This document describes the first version of the Wise-IoT platform (architecture, concepts) including the integrated platforms and components developed by the Wise-IoT consortium. It also describes the platforms that will be part of the release 2 presented in the future D3.2.
<table>
<thead>
<tr>
<th>Editor:</th>
<th>Rémi Druilhe, CEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable nature:</td>
<td>O</td>
</tr>
<tr>
<td>Dissemination level:</td>
<td>PU</td>
</tr>
<tr>
<td>Contractual/actual delivery date:</td>
<td>M11 M12</td>
</tr>
</tbody>
</table>

**Disclaimer**

This document contains material, which is the copyright of certain WISE-IOT consortium parties, and may not be reproduced or copied without permission.

All WISE-IOT consortium parties have agreed to full publication of this document.

Neither the WISE-IOT consortium as a whole, nor a certain part of the WISE-IOT consortium, warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, accepting no liability for loss or damage suffered by any person using this information.

This project has received funding from the European Union’s H2020 Programme for research, technological development and demonstration under grant agreement No 723156, the Swiss State Secretariat for Education, Research and Innovation (SERI) and the South-Korean Institute for Information & Communications Technology Promotion (IITP).

**Copyright notice**

© 2016 Participants in project WISE-IOT
## Table of contents

Executive summary .................................................. 4
Glossary ...................................................................... 5
1 Introduction .................................................................. 6
2 The Wise-IoT integrated platform R1 ......................... 8
   2.1 Architecture .......................................................... 8
   2.2 IoT platforms ......................................................... 9
      2.2.1 oneM2M Mobius ............................................... 9
      2.2.2 SmartSantander with Orion Context Broker ........... 10
3 Perspectives for the Wise-IoT integrated platform R2 .. 13
   3.1 IoT platforms ......................................................... 13
      3.1.1 sensiNact ....................................................... 13
      3.1.2 oneM2M Insator ............................................... 14
      3.1.3 GS1 Oliot ....................................................... 15
      3.1.4 NEConfMan ................................................... 16
      3.1.5 Aeron Broker ............................................... 17
   3.2 Recommendation system platforms ......................... 18
      3.2.1 IoT Recommender ........................................... 18
      3.2.2 Adherence Monitor ......................................... 19
      3.2.3 Web-based ontology and linked-data validator ........ 20
      3.2.4 Trust management .......................................... 21
4 Conclusion .................................................................... 24
5 References .................................................................... 25
Executive summary

This deliverable describes the first release of the Wise-IoT platform including the integrated platforms from the partners. This integration is the first step towards the interconnection of various and heterogeneous platforms: the Wise-IoT integrated platform aims at evolving and integrating off-the-shelf platforms in order, for the end user, to consume data seamlessly of its location and of the platform providing it.

This first release provides the very basis for demonstrating the approach. It interconnects the Orion Context Broker platform (deployed in Santander) and the oneM2M Mobius platform (deployed in Busan). The Morphing Mediation Gateway, developed in the WP2, is in charge of the translation between the two semantic models.

The Orion Context Broker has been developed by Telefonica, in Europe, whereas oneM2M Mobius has been developed by Korea Electronics Technology Institute, in Korea. Those two developments are independent leading to an incompatibility between the two platforms and their respective protocols. This deliverable provides the two platforms and the components allowing this interconnection between NGSI and oneM2M.

The second section of this deliverable discusses about the ongoing architecture of the Wise-IoT integration platform. It lists and describes the platforms that will be part of the final release. This section distinguishes between the IoT platforms that are directly connected to the sensors and the actuators, especially for the use cases, and the platforms that are part of the recommendation system.

Finally, all of platforms will be deployed by the different partners on the experiment sites to gather data and allow the developers to create end users applications that will be used in the various use cases of the project. The platforms are the first step towards the implementation of the use cases.

The D3.2 will propose the final version of the Wise-IoT integration platform based on feedbacks from the first release as well as on the feedbacks from the use cases.
# Glossary

<table>
<thead>
<tr>
<th>Term or Abbreviation</th>
<th>Definition (Source)</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interfaces</td>
</tr>
<tr>
<td>CB</td>
<td>Context Broker</td>
</tr>
<tr>
<td>GE</td>
<td>Generic Enabler</td>
</tr>
<tr>
<td>LoRa</td>
<td>Long Range</td>
</tr>
<tr>
<td>MQTT</td>
<td>MQ Telemetry Transport</td>
</tr>
<tr>
<td>MMG</td>
<td>Morphing Mediation Gateway</td>
</tr>
<tr>
<td>NFC</td>
<td>Near Field Communications</td>
</tr>
<tr>
<td>NGSI</td>
<td>Next Generation Service Interface</td>
</tr>
<tr>
<td>OCEAN</td>
<td>Open allianCE for iot stAndard</td>
</tr>
<tr>
<td>OLIOT</td>
<td>Open Language for Internet of Things</td>
</tr>
<tr>
<td>OTAP</td>
<td>Over-The-Air Programming</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio-Frequency Identification</td>
</tr>
</tbody>
</table>
1 Introduction

The emergence of the IoT comes with the development of platforms able to gather data provided by the sensors and to activate physical actuators. Those platforms cover multiple environments (e.g., smart city, smart home) and various protocols (e.g., LoRaWAN, MQTT). But, for now, no platforms cover all the protocols that can be deployed in all the environments. Thus, it is important to develop bridges between the platforms to access to this set of heterogeneous environments and the underlying sensors and actuators.

The objective of the Wise-IoT project is to propose a framework allowing the interconnection of various IoT platforms between Europe and Korea, exchanging data and consuming data seamlessly in the applications in order to provider upper layer services to end users.

For example, a user requires traffic information in the city and on the slopes of a skiing station to reach his destination. Both traffic information will not be provided by the same platform because the sensors deployed do not use the same protocol (one is using LoRa and the other is using HTTP) or are not managed by the same provider (one is provided by the city and the other is provided by the skiing resort). In this case, a semantic platform needs to uniformize both informations and provides it in the same way to the developer to ease the development of the traffic application.

To solve this interconnection issue, the Wise-IoT project proposes the use of off-the-shelf and heterogeneous platforms as the basis for the experiment. On top of that, the Morphing Mediation Gateway, defined in the D2.1, unifies the data provided by each platform and makes it available to the application developers.

This document describes the release 1 (R1) of this integration of platforms with the interconnection of the SmartSantander (with the Orion Context Broker) platform and the oneM2M Mobius platform (cf. Figure 1). This document makes the distinction between the IoT platforms that are deployed to connect with physical sensors and actuators for a specific experiment site, e.g., Santander, Busan, and the “recommendation system” platforms that process the data from the IoT platforms.

Figure 1 - Architecture of the integrated platforms R1
The Table 1 summarizes the IoT platforms as available in the release 1 and in the release 2 of the Wise-IoT project. The release 1 focuses on the interconnection between NGSI and oneM2M platforms, while the release 2 integrates other platforms from other partners and environments.

**Table 1 - List of the IoT platforms expected for the Release 1 and the Release 2**

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Provider</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>oneM2M Mobius</td>
<td>KETI</td>
<td>R1</td>
</tr>
<tr>
<td>SmartSantander with Orion Context Broker</td>
<td>UC</td>
<td>R1</td>
</tr>
<tr>
<td>sensiNact</td>
<td>CEA</td>
<td>R2</td>
</tr>
<tr>
<td>oneM2M Insator</td>
<td>Samsung SDS</td>
<td>R2</td>
</tr>
<tr>
<td>GS1 Oliot</td>
<td>KAIST</td>
<td>R2</td>
</tr>
<tr>
<td>NEConfMan</td>
<td>NEC</td>
<td>R2</td>
</tr>
<tr>
<td>Aeron Broker</td>
<td>NEC</td>
<td>R2</td>
</tr>
</tbody>
</table>

The R2 will include other platforms that will be deployed in the smart cities (cf. Table 2). It will also address another environment. Platforms will be deployed in a skiing resort in Grenoble (EU) and in Alpensia (KR) allowing to demonstrate other use cases.

**Table 2 - Deployment site of the platforms**

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Santander</th>
<th>Busan</th>
<th>Grenoble</th>
<th>Alpensia</th>
</tr>
</thead>
<tbody>
<tr>
<td>oneM2M Mobius</td>
<td>R1</td>
<td>R1</td>
<td></td>
<td>R2</td>
</tr>
<tr>
<td>SmartSantander with Orion Context Broker</td>
<td>R1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sensiNact</td>
<td>R2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>oneM2M Insator</td>
<td>R2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GS1 Oliot</td>
<td>R2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEConfMan</td>
<td>R2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aeron Broker</td>
<td>R2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This project also proposes to integrate platforms providing data processing, e.g., data quality, data trust, recommendations. Those platforms are transversal to the use case sites and are not supposed to be connected directly to the physical world. The Table 3 lists the platforms that are part of the “recommendation system” that will be deployed in the release 2 of the Wise-IoT project.

**Table 3 - List of the recommendation system platforms expected for the Release 2**

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Provider</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoT Recommender</td>
<td>IMT-TSP</td>
<td>R2</td>
</tr>
<tr>
<td>Adherence Monitor</td>
<td>FHNW</td>
<td>R2</td>
</tr>
<tr>
<td>Web-based ontology and linked-data validator</td>
<td>EGM</td>
<td>R2</td>
</tr>
<tr>
<td>Trust Management</td>
<td>LJMU</td>
<td>R2</td>
</tr>
</tbody>
</table>
2 The Wise-IoT integrated platform R1

This section describes the first release of the Wise-IoT platform, i.e., its architecture and the integrated platforms.

2.1 Architecture

The architecture of the Wise-IoT platform will be detailed in the D1.2. However, for the understanding of the document a first draft related to the release 1 is proposed in the Figure 2.

The Wise-IoT integrated platform interconnects off-the-shelf platforms provided by the partner of the project. For the first release of the Wise-IoT integrated platform, two platforms are integrated: (1) Orion Context Broker (deployed in Santander, Spain) that is part of the Context Information Management (CIM) and (2) oneM2M Mobius (deployed in Busan, Korea) that is part of the oneM2M Service layer.

To perform the interconnection, the platforms are interconnected using the Morphing Mediation Gateway. The Morphing Mediation Gateway instantiates semantic components able to exchange data between two semantic models. Thus, the applications can use the data provided by various IoT platforms.
The platforms of the release 1 are the one used in the smart city use cases of the project using distinct semantic models, i.e., respectively NGSI and oneM2M, and deployed in two distincts cities (cf. Figure 3). The main goal of this first release is to demonstrate the integration of two platforms before extending it to others platforms in the release 2.

2.2 IoTplatforms

In the release 1 of the Wise-IoT integrated platform, the platforms deployed in Busan (Korea) and Santander (Spain) are fully integrated. This section describes those associated technologies, respectively oneM2M Mobius and SmartSantander (using Orion Context Broker).

2.2.1 oneM2M Mobius

oneM2M Mobius is one of the platform that is provided from OCEAN open source project[2]. OCEAN has been established in January 2016 with Mobius (server platform) and &Cube (devices platform).

![Figure 4 - Overview of the Mobius platform](image)

The Figure 4 describes which maps to server and devices. The bottom half is the configuration of oneM2M system and components are: IN (Infrastructure Node), MN (Middle Node), ASN (Application Service Node) and ADN (Application Dedicated Node). As described, the Mobius is the open source implementation of the IN which is server-side oneM2M entity. This collects data from devices and provides data to applications (e.g. web, mobile) and also to the other gateways/devices.

Mobius Yellow Turtle (Mobius-YT) is a middleware server platform that connects diverse IoT devices through physical communication medias and creates virtual representations (oneM2M resources) for each IoT device to enable the interactions between each other as well as the communication between devices and IoT applications (cf. Figure 5). In this way, Mobius-YT provides an open environment and APIs for users to interconnect their own devices together and develop user specific IoT services to build an IoT ecosystem.
The IoT server platform can be implemented using various programming languages and the Mobius-YT server platform is developed using Node Js language [3] while the previous released Mobius server platform developed using Java Spring Framework is called Blue Octopus. In addition, Mobius-YT server platform uses Node JS express modules which provides diverse modules for developers including HTTP, XML etc. instead of Node JS express framework.

The current Mobius-YT server platform is compliant to oneM2M standards and supports HTTP, MQTT bindings specified in oneM2M standards. The Mobius-YT server platform implements oneM2M Infrastructure Common Service Entity with structured resource architectures and provides IoT services through HTTP RESTful APIs. The currently released Mobius-YT server platform implements both HTTP and MQTT servers to support HTTP APIs and MQTT related functionalities, respectively, as well as MySQL DBMS for resources storage (cf. Figure 6).

![Figure 5 - Components of the Mobius platform](image)

**2.2.2 SmartSantander with Orion Context Broker**

One of the most attractive aspect of SmartSantander comes with the implementation of a large-scale experimental facility based on a real life IoT infrastructure deployed in the Santander urban landscape. Aligned with the Future Internet Research and Experimentation initiative[4], the facility offers to the
research community the possibility of experimenting over the deployed devices. This way, data generated by the different services implemented by the project is offered to the researchers for developing new services on top of it.

Figure 7 presents the overall architecture of SmartSantander, as well as its integration with the FIWARE platform. It is based on a three-tiered network approach, composed by the following tiers:

- The IoT node tier embraces the majority of the devices deployed in an IoT infrastructure, composed by diverse heterogeneous devices, including miscellaneous sensor platforms, tailor-made devices for specific services as well as RFID and NFC tags. These devices are typically resource-constrained and host a range of sensors and, in some cases, actuators.

- The GW tier links the IoT devices on the edges of the capillary network to the core network infrastructure. IoT nodes are grouped in clusters that depend on a GW device. This node locally gathers and processes the information retrieved by IoT devices within its cluster. It also manages them (transmission/reception of commands), thus scaling and easing the management of the whole network. The GW tier devices are typically more powerful than IoT nodes in terms of memory and processing capabilities, also providing faster and more robust communication interfaces. Also, GW devices allow virtualisation of IoT devices, which enables the instantiation of emulated sensors or actuators that behave in all respects similar to the actual devices.

- The server tier, which receive data from all GW tier nodes, provides more powerful computing platforms with high availability and directly connection to the core network. These servers are used to host IoT data repositories and application servers. As a final step, the concept of federation is supported by the architecture. Servers managing networks located in different physical locations can connect among themselves to allow users of the platforms to transparently access IoT nodes that are deployed in different testbeds.

Others devices such as mobile phones and purpose-built devices with reasonable computing power (e.g. mobile devices in vehicles) as well as providing wide area communication capabilities, can behave as IoT nodes in terms of sensing capabilities and as GW nodes regarding processing and communication capabilities.

As a result of the integration carried out with FIWARE platform, data generated by SmartSantander IoT devices is directly sent to Orion Context Broker (CB) component. Thus, providing external users (either hardware and/or software experimenters or service developers) standardized access to manage the information generated from the SmartSantander facility, which includes both the sensor deployments and the applications used by the citizens.

Every time a new device is registered within the SmartSantander IoT facility (or an existing one updates its capabilities), the Resource Configurator performs a registration/update process within FIWARE, sending a registration message to the Orion Context Broker including such capabilities. Once the registration is completed, new sensor data collected from the SmartSantander facility is translated into new observations and sent to FIWARE.

Within the WISE-IoT project, the data provided by the SmartSantander facility will be offered to the Wise-IoT system through the correspondent FIWARE Context Broker. As it can be seen in Figure 7, the information gathered by the sensors is sent to the SmartSantander core platform, in which the Service Storer component forwards the information to the FIWARE GE. Being compliant with NGSI structure, the information will be stored, following the Wise-IoT data model defined in D2.4, and can be retrieved by other users or applications by using the correspondent CB APIs.
Figure 7 - SmartSantander high level architecture
3 Perspectives for the Wise-IoT integrated platform R2

For the release 2, the Wise-IoT integrated platform will integrate others IoT platforms enabling to connect to the physical devices and gathering the data from the various targeted environments, e.g., smart city, smart skiing. It also integrate platforms enabling to perform recommendation to a final user.

3.1 IoT platforms

3.1.1 sensiNact

sensiNact is a horizontal platform dedicated to IoT and able to be deployed in various environments. Heterogeneity of data is today’s reality in every environment. Emerging IoT devices, legacy systems, increasing number of social networks, mobile applications, open data repositories and web data are the potential exploitable data sources. Thus, sensiNact provides connectivity support to those data sources including today’s IoT protocols and platforms such as LoRa, Zigbee, IEEE 802.15.4, Sigfox, EnOcean, MQTT, XMPP, NGSI, HTTP, CoAP, etc. With its modular approach, connectivity support for new protocols can be rapidly developed and dynamically added to the platform, even at run-time (cf. Figure 8).

sensiNact platform provides a unified view over those heterogeneous sources of data and actions. With the lack of a “de facto” standard data model today in the IoT domain, sensiNact adopts a generic and extensible data model to facilitate building adapters for various protocols. Its core model is based on 4 types of resources: sensor data, action, state variables, and properties. Those resources are accessible by generic and easy to use API providing synchronous (on demand) and asynchronous (periodic or event based) access to data/actions of IoT devices, as well as access to historic data.
sensiNact follows a distributed approach for data and event processing. From the device layer to the cloud layer, passing through the gateway level, data are processed as close as possible to the data source. sensiNact provides data processing functions, varying from simple filters to more complex aggregation functions on streaming data. sensiNact also provides extension points allowing addition of new functions to the platform.

On top of the platform, sensiNact proposes a development tool, the sensiNact Studio, that allows managing devices/services connected to the platform and rapidly creating applications and deploying them to the platform. It is a service composition tool which assists developers in building applications by binding the services via events and actions. With a Domain Specific Language, the developers can express the application logic in terms of ECA (Event-Condition-Action) rules, which is verified and validated by the tool before its deployment into the sensiNact platform. The application developers can then remotely monitor and manage applications (install, start, stop, uninstall, etc.). The tool also provides means to easily build support for new types of protocols and platforms and add it to the platform on-the-fly.

On-going work for R2: On-going work to prepare that component for an integration in R2 includes:

- Develop the sensiNact-oneM2M binding using the MQTT and HTTP protocol.

### 3.1.2 oneM2M Insator

Insator is an enterprise-wide common platform which collect massive volumes of data for the development of various smart solutions and intelligence services through data collection and analytics.

The platform can connect various IoT devices and existing legacy systems covering on premises and cloud (cf. Figure 9). Insator can also support the development of smart applications on top of a connected data pipeline using standard APIs and a data message bus. In order to allow users to develop applications, it supports both web-based and local development environments.
InsatorIoT enables IoT services that seamlessly connect with IoT hardware devices using MQTT, CoAP, and Websocket. It also supports user-defined adapters for IoT standards and communication protocols like Zigbee and Modbus, and allows the user to monitor and control data on the cloud. With an SDK that supports C, Java, .Net, Android and iOS, InsatorIoT is compatible with a variety of languages and protocols. InsatorIoT is designed for covering millions of messages from IoT devices.

Insator™ provides an environment in which the Central Ingestion Framework collects data from various distributed data sources and combines them, i.e., a single channel integrates, collects, and manages multiple data. Insator™ APIM provides a management and operation system to deliver API-based scalable cloud services. It supports features allow. Insator™ KMS creates and manages encryption key for each service based on the tenant’s security policy in a multi-tenant big data environment. KMS can protect encryption key, and confident / available of master key also. It can scale out KMS of active-active mode ensures availability of KMS system. Multi-tenant support also. And finally, Insator™ IAM manages an integrated customer master data from cloud service and offers seamless access to services.

**On-going work for R2:** On-going work to prepare that component for an integration in R2 includes:

- Develop the Insator-oneM2M binding using the HTTP protocol;
- Get oneM2M certification to support the interoperability by Jul. 2017.

### 3.1.3 GS1 Oliot

Oliot (Open Language for Internet of Things) is an open source project which aims to implement and extend GS1 standards for the Internet of Things[5]. Using these standards, GS1 provides a common language for organizations and businesses to identify, capture, and share important information about their products. Figure 10 shows the architecture of Oliot, which is a collection of hardware, software, and data standards based on EPCglobal framework.
In the above architecture, products are tagged with unique IDs. The RFID Reader is responsible for reading EPCs of a tag within a range through a Tag Air Interface and reporting them to Application Level Event (ALE) via the Reader Interface (LLRP). The raw data coming from the readers are then processed by ALE to reduce the volume of EPC data and transform raw tag reads into stream of events which is more suitable for Capturing Application to create event from the data. After receiving events from ALE, Capturing Application generates high-level EPC-related business events and delivers them to EPCIS via EPCIS Capturing Interface. EPCIS persists the data as historical data to be accessed by Accessing Application through EPCIS Accessing Interface; in addition to the events it also stores master data. ALE can be used to handle data coming from devices different from RFID Reader (e.g. oneM2M devices) with other necessary implementations like semantic ontology (from oneM2M); EPCIS can also easily be used to store processed data from devices.

By providing an initial point of contact, Root ONS delegates Local ONSs to provide means of looking up a reference to an EPCIS services using non-serialized identifiers. The Discovery Service also provides a complimentary resolution mechanism, which is authorized most of the time, using serialized identifiers like SGTIN. For example, Accessing Application can request Discovery Service about which EPCIS has information related to a specific product or service; as a result, Discovery Service gives a response of all the EPCIS in which the object has passed through.

**On-going work for R2:** On-going work to prepare that component for an integration in R2 includes:

- Develop the GS1 Oliot-oneM2M binding using the MQTT or HTTP protocol.

### 3.1.4 NEConfMan

The IoT-Discovery GE is the part of the Backend tier of the IoT Architecture which is responsible for context source availability management. The underlying data model of this GE is based on the OMA
NGSI-9 Context Management Information Model. This model relies on the concept of context entities, which are generic entities whose state is described by the means of values of attributes and associated metadata. In the context of IoT, context entities and context entity attributes can be used to model IoT resources and the variables they measure, respectively. Additionally - and more importantly - arbitrary physical objects (Things) like rooms, people, cars, etc. and their attributes like temperature, geo-location, etc. can be used as well.

The IoT-Discovery GE is responsible for the context availability registrations from IoT Agents, thus making it the access point for information about entities and their attributes. In particular, the context availability information provided by the IoT-Discovery GE is either forwarded from IoT Agents exposing the FIWARE NGSI-9/10 interfaces. The role of IoT Agents can be played by either the Data Handling GE in IoT Gateways, or the Backend Device Management GE. It is also possible that the context information is provided by other IoT Backend systems.

Information about context source availability is typically forwarded to the FIWARE Context Broker GE so that context information about Things becomes accessible to applications. However, it can be the case that the Context Broker GE may manage context availability information that is not necessarily provided by the IoT-Discovery GE, therefore linked to the Internet of Things, but gathered from other parts of the application.

More precisely, using the FIWARE NGSI-9 interface that the IoT-Discovery GE provides, applications and services will be able to register, discover and subscribe to updates on context availability information that can be:

- Information about IoT resources and the variables they measure
- Information about Things and their attributes
- Information about Associations between entities, whereby the attributes of a Thing can be derived from attributes of other Things, or from attributes measured by IoT resources.

The IoT-Discovery GE is also specified to optionally different flavors of discovery mechanisms, along with the main mode of discovery, e.g., the geographic discovery where entities can be discovered based on their location.

### 3.1.5 Aeron Broker

The IoT Broker GE is an IoT Backend enabler. It is foreseen to run on a machine in a datacenter, where it serves as a middleware which enables fast and easy access to Internet-of-Things data. Instead of having to deal with the technical details of existing FIWARE IoT installations, application developers only need to set up their application to communicate with the IoT Broker in order to retrieve the data they need. The IoT Broker takes care of the task of communicating with the different IoT devices and gateways to retrieve the needed information on behalf of the applications.

The main interface exposed by the IoT Broker GE is FIWARE NGSI. This API has been developed by the FIWARE community as a binding of the OMA NGSI Context Management standard (OMA NGSI 9/10). This interface is exposed both southbound towards IoT gateways and devices and northbound towards IoT applications. In other words, the IoT Broker GE retrieves information from IoT gateways and devices via the FIWARE NGSI protocol, while the same protocol can be used by applications to retrieve information from the IoT Broker GE. The IoT Broker aggregates, filters, translates, and enriches the FIWARE NGSI based information before passing it to the applications.
FIWARE NGSI comprises of an information model and two distinct interfaces for information exchange. The information model is centered on the concept of so-called context entities. Context entities represent arbitrary objects of the real world, and the state of such objects is described in terms of the values of attributes. In addition, metadata can be used to describe properties of attribute values. In the context of the Internet-of-Things, context entities are used for representing devices like sensors and the values they measure. Additionally, and even more importantly, arbitrary physical objects (Things) like rooms, vehicles, or persons and their attributes like temperature or location, is as well expressed by FIWARE NGSI context entities.

### 3.2 Recommendation system platforms

#### 3.2.1 IoT Recommender

IoT recommender is mainly involved in two Wise-IoT use cases: smart parking and smart skiing. Its development is still ongoing and its architecture is still subject to change. Thus, this document will not provide a detailed description of the platform. The full description will available in the D3.2.

In smart parking use case, the main goal of IoT recommender is to provide the recommendations of a) parking areas, b) the best route from user’s current location to the selected parking area based on the geographical information, context data, and user preferences, and c) the best route from user’s current location to his/her parked car. The geographical information includes the data of road traffic and parking sensors deployed in Santander, which is obtained from the Context Information Management layer, i.e., the NGSI semantic model.
In smart skiing use case, the main goal of IoT recommender is to provide recommendations of a) the best slopes based on the user’s preferences and slope quality, b) the best route from user’s current location to the selected slope, and c) challenges inspiration (game), i.e., what is the best challenge for a user based on his/her preferences. This information is obtained using the oneM2M service layer, i.e., the oneM2M semantic model.

Additionally, IoT recommender integrates other components of the self-adaptive recommender system, i.e., Adherence Monitor platform, Adherence Monitoring platform, Quality of Information (QoI) monitoring platform and Trust management platform. IoT recommender will be implemented in an independent manner with very minimal or no dependency on others components, and each component serves as a plugin which can be added or removed at any time. This modular approach eases the adaption of the IoT recommender to others use cases.

On-going work for R2: On-going work to prepare that component for an integration in R2 includes:

- Develop the use case applications, i.e., smart parking and smart skiing, to use the IoT recommender in Wise-IoT;
- Creating web services and API for accessing and using IoT recommender in Wise-IoT.

### 3.2.2 Adherence Monitor

Trust in a system is the firm belief in the reliability and ability of the system. The aim of the adherence monitor is to offer insights into the users’ trust in the system by monitoring the system’s use and eliciting information about the system’s usefulness according to these users. Such insights are challenging to obtain, especially for large-scale personalized systems. For these systems, requirements analysis cannot be easily scaled to a large number of users with individual preferences. Similarly, system maintenance cannot be easily scaled as these systems exhibit a large number of devices that may break or fail. The insights generated by the adherence monitor supports the scaling by offering cues for focusing requirements analysis and system maintenance.

In the context of a recommendation, we define usefulness as the ability of the system to effectively guide the user in achieving the goals of the user. A user not adhering to a recommendation is a cue that the system was unable to guide the user effectively. The recommendation may have been based on wrong assumptions about the user’s needs and preferences or a wrong assessment of the state of the real world, e.g. due to erroneous context data.

In the current R1 implementation, route-based recommendations are monitored. The adherence monitor can follow a user that seeks a parking space in a city network or a point of interest in a ski resort. This scope has been chosen for ease of trials with the aim of validating the design and use of the adherence monitor. The adherence monitor obtains a route and the user’s location from the use case application. Once a deviation is detected, the adherence monitor publishes the deviations to subscribers through a context broker. Also, the adherence monitor issues a request for user feedback about the reasons for non-adherence and publishes these reasons again through the context broker. Upon successful goal completion, the adherence monitor issues a request for feedback about the user’s quality of experience. The requests are issued and the feedback collected through the use case application with the help of the Supersede framework. Figure 12 gives an overview.
On-going work for R2: On-going work to prepare that component for an integration in R2 includes:

- Expose the Adherence Monitor services through a RESTful API;
- Run the Adherence Monitor in a Docker container;
- Implement the user feedback using the SuperSede framework;
- Generate the user ID using the Adherence Monitor to monitor sessions;
- Implement a subscription mechanism with the FiWare Context Broker;
- Add point-of-interest recommendations and challenge recommendations;
- Orchestrate the use of the adherence monitor on behalf of the use case application;
- Allow the logging tools to subscribe to the context broker.

3.2.3 Web-based ontology and linked-data validator

The Web based ontology and linked-data validator Tool is a web service integrated within a web-based client-server architecture. A simple web client is proposed offering an easy and intuitive user interface to access to the service. Through the GUI, user is able to upload his ontology or semantically annotated data stream to be validated against one or several reference ontologies such as the W3C SSN ontology, oneM2M base ontology. The validator detects syntactic and semantic issues if any, and produces a detailed test report at the end of the process which will help the user to correct the issues.

This tool combines three main functionalities that we can find in separate tools nowadays: lexical validation of XML/JSON format, syntactic validation of an ontology or RDF data regarding to standard specifications (rdfs, owl, etc) and against a given reference ontology, and semantic validation. But all these current tools suffer from several week points such as the need of desktop software installation and configurations (Fluent Editor [7], Protégé [8]), and even for the web based solutions requires to setting an appropriate environment (Moki [9]: wiki system).

This tool offers a a user-friendly, intuitive and fully based web interface that remainsthe user throughout the validation process, we provide also the tool as RESTful API for M2M validation, which makes the service a relevant input for more advanced technologies (ontology alignment). It provides also a validation example with explanation on the validation report in order to help user to understand the validation result. Following the report, user can make corrections to the reported error and make his ontology or RDF linked data valid. Thus, it is a good support to achieve interoperability.

The architecture of the Validation Tool is based on two major parts: the front end and the back end (cf. Figure 13). The front end takes most of the popular RDF serialization formats as input in order to
produce a set of evaluation results. In response to user interaction, the back end parse the initial file, using either a XML or JSON parser. If there is no errors, the RDF parser is getting the result as input. If it respect the specification of the RDF model, triples in this model are extracted to serve as the input for the next validation step which is the “validation” module.

Finally the validation module takes the reference ontology which is constructed from the different checked ontologies. The checked ontologies are merged into a single ontology and the triples are also passed as input of the validation module. According to the predefined reference ontology, it checks the syntactic errors in the document which is based on the functionalities implemented in Eyeball, a jena ontology validator[6]. A reasoner is used to enable the logical level verification of the RDF document such as the respect of subsumption relationships between classes, restrictions on class properties and cardinality. The validation results are sent to a reporting server showing a list of errors (Syntax, Lexical and logical) in JSON format, and explanations regarding the ontology affected elements. Once the report is ready in the database, the frontend will get a notification in order to get the available ontology validation report related to the user.

On-going work for R2: The component was initially developed as a syntactic and semantic ontology validator within the H2020 FIESTA project. On-going work to prepare that component for an integration in R2 includes:

- Checking tool compliance with ontologies being used within Wise-IoT;
- Evolving the tool to make it useable as a service, sensors data flow validation, within the recommendation system;
- Investigating the extension of the tool to include new evaluation dimensions so to provide a quality of information evaluation.

### 3.2.4 Trust management

In an IoT platform, such as Wise-IoT, everyday objects are connected to a network in order to share data, generating value from aggregating and analysing large quantities of data. Heterogeneous large-
scale IoT requires close interaction between humans and systems (things), and the markedly increased access to a large amount of personal data/information must be prepared to confront security threats and to protect privacy in various application domains. Furthermore, trust is an important issue if data is to be processed and handled in compliance with user needs and rights in autonomous services without human intervention. To satisfy requirements of trust, significant challenges in data confidentiality and trust among users and things, and its enforcement must be addressed.

As a functionality of Wise-IoT platform, trust management concerns part or all of trust properties in different contexts for achieving certain goals. Trust management is required since participating objects, without any previous interactions, may desire to establish a communication with an acceptable level of trust relationships among themselves. Examples would be in building initial trust bootstrapping, coalition operations without predefined trust, and authentication of certificates generated by another party when links are down or ensuring safety before entering a new zone. In addition, trust management has diverse applicability in many decision making situations including intrusion detection, authentication, access control, key management, isolating misbehaving nodes for effective routing, and other purposes.

Existing trust management system architecture has been design based on either centralized or decentralized approach, depending on trust model and trust-related information processing. In the centralized approach, the trust information can be computed on demand, whenever an entity needs to rely on its cooperative entities, and delivered to the requesting entity at that moment. On the other hand, the distributed approach computes trust on a regular basis and it is propagated throughout the topology. Moreover objects in the large scale network like social IoT possibly lack the knowledge to evaluate trust. It certainly needs help from others such as trusted authorities. Moreover, a real-time trust data flow would result in communication overhead, detrimental to network performance as well as to constrained entities battery life. However, the traditional strategies for centralized system are difficult for solving trust issues of a large scale distributed network like social IoT because of their poor scalability as well as centre dependence leading to single point of failure.

![Figure 14 - Trust Analysis and Management Platform](image.png)

The proposed trust service platform for Wise-IoT consists of components such as Trust Agent, Trust Broker and Trust Analysis and Management, which have been defined with their responsibilities and
interactions as depicted in Figure 14. The most crucial components of the Trust platform for Wise-IoT include:

- **Trust Agent:** used to collect trust-related data from physical, cyber and social IoT domains. The data could be TAs or opinions of entities as recommendation or feedbacks to other entities, applications or services.
- **Trust Broker:** used to provide the trust knowledge to various types of applications and services in social IoT. It is required to register information such as knowledge TM ontology or service requirements prior to the use of the trust service platform.
- **Trust Analysis and Management:** all trust-related mechanisms such as ontology-related manager, information model, reasoning mechanisms, trust cloud infrastructure, Knowledge TM evaluation mechanisms, and trust calculation algorithms are implemented in this module.

**On-going work for R2:** On-going work to prepare that component for an integration in R2 includes:

- Keep track of information obtained from objects (as a feedback) as well as from software components like Adherence Monitor and Context Broker;
- Evaluate the trustworthiness of the entity (trustee);
- Create the machine learning predictive model to generate the trust values.
4 Conclusion

This deliverable describes the first release of the Wise-IoT and the perspectives for the second version of the Wise-IoT platform. The first version of the Wise-IoT platform provides a first deployment to test the Morphing Mediation Gateway between two platforms, enabling the interoperability between two heterogeneous platforms.

This first release demonstrates the interest of the approach in a world in which the set of available platforms is increasing, addressing more and heterogeneous environments. Thus, to demonstrate the scalability of the approach, the second version will use more platforms, e.g., OIC, LoRa, GS1, sensiNact, deployed in other environments.

Because of this larger set of platforms, the architecture must be reviewed and must be more independent and generic to address multiple platforms and environments. Thus, the second release will also propose a new architecture, derived from the one of the first release. This new architecture will be first detailed in the D1.2 and D1.3.

The next deliverable, the D3.2, will describe the final integration of the platforms mentioned in this document. The final deliverable will consider the other targeted environments such as the smart skiing environment. It will also include the platforms dedicated to the processing of the data provided by the environments, in order to achieve the implementation of the use cases described in the D1.1.
5 References