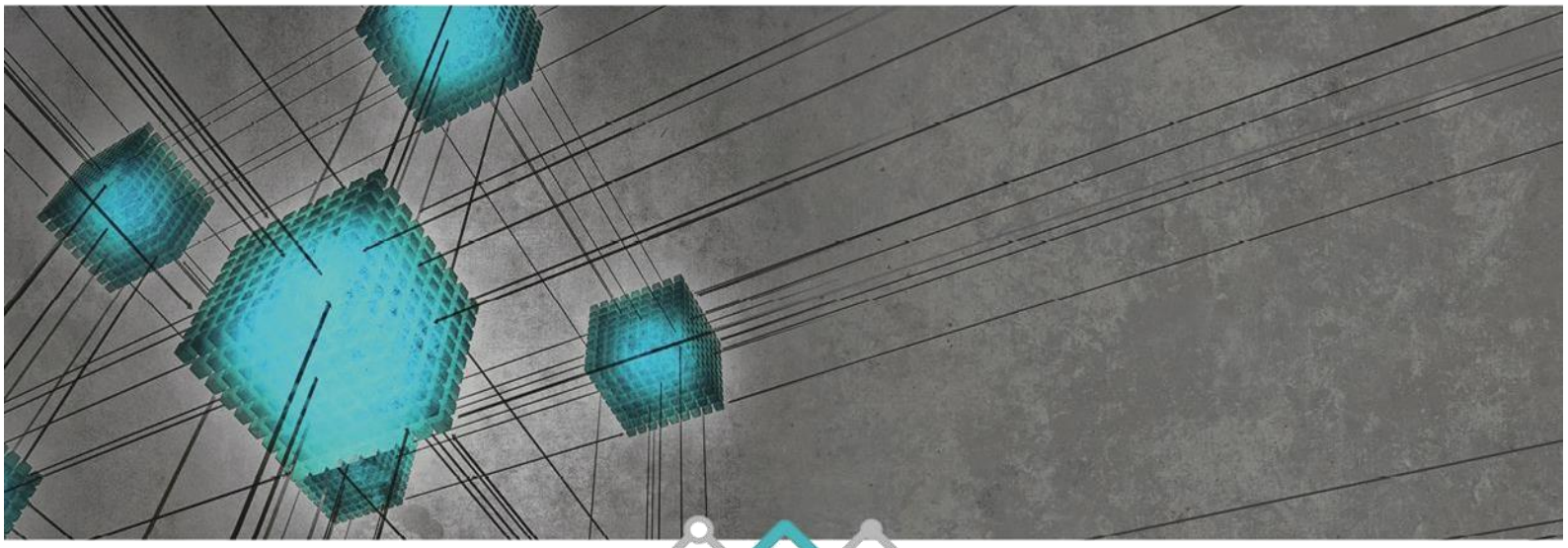

interiot

INTEROPERABILITY
OF HETEREOGENEUS
IOT PLATFORMS.

Inventory of Lessons Learnt

For future users and IoT platform integrators

September 2018

INTER-IoT

INTER-IoT aim is to design, implement and test a framework that will allow interoperability among different Internet of Things (IoT) platforms.

Most current existing IoT developments are based on “closed-loop” concepts, focusing on a specific purpose and being isolated from the rest of the world. Integration between heterogeneous elements is usually done at device or network level, and is just limited to data gathering. Our belief is that a multi-layered approach integrating different IoT devices, networks, platforms, services and applications will allow a global continuum of data, infrastructures and services that can will enable different IoT scenarios. As well, reuse and integration of existing and future IoT systems will be facilitated, creating a defacto global ecosystem of interoperable IoT platforms.

In the absence of global IoT standards, the INTER-IoT results will allow any company to design and develop new IoT devices or services, leveraging on the existing ecosystem, and bring get them to market quickly.

INTER-IoT has been financed by the Horizon 2020 initiative of the European Commission, contract 687283.

INTER-IoT

Inventory of Lessons Learnt

Version: 2.0

Security: Public

September 2018

The INTER-IoT project has been financed by the Horizon 2020 initiative of the European Commission, contract 687283

Disclaimer

Executive Summary

This document provides a summary of the Lessons Learned for future INTER-IoT users and IoT platform integrators. It is focused on the approach followed in order to integrate different legacy and non-legacy IoT platforms with INTER-MW. To elaborate this document, all information obtained during the execution of the project has been taken into account, including pilots and collaboration of third party opencall members. A second version of this document was created after the integration with IoT-01-2016 Large Scale Pilots, adding new lessons and updating some of them given the feedback obtained.

List of Authors

Organisation	Authors
UPV	Carlos E. Palau, Eneko Olivares, Andreu Belsa, Jara Suárez de Puga
UNICAL	Giancarlo Fortino, Raffaele Gravina, Pasquale Pace, Gianluca Aloi, Wilma Russo
PRO	Miguel Montesinos, Miguel A. Llorente, Amelia del Rey
TU/e	George Exarchakos
VPF	Miguel Llop, Pablo Giménez Salazar
XLAB	Flavio Fuart, Damjan Murn
SRIPAS	Maria Ganzha, Paweł Szmeja, Wiesław Pawłowski, Katarzyna Wasielewska-Michniewska
RINICOM	Eric Carlson, Garik Markarian
NPV	Ignacio Huet, Antonio Broseta
ASLTO5	Margherita Gulino, Anna Costa, Marina Mortara, Ilaria De Luca
AFT	Moncef Seminchi, Kimvy Bui
NEWAYS	Roel Vossen, Johan Schabbink, Frans Gevers , Dennis Engbers
ABC	Alessandro Bassi
SABIEN	Gema Ibáñez, Alvaro Fides

Change control datasheet

Version	Changes	Chapters	Pages
0.1	ToC build up responsible assignation	All	3
0.2	Objectives	2	3
0.3	Tables	2	17
1.0	Final version, quality check	All	30
1.1	Update document with feedback from LSP	All	34
2.0	Final version, quality check	All	34

Contents

List of Figures

Acronyms

AIOTI	Alliance for Internet of Things Innovation
BIP	Best Ideas and Projects
EC	European Commission
IERC	European Research Cluster on the Internet of Things
INTER-LAYER	INTER-IoT Layer integration tools
INTER-FW	INTER-IoT Interoperable IoT Framework
INTER-METH	INTER-IoT Engineering Methodology
INTER-LogP	INTER-IoT Platform for Transport and Logistics
INTER-Health	INTER-IoT Platform for Health monitoring
INTER-META-ARCH	INTER-IoT Architectural meta-model for IoT interoperable platforms
INTER-META-DATA	INTER-IoT Metadata-model for IoT interoperable semantics
INTER-API	INTER-IoT Programming library
INTER-CASE	INTER-IoT Computer Aided Software Engineering tool for integration
INCOSE	The International Council on Systems Engineering
IoT	Internet of Things
ITU	International Communications Union
SDO	Standard Development Organisation
SDR	Software Defined Radio
IOT-A	Internet of Things - Architecture
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
SPEM	Software and Systems Process Engineering Meta-model
M2M	Machine to Machine
RFID	Radio Frequency IDentification
MAC	Media Access Control address
SDN	Software Defined Networking
W3C	World Wide Web Consortium
SSN	Semantic Sensor Network
SAREF	Smart Appliances REFerence
OGC	Open Geospatial Consortium
LTE	Long-Term Evolution networks
DSL	Digital Subscriber Lines
CAN	Controller Area Network
API	Application Programming Interface

CRUD	Create, Read, Update and Delete
SDO	Standards Developing Organization
GOIoTP	Generic Ontology for IoT Platforms

1 Introduction

1.1 Overview

The overall goal of the INTER-IoT project is to provide an interoperable and open IoT framework, with associated engineering tools and methodology, for seamless integration of heterogeneous IoT platforms, regardless of the application domains. Regarding the software component from INTER-IoT involved in this document, INTER-MW, it was the most challenging piece of software developed under the project, since it provides core functionalities related to facilitate interoperability among IoT Middleware platforms, as well as the provision of a common abstraction layer to provide access to platform features and information. The bridges in INTER-MW act as a middleman between INTER-MW and IoT platforms. For each IoT platform to be added to INTER-MW a new bridge has to be developed.

In order to elaborate this document, it has been analyzed the information obtained during the development INTER-MW and everything that has been learned regarding the bridges development and integration in the pilots. These pilots are related with health and port transportation and logistics. In addition, the feedback obtained during the collaboration with the opencallers and with the Large Scale Pilots, in particular, Activage Project.

1.2 Structure of the document

There have been a lot of lessons learned during this project related to integrate different legacy and non legacy platforms with INTER-MW. Therefore, in order to provide the information in a more structured way, it has been decided to organize all the knowledge in different areas of interest to the future users. Figure 1. shows these areas of interest.

Figure 1: Lessons Learnt areas of interest

The main topics about each area of interest have been identified and listed. Figure 2 provides a representative sample of the main approaches or problems that may interest a new user or future integrator.

To explain all these lessons learned in detail, Section 2 of this document contains a table for each of the most challenging difficulties faced, filled by all the partners involved in resolving the issue. First, each table describes the problem or difficulty encountered. Afterwards, an explanation of the approach that was followed in order to overcome the problem is presented, describing the process we followed in order to obtain the solution. Lastly, a real use case that the project faced when the described problem emerged is described, with a brief explanation of the actors involved in the use case, and which bridge/feature was being developed when we encountered the issue.

Platforms Ontologies

- IoT Platform provides its own ontology or uses a standard one.
- IoT Platform has the possibility of defining an ontology or custom data model.
- IoT Platform has no ontologies.
- Use of semantically annotated formats is not widespread in industry.
- IoT Platform has some critical data model not present in INTER-IoT.

Platforms functionalities

- IoT Platform does not implement mechanisms to support some functionalities.
- There are IoT Platforms that state being 'open' but in practise they are inaccessible or not maintained.

Platforms deployment

- IoT Platform cannot be deployed locally only cloud access.
- IoT Platform cannot be deployed in cloud (container) only local.

Platforms versioning

- IoT Platform has not a unique implementation.
- Different versions of a platform
- There are IoT Platforms that state being 'open' but in practise they are inaccessible or not maintained.

Platforms Security

- IoT Platform provides a non-standard REST API or Web Services to connect with its functionalities.
- IoT Platform does not provide a REST API or Web Services to connect with its functionalities.
- IoT Platform has the possibility of defining your own interface to access data.
- Publication of data from a legacy data source.

Platforms API

- IoT Platform does not implement mechanisms to support some functionalities.
- There are IoT Platforms that state being 'open' but in practise they are inaccessible or not maintained.

Platforms Privacy

- IoT Platforms are thought as internal systems in industry, so they do not provide a reliable and privacy aware way to share data. DMZ or VPN is needed.
- Not all the platforms are designed to share data. Thus, privacy and industrial protection issues raises constantly.
- Some industrial companies are reluctant to share data.

Devices setup

- Mechanisms to configure IoT devices are not homogeneous (configuration files, commands sent to a broker...)

Custom solutions

- The variety of IoT platforms is huge and there are many industries with custom solutions. Custom solutions usually are designed to solve specific problems so that they do not follow standards or conventions.

Figure 2: Main apporches of each lesson learnt

2 Lessons Learnt

Approach to heterogeneous IoT platforms APIs
The aim of INTER-MW is the implementation of IoT platforms interoperability through platforms' standard APIs that expose their functionalities. In a real-life integration scenario one may encounter different levels of complexity with regards to platform APIs. In addition, some IoT platforms may not follow established standards, or even do not provide any API. We can broadly define the following situation regarding IoT Platforms APIs: 1. IoT platform does not provide an API 2. IoT platform provides a programmatic API (like Java or C++ interface) 3. IoT Platform provides a well-documented and standardised Web Service (like HTTP, WSDL or similar) 4. IoT Platform provides a custom (non standard) Web Service over HTTP(S) 5. IoT platform provides data access through a data broker (such as RabbitMQ, ApacheMQ, …) using standard protocols (such as AMQP(S), MQTT, …). 6. IoT Platform has the possibility of defining your own interface to access data 7. Publication of data from a legacy data source (as in the LogP pilot, see privacy section for approach). The aim of INTER-IoT is to facilitate to a maximum extent possible the integration of any of the APIs mentioned above.
Approach: The approach to IoT platform interoperability taken by INTER-IoT lends itself to a wide range of transport protocols, communication standards, and data formats. This open approach allows for implementation of virtually any transport protocol to connect to INTER-IoT. For example, HTTP(S) and MQTT have already been implemented. As what regards messages data format, our approach facilitates development of a set of "syntactic translators" that take care of two-way conversion between a platform-specific syntax and INTER-MW JSON-LD messages. Once under this common data format, the semantics of messages are translated on a different layer, using IPSM alignments (this is described in detail in the relevant section of this paper). The main challenge of IoT Platform integration is the development of platform-specific bridges and corresponding semantic alignments for different platform deployments. This process has been refined throughout the project lifecycle by providing additional documentation, examples and integration tests in order to facilitate the development to the maximum extent possible. A set of bridges with corresponding generic syntactic translators, and deployment-specific (i.e. semantics-specific) alignments are part of the final INTER-IoT release. The complexity of bridges development for a specific platform depends on the level of standardisation provided by the platform itself. Platforms providing well-documented, industry-

standard APIs are much easier to integrate. For example, for a platform providing REST API with well-documented JSON or XML data format INTER-MW would already provide a set of helpers and guidelines, while for a platform providing some proprietary interface, or even just database access, much more effort would be required from bridge developers.

The main challenge is, of course, closed platforms that do not provide any documented API. A case by case approach my help finding workarounds, but as a general rule it would be very difficult to ensure a stable and reliable system built upon an undocumented data source.

Real case:

The bridge for FIWARE was the first bridge to be implemented, and this it has been used as a proof-of-concept for the definition of a proper INTER-MW architecture. The outcome has been a clear split of syntactic and semantic part of data transformation.

While developing the BodyCloud bridge, we have encountered the problem of dealing with a distributed platform (installed on smartphones) with multiple endpoints and floating IP addresses. In order to fit into a broad INTER-MW architecture a separate BodyCloud proxy has been developed in order to provide a single HTTP endpoint for the INTER-MW bridge. This approach also helped with the simplification of server network access rules, thus strengthening the overall system security.

While developing the Sofia2 cloud platform bridge we found three different versions of the instances of the same platform. We mainly use the standard API's that provide the platform, but for the instances more personalized of this platform, for example an instance a public institution, to implement some functionalities they use personalized API's that is necessary to readapt in the code of the bridge. This is because Sofia 2 allows developers to publish their ontologies as REST APIs. This is especially interesting in the integration with other systems, because the information is available to be consulted and updated as like an HTTP resource maintaining the same principles of authentication and authorization of the Sofia2 platform. It offers a REST API with custom operations, to access via HTTP methods to the data stored by and invocation to a REST API method by client integrated in the platform. A concrete use of this is to allow Sofia 2 Platform expose ontologies as entities to be consulted through the NGSI v2 protocol.

The IoTivity platform supports information exchange and control based on messaging through the Constained Application Protocol (CoAP). Two modules had to be developed for the IoTivity bridge. The first one is the syntactic translator that uses the same procedure followed for other bridges. The second one is the COAP client responsible for communication with the IoTivity server. When the bridge receives a request the translator translates it to the IoTivity format and the request is then sent to the IoTivity server through the CoAP client.

We have provided few most representative examples of integration challenges we faced, but in fact with each new bridge developed there a new, unforeseen, detail to be solved. But, our open approach has allowed us to always find an efficient solution to integration.

Approach to Platforms ontologies

To successfully interoperate IoT platforms it is crucial to allow them to communicate in a meaningful manner. However, it would be totally unrealistic to assume that all of them “speak the same language”. Therefore, an efficient translation mechanism, supporting/enabling the information exchange is obviously required. To achieve this goal, platforms absolutely need to provide descriptions of data models they expose to the outside world. Such a description can be given via a standard or proprietary/custom ontology, i.e., a (formal) representation of a domain knowledge by a set of concepts and the relationships between those concepts. Of course, the more standardized the platform ontology the less difficult the interoperability problem becomes, as reuse of existing tools and solutions becomes feasible.

Unfortunately, not all IoT platforms have defined their data models rigorously enough, and even fewer did it using explicit ontologies. In the worst case scenario, a platform might not offer any semantic description of the data whatsoever. In such a case an extra effort has to be made to "extract/lift" the necessary information, and build/choose a suitable ontology for it. Another possible problem stems from the fact that semantic technologies are still not widely accepted by the industry, and even simple features like semantically annotated/enriched data formats are not that commonly used. To make the situation even more difficult, at the same time, many of the available semantics-related tools are not really "production-ready" and require specialized knowledge from the user.

In some cases, IoT platform may support semantics (custom or one of the openly available standard ontologies) but uses a data model that contains very particular elements, e.g. description of a port machinery. It is impossible for the interoperability solution to foresee all such situations. Instead, a successful interoperability approach has to offer a modular architecture, allowing for extensions. Preferably, the architecture should also make the process of adding a new platform to the ecosystem as simple/efficient as possible.

Approach:

Interoperability on data and semantics layer is performed by INTER-IoT tool called IPSM (Inter-Platform Semantic Mediator) that is an alignment-based, parametrized semantic translation engine with a distributed, stream-oriented architecture, built with the modular-central-ontology approach in mind. Central ontology can be of arbitrary choice, but within INTER-IoT a well-documented and carefully designed Generic Ontology for IoT Platforms (GOIoTP) has been proposed as the core. Modularity of the approach makes adding new features (relatively) easy. The choice of modules that should be included in the central ontology is one of the tasks when setting an INTER-IoT ecosystem.

Well established, standard tools were selected as a foundation of the proposed solution. RDF – an intuitive standard language for expressing semantics, has already proven its value in the Semantic Web context. Data elements and their semantic relationships are represented via RDF graphs (sets of triples), and such graphs constitute input and output to the translation process

performed by the IPSM. Because the the high popularity of JSON format a JSON-based RDF serialization (JSON-LD), was selected as the INTER-MW data representation.

The process of achieving semantic interoperability, and tutorial for writing alignments are described in a dedicated documentation.

Let us now briefly present approaches to some specific problems that can be encountered during the integration of IoT platform into INTER-IoT ecosystem.

- IoT platform uses data model with no explicit semantics
  It may happen, that IoT platform does not use ontologies and semantically-annotated data, but needs to be integrated into INTER-IoT ecosystem. In such a case, the first step is to select part of the data model that will describe data exchanged with other platforms (not all internal data models need to be exposed to the outside), and "lift" it to ontology i.e. design an ontology or choose a standard one that can be used to describe the data. Methods and methodologies for lifting/extracting semantics from data where analyzed in a dedicated publication and are referenced from the documentation.
  Since, IPSM takes JSON-LD message with RDF graph inside as input, transformation of non-semantic data into semantic-annotated data (with respect to designed/selected IoT platform ontologies) is performed by an INTER-MW bridge component (syntactic translation). IoT platform specific semantic-annotated data can then be translated by IPSM to other semantics.
  The translation is alignment-based so alignment(s) should be created between IoT platform designed/selected ontology and the central ontology. Note, that the alignment needs to be created only to/from modules of the central ontology that are relevant to the platform.
- IoT platform supports semantics
  In this case, the integration is easier since no semantic-annotation needs to be performed by INTER-MW bridge component. Data in the message can be semantically-annotated with IoT platform ontology. The alignment(s) should be created between IoT platform ontology and central ontology.
- IoT platform uses custom ontologies instead of widely accepted standards
  Using custom ontologies is not a problem when preparing for semantic integration. The alignment can be defined between any pair of ontologies. The advantage of using a standard ontology is the possibility of re-using existing alignments that have already been defined and are publically available.
- IoT platform uses data model having no adequate representation in the central ontology
  The central ontology is modular, therefore it is easily extendable. If IoT platform that wants to join INTER-IoT ecosystem uses a specific data model that so far is not represented in the central ontology, it needs to propose a well-documented ontology (new/standard/platform custom) that will form a new module. IPSM does not require physical ontology to be uploaded as part of its configuration, so the new module only needs to be accepted by INTER-IoT administrator and made available to other parties as a part of the ecosystem documentation.

The INTER-IoT semantic interoperability approach has many important advantages.

- A platform can join the ecosystem by simply defining alignments between its semantic data model and the central ontology (in one or two directions) which is considerably easier than separate integration with individual platforms in the ecosystem.

- Central ontology is modular so it is easily extendable without the need to redeploy/restart the system.
- IPSM-AF alignment format is a structured and well documented format for defining alignments that is directly consumed by the IPSM tool.
- IPSM-AF and IPSM support alignments versioning so the configuration is easily updatable.
- INTER-IoT semantic interoperability solution does not required any changes in the data models and data formats used internally by IoT platforms belonging to the ecosystem.

The are two main drawbacks of the approach described above.

- Working with ontologies and alignments requires some degree of semantic expertise. Fortunately, IPSM-AF helps here with its Turtle-based alignment cell format and simple structure.
- IoT platform that does support semantics requires a syntactic translation designed and implemented within INTER-MW bridge. This procedure although well documented, requires understanding of semantic-annotations.

Real case:

Integration at the data and semantics layer was a part of the work done during INTER-IoT pilot deployments, several Open Call projects, and demos that were prepared for Athens review and IoT Week in Geneva. These different application scenarios covered all aforementioned cases, so the proposed approaches could be tested in practice.

- <u>IoT platform uses data model with no explicit semantics</u>
  This is a very common situation that was encountered e.g. in port pilot deployment. In the port environment there were already IoT platforms deployed with their internal data models before INTER-IoT deployment. One of the platforms, collected data from the machinery e.g. cranes and internally used SEAMS2 JSON-based format. The first step to set up an integration, was to design an RDF vocabulary that allowed for an easy representation of any JSON data. In fact, this method of designing naive ontology for JSON format is fully reusable, and is included in examples for syntactic translation that should be implemented in the INTER-MW bridge. Next step, was to design an alignment between the naive RDF representing IoT platform original JSON data and central ontology based on GOIoTP. Such approach can be recommended for integrators that are not experienced in semantics, but want to quickly integrate into INTER-IoT ecosystem. Other, approach would be to design/select proper, more robust ontology for the platform according to ontology design guidelines.
- <u>IoT platform supports semantics</u>
  In INTER-IoT (e/m) Health pilot deployment, one of the platforms being integrated was UniversAAL that internally supported semantics and used RDF as data exchange format. In this case implementation of INTER-MW bridge did not require syntactic translation. The original RDF from UniversAAL message was passed on to INTER-IoT. The alignment was defined between UniversAAL ontologies and central ontology, again based on GOIoTP. Note, that the alignment was not defined between complete ontologies but only between parts that were referenced in the exchanged messages. Defining the alignment between full ontologies is possible and supported, however it requires in depth analysis and is more

time-consuming, so in most cases, to enable fast integration only parts of the data model need to be considered.

- IoT platform uses custom ontologies instead of widely accepted standards
  In INTER-IoT we have experienced both cases, when custom and standard ontologies have been used. Custom ontology has been created within the port pilot for meteorological data. The reason was that, having a set of data originally exchanged in JSON, we could not find any ontology that would allow to semantically annotate them without losing information – no ontology covered the information content from original messages to an acceptable degree. The Valencia port ontology has been defined and used in the bridge to semantically annotate original messages.
  On the other hand, Open Call project INTER-IoT-EWS used SAREF ontology. Within this project an alignment between parts of SAREF ontology and GOIoTP has been defined. The advantage of using well established standards is that such alignments are reusable, or can provide help for newbies in alignment creation.
- IoT platform uses data model having no adequate representation in the central ontology
  This may happen, when an IoT platform wants to join the ecosystem and publish data that has not been exchanged so far in the ecosystem e.g. domain specific data like lighting information. In the port pilot, after first version of the alignment has been created, the requirement to include data from the lighting system came out. Up to that point, such data had not been exchanged so it was not included in the alignment and had no representation in the central ontology. The steps undertaken were to: a) define an ontology module in central ontology for describing lighting data, b) define an alignment from port ontology lighting representation to central ontology lighting module, c) inform interested parties that such module has been included and can be used. Another example one can imagine, is when IoT platform uses SAREF ontology and publishes measurements done by devices but at some point wants to also publish information about devices and their power level. In this case, central ontology should be extended with a suitable power module that is not by default included in GOIoTP.
- Integrator from IoT platform side has no expertise in semantic technologies
  Unfortunately, it turned out that many IoT platform administrators did not find it easy to understand semantics. In such cases, it would be recommended to perform the semantic integration task as straightforward as possible. For IoT platforms using JSON data format, the syntactic translation to naive JSON ontology (as described above) can be used. For other formats, e.g. XML, an integrator can use one of the methods/tools that allow to lift data to ontologies. This will allow to produce “some” ontology, without going into the constructs. However, the next step being the alignment creation, requires at least understanding of RDF and the target GOIoTP module. To make this easier, documentation with many examples has been prepared. Sample alignments created within the project are publically available. A prototype GUI providing support for defining alignments has also been proposed, that will be further enhanced and extended (beyond the scope of the project).

Approach to Platforms versioning

There are different situations that have led to continuous changes of versions of the IoT platforms. Firstly, these platforms appeared few years ago and during this time new technologies, standards and trends have been appearing. Secondly, many platforms were launched in a state of maturity that was not optimal, with the aim to be tested and improved by an open community of developers. But sometimes happens that there are IoT Platforms that state being 'open' but in practice they are inaccessible or not maintained. Thirdly, the companies that are in charge of developing and implementing this type of solutions, usually offer different commercial versions according to the economic investment made in them, therefore offering different functionalities and components according to the acquired version. But in other cases, they adapt the solution to the specific needs of the clients.

These situations identified implies that an IoT Platform has not a unique implementation or offer different versions. Therefore, it is necessary to consider all these factors in order to achieve the correct development of the bridge to connect with the platform.

Approach:

There are several situations in which the different versions or implementations of a Platform must be considered:

- A bridge for this platform has not been previously developed: In this case it is very interesting to carry out a preliminary study of the platform to find the possible modifications that may be required in the future. A good planning at that time will involve fewer changes in the future. It is recommended to create a class related with the utilities of the platform, to access to the concrete functionalities that offer the platform, indicating to which version of the platform this bridge corresponds.

- There is a bridge previously developed to this platform, but it corresponds to a different version: If during the realization of the first bridge the steps indicated in the previous section were carried out, then, it is possible to create a utilities class corresponding to this version and adapt those functionalities that have been affected by the change of version. That implies less effort in the development of bridge. Otherwise, if the developer of the first bridge has not taken into account these changes, the development of new bridge may involve having to create a new version of the bridge adapted to this version of the platform.

- A developer wants to develop a bridge for the platform, but the instance of the platform has not the same implementation as the instance of the platform that has a bridge previously developed: In these cases it usually happens that, however it is the same platform and internally it has the same operation, the interfaces to access to the functionalities are completely different. In these cases it may involve the development a new bridge to a adapt to the new interfaces.

Real case:

To illustrate the first bullet point is interesting to explain the situation during the development of the FIware bridge. The bridge developed for Fiware is the example of a platform with a bridge developed considering the different versions that may appear in the future. In this case it was taken into account that the current version of the platform is the Orion NGSI API v2. Then a class called utils was created in which all the features of this version were included. Therefore, if a later version appeared, a new class of utilities could be created.

To illustrate the second bullet point is interesting explain the situation during the development of the bridge for Sofia2 Cloud Platform. There was access to three versions of the cloud platform. The first was the cloud lab, an instance to test open to the public community of developers. Second, a version of the platform located in a telecommunications company that collaborates with a public entity. Finally, a deployment located in a public entity, with large security measures and which was not allowed access to third parties. Due to the availability of instances, bridge was developed firstly to the cloud lab version. The information indicated in the first section of the previous point was taken into account and a useful class was created for the platform. This facilitated adapting the version of the telecommunications company, since it required minimal changes. Therefore, they could use the same bridge. In contrast, the version developed to connect with the instance public entity required profound changes and a considerable extra effort in the development.

Related with Iotivity Platform developers need to create extra functionalities in the instance of the platform to provide a REST API to connect with the platform. This development was made focused in a concrete instance of the platform, in part, it was influenced because in some aspects the platform has similarities with an SDK. For that reason, the bridge developed is more dependent with this concrete instance of the platform and it is very difficult reuse the bridge for another instance of the platform.

Approach to Platform deployment

An added difficulty of trying to connect so many disparate platforms is the different deployment configurations that can be adopted by those. Some of the integrated platforms have been designed to be deployed only in a cloud-based architecture. Others are precisely the opposite and can only be deployed locally, not in the cloud.

INTER-IoT has to be able to accommodate any option and provide any possible combination to the integrators, regardless of the limitations of the platforms being integrated. This is a challenge that cannot be solved with the design of a particular software module, component or interface, but rather requires an architectural approach. One that will have to be tackled and adapted on each deployment of INTER-IoT, as it will vary depending on the platforms being integrated in each particular case, and the available infrastructure.

But there are certain steps that are repeatable, and we can abstract an approach for these cases.

Approach:

The easiest and more direct approach to tackle this issue is relying on REST APIs*. REST APIs are widely known and adopted, and are the de-facto “standard” for interfaces when dealing with server-based services, whether they are singleton (traditional) servers deployed on the internet, a local network, or clustered or cloud-deployed.

This is applicable in different points of the INTER-IoT integration. The INTER-API is accessible by both integrated platforms and clients through its own REST API, making it accessible through many possible configurations depending on the deployment (deploying it in the cloud, or in a local server, for instance).

The same approach can be adopted from the point of view of the integrated platforms: if they provide a REST API endpoint, this can be used to perform the connections implemented in their respective bridges. Then the bridges can be configured to contact the appropriate endpoint, whether that is in a local instance, an internet server or in the cloud.

For platforms that do not even provide a REST API of their own, because it does not match their archetypal infrastructure, bespoke integrations can be achieved through their bridges, which at the same time can rely on INTER-IoT’s own REST APIs to solve this issue.

** When referring to REST API here we also include quasi-REST API, or simply HTTP APIs: APIs available through HTTP protocol, deployed in a host somewhere accessible for other peers in the infrastructure of the deployment. REST APIs are the most extended way nowadays for implementing such an interface, but here we also address those HTTP APIs that do not comply 100% with the principles of REST (e.g. HATEOAS). We call them all REST API for simplicity.*

Real case:

We will give two opposite examples of this issue: a platform that cannot be deployed in the cloud, and one that can only be deployed in the cloud.

The former is the case of universAAL, used in the INTER-Health pilot. The original design of the universAAL middleware was focused on being deployed in a local machine, at a smart-home environment that may or may not be connected through the internet to other nodes. It is possible to deploy an instance of universAAL in a server, and indeed it provides a REST API to access its features if that is the case. It even includes multi-tenancy features to act as a centralized node for several smart-homes. But it is still not prepared for a cloud-based deployment, ready for dynamic scalability and replicability. Every deployment of universAAL has to be tailored to each case, such as the INTER-Health pilot: a universAAL instance was prepared and run in the dedicated server of the pilot, as a local service, accessible only through the local network. This is how the INTER-MW and its unviersAAL Bridge contacted the universAAL platform. The bridge connects and translates to and from the universAAL REST API. It is configured to contact such API to the local service and port, instead of accessing a remote cloud endpoint, which is not possible with universAAL.

In the latter case we find Sofia2, which in its open distribution does not allow for on-premises deployment, thus it had to be used in a cloud endpoint. For testing purposes it was possible to use Sofia2 own sample cloud endpoints, but for a proper deployment, it was necessary to rely on the cloud endpoint owned by a private company, and then a public entity. This required additional privacy provisions and authorizations, but from the technical point of view, the main obstacle was the different versions of the APIs in each cloud endpoint (as explained in the versioning approach. Being deployed in the cloud, these API may change overtime). In any case, all these obstacles had to be resolved at bridge level (and its consequent configuration).

Approach to Platforms functionalities

There are a lot of IoT platforms and in a project like Inter-IoT that it is related with interoperability, a series of basic functionalities that should be offered by IoT platforms were established. Among them the ability to subscriptions, actuations, virtual devices management and discovery. The platforms offer access to its functionalities, offering their own methods. The main functionalities are always covered by the platforms. But in some situations, it may be necessary to make some module or component to obtain the information in the desired way or format or to perform some action in the platform.

The most difficult case is a platform that does not have a concrete functionality implemented and therefore the bridge does not have the capacity to perform that functionality. Also, related with the absence of functionalities is the case of functionalities sometimes IoT Platforms that state being 'open' but in practise they are inaccessible, not maintained or not finished.

Approach:

When a bridge is implemented to connect an IoT platform with Inter-MW it is necessary to know the platform and the functionalities that it offers. Therefore, it is a very important task to see what methods can be used to access to these functions. In those cases, in which it is detected that some functionality is not available, it is necessary to perform a function or component that is responsible for performing the necessary actions to obtain the information or perform the desired actions. Not offering such functionality is the last option. It can implies create a module to implement a way to access to that function in the platform or add a component in the bridges.

The case of platforms not maintained, is more difficult. In many cases, some platforms have been discontinued, for example, because they have become part of another project. But sometimes the documentation of the platforms that are not maintained is not easy to follow. Because developers are improving functionalities but they don't document how to use it. Sometimes platforms provide deployments of their functionalities in Docker or in a virtual. They have helped to make it easier to test and have access to some functionalities, because it makes it easier to have an accessible environment in which they work and to know what is the real status of the platform.

Real case:

A platform not offering an expected functionality is not a very common situation, because all the platforms usually cover the basic needs that bridges require. But in some cases, some small adjustments have been required.

A concrete example of a platform that does not implement direct mechanisms to support some functionalities can be explained with the example of Fiware. This platform stores the current state and context information of the devices, but in that case this platform not implement a direct method to perform actuation in its sensors using its Context Broker. To solve it, some adjustments have been made. At the level of the platform there are the possibility of use agents or create subscriptions to the information of sensors to develop scripts that perform the actuation when the

subscription notifies a change of state of the sensor. At the level of the bridge the way to perform that action is the same that send an observe to the bridge.

In the case of platforms with pending or not maintained functionalities. We have used its forums to find a solution, contacted the main developers of the platform or open issues in the repositories. These platforms usually has few documentation and is difficult to follow.

Approach to Platforms security
Each integrated interoperable platform has its own approach regarding security. It not only depends on how to allow a particular client to the platform but also how to secure its communication, its stored data, etc… It is not possible for ITNER-IoT to enforce a unified security approach for all integrated platforms, since security is integral to each independent platform and they cannot be modified to be plugged into INTER-IoT (that would defy the interoperability goal). However, there are some approaches to follow when securing an overall ITNER-IoT deployment, with a particular infrastructure and a given set of integrated platforms. In addition to this, INTER-IoT provides its own security mechanisms for its own features. All together, combined with the security features, ought to provide a desired level of security for the whole solution. Even in cases where some of the integrated platforms do not provide enough security in specific scenarios, the overall solution orchestrated by INTER-IoT should be enough to secure the entire deployment.
Approach: The INTER-API is properly secured through authentication tokens, so all connections from platforms or clients through that API can be considered safe (with the appropriate provisions). Connections to and from platforms to their respective bridges depend on each platform security implementation. Security in these cases depends on the proper setup of the security environment for each platform when installed individually in each deployment. There is no one-size-fits-all solution but rather recommendations and proper steps to follow when installing them. This is up to the staff that takes care of the installation and deployment. The more complex security mechanisms imposed by some platforms are solved at the level of their bridges, as well as following the specific installation instructions for them. On the other hand, platforms that have lower security requirements will have to depend on the security provisions made at each deployment to increase the level of security over their basic features. This can be achieved through proper configuration of the network and the hardware and premises where it runs.
Real case: In the INTER-Health pilot there were a set of security restrictions imposed by both European and local regulations. A particular case, for example, is that of universAAL, which was installed in this pilot. The universAAL REST API relies on HTTP Basic Authentication, which on its own does not provide proper security. An alternative was to use the HTTPS version of the API (by properly configuring the platform for this feature). In addition to this, the infrastructure took advantage of the scenarios and use cases of the pilot to limit the use of the universAAL platform only within the premises of the pilot, utilizing it only from within their secured intranet. This adds a layer of security at the network level. The data stored from the platform was encrypted, which complies with the regulation, and the hardware was physically isolated and accessible only by authorized staff.

In the INTER-LogP pilot each of the three IoT platforms involved has its own security. The port and the terminal have the IoT platform installed in their premises for data privacy reasons. Regarding the access security, the port (WSO2) and the haulier company (Azure) platforms provide a Rest API with Oauth2 and SSL, so you need a token and a certificate to access. The access to the terminal network is very limited and it is necessary a VPN to connect to the different services.

In the case of Sofia2, the cloud endpoint deployed in a public entity came with its own set of strong security and privacy restrictions. Among these it required a specialized library to deal with its authentication mechanism. This procedure was not needed with the basic authentication used in the lab cloud server at first, so it had to be later integrated into the bridge.

Approach to privacy in industrial sectors

In an industrial sector, like the port, where there is a huge competition between different companies is a hard task the interoperability of data among companies. Right now, they are only sharing the minimum documentation required, for instance by a public body such as customs.

In addition, another reason for not sharing data with other entities, is that their systems are not prepared to operations. This new requirement supposes a considerable economic and personal effort, so the benefit should be must be greater than the problems caused

In case, there is an agreement to exchange some data, the companies want to be sure that the data is accessible by authorized people and it is used for the defined purpose.

Approach:

The process necessary to get a port stakeholder to participate in the project is as follows:

- Interview with the company managers where the INTER-IoT project is explained. In this meeting you have to introduce the project and the objectives. You can also show the architecture and the security processes. But the main issue should be the benefits for the company. Maybe is necessary other meetings with technical people from the company.
- Before start any integration it is important to sign a confidentiality agreement between the company and the INTER-IoT partner where is described the data that is going shared, who is going to access to the data and how is going to be used.
- As part of the previous point it is necessary to explain the different security mechanisms in the INTER-IoT components to grant the privacy of the data described in the confidentiality agreement.
- Finally, you can start the integration with the company systems. It is likely that the company cannot allocate resources to the integration, so you must be prepared to perform the integration with the support of the company's technicians.

Real case:

During the stakeholder analysis we identified several companies that we were interested in them participating in the project. So we contacted with some of them to see their availability to participate in the port pilot (INTER-LogP).

The main actors of the pilot are the port and the terminal. Although both are part of the INTER-IoT consortium, during the project proposal, we had several interviews with the managers to explain the project objectives.

The other actor in the pilot is the haulier company. We had interviews with 3 companies that were willing to participate. In this meetings we explained the project objectives, and the architecture and data used in the pilot. Finally, we decided to install the devices manufactured by company called Traxens. These devices can be used to track containers or trucks.
The first stage was to sign a confidentiality agreement with them regarding the use of the data. After that we worked in the implementation of the INTER-MW bridge using its API.

Approach to privacy in healthcare sectors

In the healthcare sector the preservation of privacy is a major concern for any IT solutions in the field. These solutions are dealing with very personal and sensitive data, and as a consequence it is a heavily regulated environment, from the new GDPR to local regulations. These impose many restrictions to the technological solutions being used, not just technical but procedural and legal too.

Another obstacle is that so far it is difficult, at least in some healthcare environments, to find integrated IT solutions. There are many subfields (almost as many as practitioners) in any given healthcare institution. There may be IT solutions for each, and historically, both because of the regulations mentioned above but also due to many other factors, these systems have not usually been interoperable, to say the least.

Approach:

The process necessary to get a health stakeholder to participate in the project is as follows:

- Interview with the company managers where the INTER-IoT project is explained. In this meeting you have to introduce the project and the objectives. You can also show the architecture, and the security processes. But the main issue should be the benefits for the company. It would be worthy to meet with other clinical and technical people from the company.
- Then, next step is to define the purpose of the integration in terms of final solution: IoT platforms, services, devices… to be integrated by means of INTER-IoT.
- Before starting any integration, it is important to meet the General Data Protection Regulation (GDPR), which includes ethical and security aspects. For that purpose, first step is to check the corresponding directives at national (per country) and European level. For mHealth solutions, that may include software and medical devices, the following documents are the most relevant:
    - Regulation UE 2016/679 (Directive 95/46/CE). Protection of individuals with regard to the processing of personal data
    - Directive 93/42/CEE. Medical devices
    - Directive 2014/31/UE. Non-automatic weight instruments
- Thus, it is important to sign several agreements between the company and the INTER-IoT partner. Identifying the following documents:
    - Agreement of external data processor. It describes the data is going to be shared, who is going to access to data and how is going to be used, in terms of ethics and security.
    - Loan of use. If the technical partner lends medical devices to the health stakeholder during a test phase, this document should be signed. It should include the number of devices, description of each one, if possible serial numbers to identify unequivocally the devices, period of loan and any action related to malfunction, and/or responsibilities.
- Furthermore, to carry out any intervention with subjects/patients requires the approval of the corresponding Bioethical Committee to assure that GDPR is being accomplished.

Derived from the above regulations and directives, documentation to be presented into ethical committees should include: Clinical trial protocol, Informed consent, Information sheet, Information on the processing of personal data, Case report form, and the two documents described for the agreement of external data processor and loan of use agreement.

- Once the Bioethical committee approves the protocol, you can start the integration with the company systems. It is likely that the company cannot allocate resources to the integration, so you must be prepared to perform the integration with the support of the company's technicians.

Real case:

ASLTO5 was already part of the consortium, so steps done were mainly focused on:

- Identification of the needs of the health operator, being translated into scenarios, use cases and finally design of the whole eHealth system (functionalities). It was done by having several meetings in person at the ASLTO5 premises.
- Selection of the medical devices that fit with the EU regulation.
- Preparation of the agreement of external data processor and loan of use.
- Preparation rest of documentation for the Bioethical Committee. Apart from the EU regulations and directives, we had to consult the DL 196/2003. Garante Privacy applied in Italy.
- Implementation and testing of the solution.
- Finally, it was deployed in ASLTO5 premises in Torino (Italy).

Approach to Devices setup

One of the most common problems faced by the IoT deployments is to finding devices (sensors, actuators)
capable to do the task(s) required for the use cases that are lead to the deployment of an IoT network. In some cases, there is a great variety of sensors and in some other cases, there are few. Selecting the appropriate sensors for a use case usually requires of working with different manufacturers and perhaps different interfaces.

In this scenario, different devices can have different configuration mechanisms, such as using a form integrated in a web server, modifying a configuration file stored in the operative system, using a REST API or sending commands through a communication channel. This represents a difficulty for D2D layer and MW2MW layers since this kind of operations are difficult to be standardized, preventing to be inclusion of these features in interoperability layers.

Approach:

The interoperability layers contain the features that are considered generic enough to be managed in a single point. Thus, the configuration of devices has been kept out of the scope of the bridges and the gateway. To perform the configuration of devices following the INTER-IoT methodology and deployment mechanism it has been followed this script:

1) Identify the mechanism of configuration of specific devices. If this is automatically done by the IoT Platform, then the problem is automatically solved.
2) If they need specific configurations, check if they can be automated or not.
3) If they can't be automatized, specific actions to configure them must be done. These actions must be performed by the system integrator or the use case owner (depending on the responsibility schema of the IoT installation).
4) If they can be automatized, develop the mechanisms and include them in the initialization scripts of INTER-IoT.
5) Include the new scripts in the containers of the affected modules according to the standard deployment approach.

This approach has the advantage that it can be performed easily with small complexity in most of cases. Also, usually it has to be fulfilled only once since the majority of sensors have persistent configuration. The disadvantage is that there is not developed a standardized way to deal with deployment so it avoids automation. With some research, this task and could be part of future work of D2D/INTER-MW modules.

Real case:

Sensors from E3tCity need to be configured by sending a sequence of custom commands in a specific MQTT topic on a cloud or local deployment of an MQTT broker. Other devices (such as terminal lighting or heavy machinery sensors) are pre-configured in web interfaces and this configuration keep persistent. For non-persistent configuration, specific scripts were developed in order to make possible a kind of automatic configuration.

Approach to Custom solutions

In industrial cases, there are numerous platforms that are built initially to solve specific uses. Some of them evolve and become a kind of IoT platform. Due to this, in most of these cases, they do not follow standards, best practices or methodologies, which make harder to adapt interoperability mechanisms as the ones developed in INTER-IoT. INTER-IoT frequently relies in the idea that to perform IoT-related taks, devices and platforms must perform similar activities that can be generalized.

In this scenario, is difficult to find these generalizations since some of the common methods can be missing because they are not specifically needed for the initial features of the platform.

Approach:

INTER-MW has addressed this with several strategies:

1) Interoperability methods in bridges are always mandatory to be implemented. However, these can return that the function is not supported. From the programming point of view, the interoperability interface is propagated to the bridge but at the same time allowing the “Not supported” feature provides flexibility to face real scenarios.
2) Some features can be emulated by doing some extra programming in the bridge side.
3) If the platform is under development or is actively maintained, a request for changes is issued to the development team.
4) Some features are mandatory due to the INTER-MW architecture. If these cannot be emulated, then the bridge is not possible and the platform is not considered compatible with INTER-IoT.

For INTER-IoT is important to provide simple yet powerful mechanisms of interoperability to as much platforms as possible. The strategies followed ensure a high number of compatible platform and also identify those platforms that due to their construction they are far from the standards and must be considered a special case.

Real case:

Some examples of real cases are:

- FIWARE Orion does not support actuation method: although Orion is not an industrial platform, it lacks of actuation mechanisms. This method was implemented as “unsupported”.
- Discovery of sensors is not possible in SEAMS2 Platform. This is emulated in INTER-IoT by listing the sensors and sending to INTER-MW.

Subscription mechanism is not supported in Bodycloud. It is emulated by doing periodical data polling from the bridge side to the platform.