MCS-51 microcontroller with an internal system clock of 16 MHz. The RF part of the chip contains an GisFSK transceiver for a software selectable frequency. With a power supply 2.2-3.6 volts, it consumes 23mA in transmit mode.

3 Report Conclusion

The outcomes of this report feed into further deliverables incl. D2.2 (Initial version of Gateway Self- configuration, IoT Device discovery & Remote gateway management) due on M9, and D2.3 (Final version of Gateway Self- configuration, IoT Device discovery & Remote gateway management) due on M18.

An extensive review of the protocols and standards that were assessed for their fitness in AGILE follows in the Appendix.

Appendix A – Protocols and Standards in IoT

Connectivity Protocols

	Communication			Technology			
Protocol/Standard	Asset ↔ GW	GW ↔ GW	GW ↔ Cloud	Wired / wireless	Range	Max throughput	Security
6LoWPAN	✓	✓	✓	wireless			
802.15.4	✓			wireless	10-20m indoors 300m "free-range"		✓
Bluetooth 4.0	✓			wireless	~10m	up to 24 Mbps	
Bluetooth Smart (BLE)	✓	✓		wireless	>100m	1 Mbps	✓
enOcean	✓			wireless	30m indoors 300m "free-range"	120 kbps	✓
Ethernet	✓	✓	✓	wired			
KNX	✓			two-wire bus PLC RF 868 MHz	~1000m - -	9,600 bps 1,200 bps 16.4 kbps	✓
LoRA	✓			wireless	2-5km urban <15km suburban	0.3 to 38.4 kbps	✓
NB-LTE			✓	wireless	<15km	up to 150 kbps	✓
NFC	✓			wireless	<10cm	~2.5 kbps	✓
RF (433/833MHz)	✓	✓		wireless			
RS-232	✓			wired	<300m	115,200 bps	

RS-485	✓	✓		wired	1,200m	35 Mbps up to 10 m 100 kbps at 1200 m	
Sigfox			✓	wireless	10km urban 50km rural	100bps	
Wi-Fi	✓	✓	✓	wireless	50m indoors 100m "free-range"		
Zigbee				wireless	10-100m "free-range"		✓

Messaging Protocols

	Communication			Technology		Open source aspects	
Protocol/Standard	Asset ↔ GW	GW ↔ GW	GW ↔ Cloud	Discovery	Security	Open standard?	OSS implementations available?
CoAP	✓	✓	✓	✓	✓	✓	+++
Modbus						?	
MQTT			✓		✓	✓	+++++
MQTT-SN	✓			✓	✓	✓	++
WebSockets			✓		✓	✓	++++

Remote Management Protocols

Protocol/Standard	Communication			Open source aspects	
	Asset ↔ GW	GW ↔ GW	GW ↔ Cloud	Open standard?	OSS implementations available?
LWM2M			✓	✓	✓
OMA-DM			✓	✓	✓
OneM2M	✓	✓	✓	✓	✓
OpenFlow		✓	✓	✓	✓
SNMP		✓	✓	✓	✓

Data Encoding Protocols

	Communication			Technology		Open source aspects	
Protocol/Standard	Asset ↔ GW	GW ↔ GW	GW ↔ Cloud	Bandwidth usage	CPU/memory usage	Open standard?	OSS implementations?
CBOR	✓	✓	✓	+++	+++	Y	++
XML		✓	✓	---	-	Y	++++
JSON	✓	✓	✓	+		Y	++++
OGC SensorML		✓	✓	?	?	Y	
Protocol Buffers	✓	✓	✓	++	++	Y	++++

Security Protocols

Protocol/Standard	Communication Asset ↔ GW	GW ↔ GW	GW ↔ Cloud	Open source aspects Open standard?	OSS implementations available?
DNS-SEC			✓	✓	✓
DTLS		✓	✓	✓	✓
oAuth		✓	✓	✓	✓
TLS		✓	✓	✓	✓

Full-Stack Protocols

Protocol/Standard	Communication			Technology		
	Asset ↔ GW	GW ↔ GW	GW ↔ Cloud	Connectivity	Messaging	Encoding
OIC/OCF	✓	✓	✓		CoAP	CBOR, JSON
AllJoyn	✓	✓	?			
Zigbee	✓	✓		802.15.4		
Z-Wave	✓	?		Proprietary	Proprietary	Manchester Code