

## **AI-PROFICIENT**

Artificial intelligence  
for improved *production efficiency*,  
quality and maintenance

# **Deliverable**

**D5.1: Communication middleware and IIoT interoperability – design and specification**

**WP 5: AI-PROFICIENT system integration and deployment**

**T5.1: Smart component integration and IIoT interoperability**

**Version: 1.0**

**Dissemination Level: PU**



# Table of Contents

Table of Contents .....	2
List of Figures .....	3
List of Tables .....	4
Disclaimer .....	5
Executive Summary .....	8
1 Introduction .....	9
1.1 Scope .....	9
1.2 Audience .....	10
1.3 Relations to other tasks and work packages .....	10
1.4 Structure .....	10
2 AI-PROFICIENT platform middleware .....	11
2.1 Message broker .....	11
2.2 Data layer .....	12
2.2.1 Relational database .....	12
2.2.2 Time series database .....	13
2.2.3 Semantic repository .....	16
2.2.4 Visualization for development – Grafana .....	16
2.3 Orchestration of system components .....	18
2.4 Canonical data model .....	19
3 Interface to smart components and edge AI .....	23
3.1 Industrial Internet of Things (IIoT) interoperability .....	23
3.1.1 Technical interoperability .....	23
3.1.2 Syntactic interoperability .....	23
3.1.3 Semantic interoperability .....	24
3.2 Communication protocols and integration .....	27
3.2.1 Profinet, Profinet CIFS .....	27
3.2.2 MQTT .....	27
3.2.3 OPC DA (Classic) .....	28
3.2.4 EthernetIP .....	28
3.2.5 DeviceNet .....	28
3.2.6 INOS Standard Ethernet Socket .....	28
3.2.7 Profibus, Modbus (via scripting interface) .....	28
3.2.8 Database triggering via polling .....	29
4 IT infrastructure at pilot sites to host AI-PROFICIENT platform .....	29
4.1 Continental .....	29
4.1.1 Virtual Machine .....	29
4.2 INEOS Geel .....	31
4.3 INEOS Cologne .....	32
5 Conclusions .....	33
6 Acknowledgements .....	33
7 References .....	34

## List of Figures

Figure 1 AI-PROFICIENT platform architecture (D1.5).....	9
Figure 2: Relation of Task 5.1 with other tasks.....	10
Figure 3: AI-PROFICIENT platform middleware .....	11
Figure 4: PHPMyAdmin web interface for managing relational database .....	12
Figure 5: InfluxDB main dashboard .....	13
Figure 6: InfluxDB data visualization .....	14
Figure 7: InfluxDB dashboards.....	14
Figure 8: Grafana main screen.....	17
Figure 9: Grafana dashboard .....	17
Figure 10: Observation classes in SOSA .....	19
Figure 11: Actuation classes in SOSA.....	19
Figure 12: CDM classes.....	20
Figure 13: Interoperability layers [10] .....	23
Figure 14: (a) Direct protocol conversion, (b) Canonical data model .....	24
Figure 15: Presentation of the steps to connect to the Virtual Machine at Sarreguemines plant.....	30
Figure 16: Description of the Remote Desktop Protocol .....	30
Figure 17: Description of the Virtual Network Computing Protocol .....	31
Figure 18: Remote access protocol Ineos Geel site .....	32
Figure 19: Remote access protocol Ineos Cologne site.....	33

List of Tables

Table 1: Fields of Observation object ..... 20

Table 2: Fields of Actuation object ..... 21

Table 3: Fields of Sensor object..... 21

Table 4: Fields of Actuator object..... 22

Table 5: Fields of Observable/Actuatable Properties..... 22

## Disclaimer

This document contains description of the AI-PROFICIENT project work and findings.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any responsibility for actions that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the AI-PROFICIENT consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the Member States cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors (<http://europa.eu/>).

AI-PROFICIENT has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957391.

**Title: D5.1 Communication middleware and IIoT interoperability – design and specification**

<b>Lead Beneficiary</b>	IMP
<b>Due Date</b>	28.02.2022.
<b>Submission Date</b>	28.02.2022.
<b>Status</b>	Final
<b>Description</b>	Design and specification of AI-PROFICIENT Communication middleware and IIoT interoperability
<b>Authors</b>	IMP, CON, TEK, INEOS, INOS, TF
<b>Type</b>	Report
<b>Review Status</b>	PC + TL accepted
<b>Action Requested</b>	For acknowledgement by partners

VERSION	ACTION	OWNER	DATE
0.1.	Initial table of contents	IMP	09.12.2021.
0.2	Addition of the Continental part	Continental	18.01.2022
0.3	Contribution	INEOS	31.01.2022.
0.4	Contribution	IMP	01.02.2022.
0.5	Contribution	TEK	04.02.2022.
0.6	Contribution and review	TF	10.02.2022
0.7.	Draft for approval	IMP	11.02.2022.
0.8	Corrections by partners	IMP, TF, TEK, INOS, CONTI, INEOS	17.02.2022.

1.0	Final version submitted	UL	25.02.2022.
-----	-------------------------	----	-------------

## Executive Summary

The Deliverable D5.1 is a public document of AI-PROFICIENT project delivered in the context of WP5, Task 5.1 (Smart component integration and IIoT interoperability), with regard to the design and specification of the approach which has been taken by the involved partners.

The aim of this document is to provide the design and specification of AI-PROFICIENT platform middleware and interface towards smart components and edge AI. First, we present the main components of AI-PROFICIENT platform middleware. Next, we provide the description of interfaces towards smart components and edge AI which encompasses the interoperability aspects, communication protocols, canonical data model and semantic interoperability. Finally, we present the approach that will be used in pilot sites to implement the aforementioned middleware, having in mind the security constraints at the plants and the data protection.

# 1 Introduction

## 1.1 Scope

The aim of this document is to provide the specification of AI-PROFICIENT platform middleware which will serve for integration of core AI-PROFICIENT services and system components. More specifically, the related task 5.1 will employ MQTT broker to serve as a communication hub offering connectivity to different field-level equipment, platform services, in addition to plant management systems. Data management and components orchestration will also be presented. Next, this document will describe the interfaces used to communicate with the plant systems, communication protocols and message formats. Finally, in this document we provide more details on how the aforementioned design will be implemented in three pilot sites: CONTINENTAL, INEOS Geel and INEOS Cologne. Namely, due to strong security measures, with respect to access to monitoring data and machinery, there is a need to deploy the instances of the platform in dedicated computers hosted in plant premises.

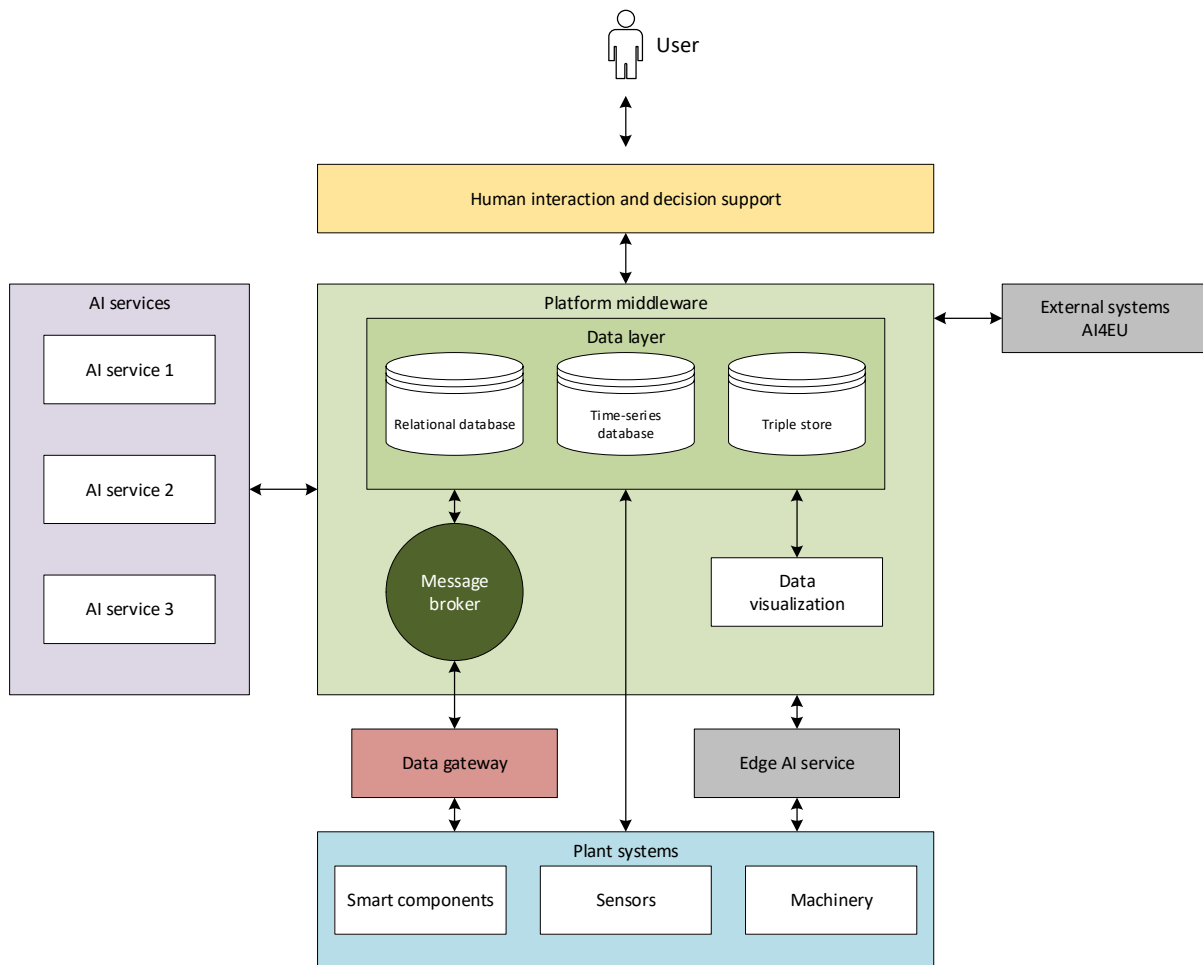


Figure 1 AI-PROFICIENT platform architecture (D1.5)

The build of this document comes from the deliverable 1.5 where the AI-PROFICIENT system architecture (see Figure 1) has been described considering the review of existing standards, former EU project and commercial solution (D1.5 section 3) and crossing them with AI-PROFICIENT requirements from D1.4. The high-level description of the platform has been provided in D1.5 section 4. The present document will detail the description of the platform middleware.

## 1.2 Audience

The intended audience for this document is the members of AI-PROFICIENT consortium, Project Officer as well as the general public, since this document is public. More specifically, the consortium members in charge of development, integration, AI service deployment and platform implementation are expected to use the content presented here.

## 1.3 Relations to other tasks and work packages

As it is shown in Figure 2, Task 5.1 takes inputs from general AI-PROFICIENT Scientific and Technical Objectives (STOs) and Task 1.5 which provides AI-PROFICIENT system architecture. This task provides outcomes which will be used in subsequent activities within the following tasks:

- T5.2 – Semantic knowledge graph for integrated digital twins
- T5.3 – Data privacy, protection and security measures
- T5.4 – Integration with AI4EU platform, data sources and services

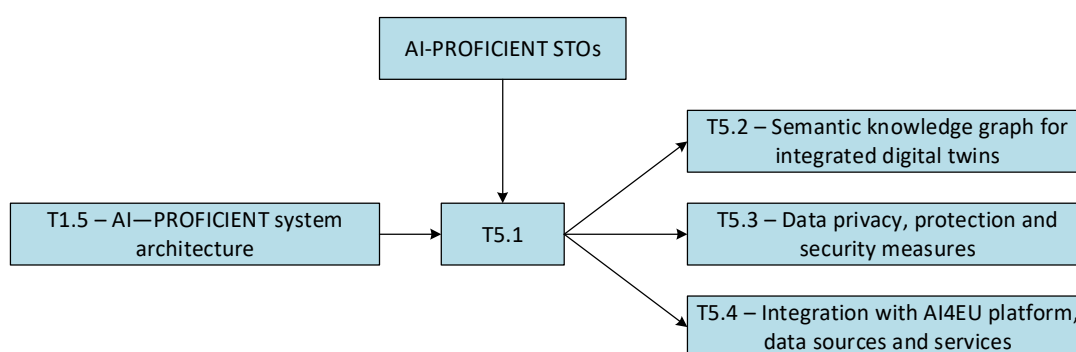


Figure 2: Relation of Task 5.1 with other tasks

## 1.4 Structure

The present document is divided into the following sections:

- **Section 1** provides the introduction to the content presented in this document
- **Section 2** presents the AI-PROFICIENT platform middleware
- **Section 3** provides detailed description of interfaces, communication protocols, and semantic repository
- **Section 4** presents the implementation at pilot sites which considers the platform middleware and security constraints of the pilots
- **Section 5** concludes the document

## 2 AI-PROFICIENT platform middleware

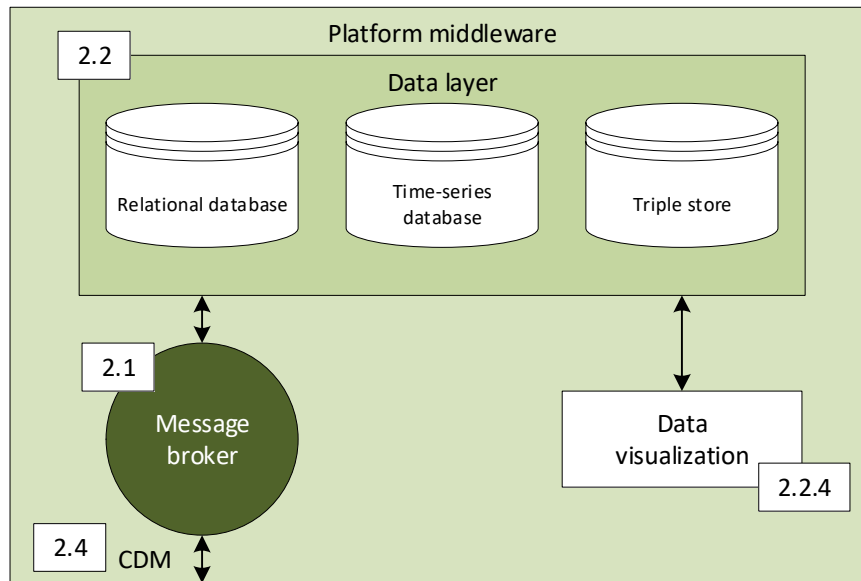


Figure 3: AI-PROFICIENT platform middleware

The focus of the present document is on the platform middleware (Figure 3), which aims to integrate the AI-PROFICIENT services and other system components. AI-PROFICIENT services will be developed within the corresponding tasks of WP2, WP3, and WP4. The high-level description of the services is provided in Deliverable D1.5, Section 4.4. Moreover, middleware will enable acquisition, storage and retrieval of the collected data from the project pilot plants. As it is shown in Figure 3 above, the AI-PROFICIENT middleware consists of the following components:

- Message broker (described in section 2.1)
- Data layer with the database software (described in section 2.2) and the Canonical Data Model (CDM) (described in section 2.4)
- Data visualization for development (described in section 2.2.4)

In the sequel, we will provide more details for each of them.

### 2.1 Message broker

The aim of message broker is to enable communication between different system components by using the publish/subscribe message exchange pattern. MQTT protocol, which is commonly used in different Internet of Things (IoT) applications, supports custom payload type, text or binary. The publish/subscribe communication pattern allows the sending and receiving end to be decoupled from each other, which provides the following benefits:

- One Publisher can send a single message to many different Subscribers
- One Subscriber can receive messages from many different Publishers
- It is not required that Publishers and Subscribers be aware of each other

When compared to classical client/server architecture, the publishers and subscribers do not communicate directly. Instead, they are using a messaging broker which manages the communication between them. More specifically, Message broker filters messages received from different publishers and sends them to the corresponding subscribers. In such a way, publishers and subscribers need to know the connection details of message broker and are not required to be connected at the same time.

MQTT protocol uses a subject-based message filtering. In particular, each MQTT message has a topic – information used by the message broker to send the message only to the clients which have been

previously subscribed to that same topic. The MQTT topic consists of one or more topic levels where each of them is separated by a forward slash (e.g., plant/machine3/sensor4/pressure). An MQTT client is not required to create the specific topic in advance before publishing and subscribing to it. Instead, the broker automatically accepts the topic.

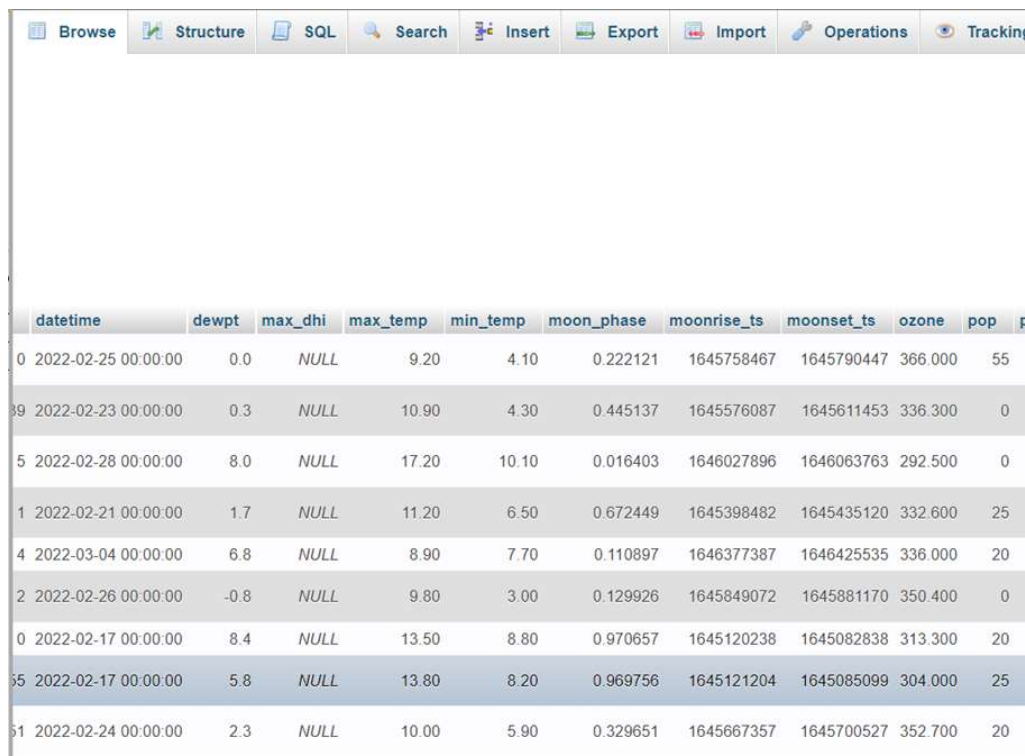
## 2.2 Data layer

Data layer represents a part of AI-PROFICIENT middleware which will provide the data storage and access capabilities to other parts of the platform. The data collected from the pilot sites and generated by the AI services are of different types (timestamped measurements, vectors, etc.). Depending on the data type, the specific data storage technology will be adopted. At the moment, the relational database, timeseries database and triple store (semantic repository) are planned to be deployed in order to fulfil the requirements of the envisioned AI services and plant systems. Nevertheless, once the service development reaches the final stage, the list of available databases will be reviewed, and additional ones will be added if needed. In the sequel, we will describe each of them. More details on their deployment and configuration will be provided in the subsequent deliverable D5.7 which will describe the implementation of AI-PROFICIENT middleware.

### 2.2.1 Relational database

AI services which will be developed within AI-PROFICIENT project will take as an input the measurements from the pilot plants, perform data analysis and provide results which will be presented to the end users via user interface. All the results of the AI services needs to be stored in the database, so that it is available to UI for query and visualization. It is envisioned that AI-PROFICIENT platform will have an instance of MySQL database installed. The exact database schema will be reported once the AI service development reaches the final stage. In order to ease the development tasks, a web interface for database management – PHPMYAdmin (see Figure 4) will be setup.

The Canonical Data Model that defines structure of the data model to be supported by the database is described in section 2.4.



The screenshot shows the PHPMYAdmin web interface for managing a relational database. The top navigation bar includes links for Browse, Structure, SQL, Search, Insert, Export, Import, Operations, and Tracking. Below the navigation bar, a table of data is displayed. The table has columns for datetime, dewpt, max\_dhi, max\_temp, min\_temp, moon\_phase, moonrise\_ts, moonset\_ts, ozone, pop, and a final column with a partial 'p' label. The data rows show various meteorological measurements over time, with some values being NULL.

	datetime	dewpt	max_dhi	max_temp	min_temp	moon_phase	moonrise_ts	moonset_ts	ozone	pop	p
0	2022-02-25 00:00:00	0.0	NULL	9.20	4.10	0.222121	1645758467	1645790447	366.000	55	
39	2022-02-23 00:00:00	0.3	NULL	10.90	4.30	0.445137	1645576087	1645611453	336.300	0	
5	2022-02-28 00:00:00	8.0	NULL	17.20	10.10	0.016403	1646027896	1646063763	292.500	0	
1	2022-02-21 00:00:00	1.7	NULL	11.20	6.50	0.672449	1645398482	1645435120	332.600	25	
4	2022-03-04 00:00:00	6.8	NULL	8.90	7.70	0.110897	1646377387	1646425535	336.000	20	
2	2022-02-26 00:00:00	-0.8	NULL	9.80	3.00	0.129926	1645849072	1645881170	350.400	0	
0	2022-02-17 00:00:00	8.4	NULL	13.50	8.80	0.970657	1645120238	1645082838	313.300	20	
5	2022-02-17 00:00:00	5.8	NULL	13.80	8.20	0.969756	1645121204	1645085099	304.000	25	
51	2022-02-24 00:00:00	2.3	NULL	10.00	5.90	0.329651	1645667357	1645700527	352.700	20	

Figure 4: PHPMYAdmin web interface for managing relational database

### 2.2.2 Time series database

It is expected that a large number of measurements will be collected from the pilot sites in AI-PROFICIENT project. All these measurements can be modelled as a timeseries data, where each measurement has a measured value, timestamp, and attached metadata (e.g., sensor identifier). A timeseries database is a specific type of database which is suitable for storage of time series data. In the market, there exist different timeseries database vendors. Based on previous positive experience, consortium members have decided to employ InfluxDB [1].

InfluxDB is designed to handle high write and query loads, which makes it perfect for large plants where hundreds and thousands of digital and analog signals are collected in real time. Usually, there is a need to query such data by e.g., data analytics software. Beside legacy InfluxQL querying language, since version 2.x, InfluxDB also supports Flux. Flux is a fourth-generation programming language designed for data scripting, monitoring and alerting. As a functional language, queries can be structured by separating common logic into functions and libraries that are easily shared and help speed development. Flux can also be used in order to combine time series data with other SQL data stores (Postgres, Microsoft SQL Server, SQLite, and SAP Hana) along with cloud-based data stores (Google Bigtable, Amazon Athena, and Snowflake). Enriching time series data provides additional information and context which can provide further insights into the collected data.

For managing databases and administrative tasks (e.g., creating access tokens for other users), InfluxDB provides a web interface, as it is shown in Figure 5.

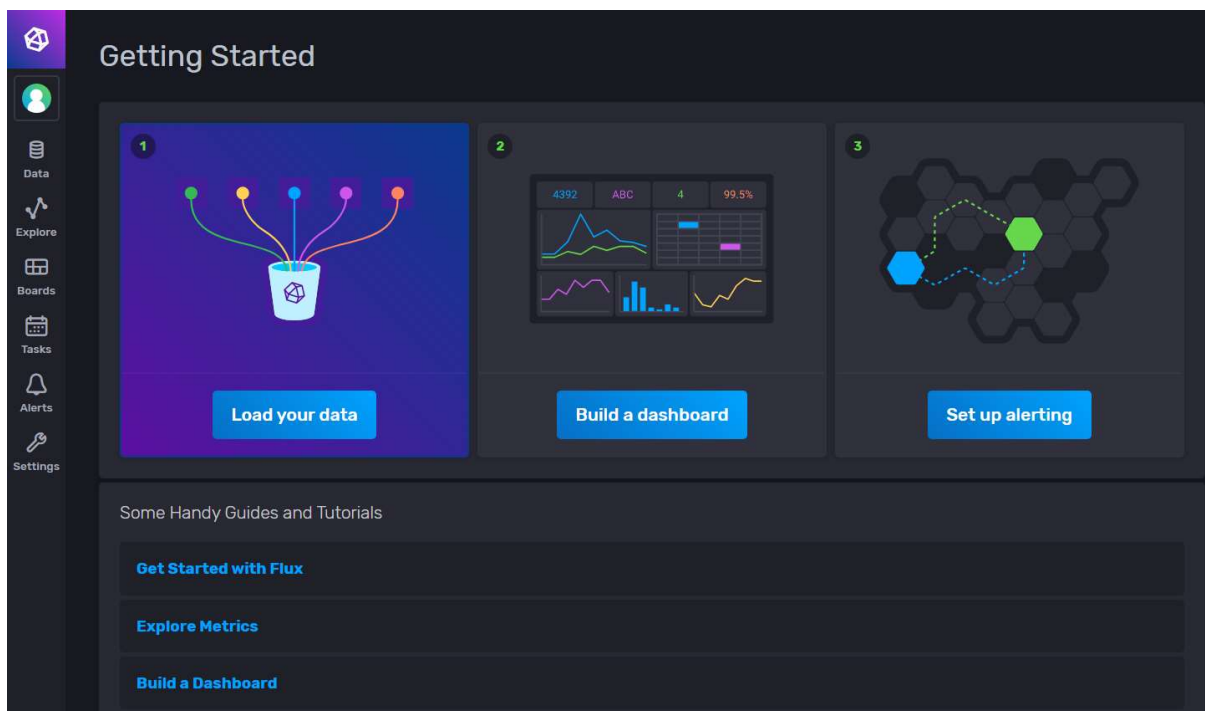


Figure 5: InfluxDB main dashboard

On the left-hand side, there exist a number of options, where one of them allows a user to visualize the collected data, as it is shown in Figure 6.



Figure 6: InfluxDB data visualization

Besides, there is a possibility to create new dashboards that could contain multiple data from different sensors (see Figure 7).

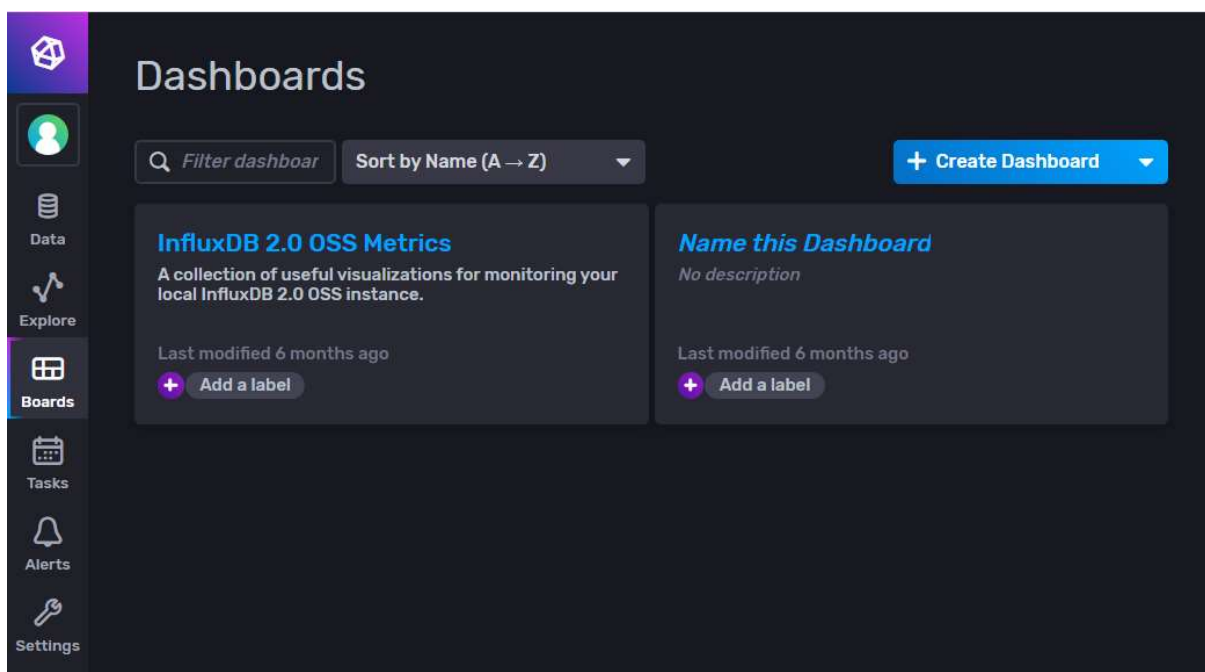


Figure 7: InfluxDB dashboards

In addition to simple querying of data, InfluxDB can perform different functions such as aggregation, integration, sum, mean, etc. All these are performed via the same API, which can be accessed on the specific port.

To be able to understand the InfluxDB schema, the following concepts need to be understood:

- Database – is a logical container for different time-series data, users, retention policies, and continuous queries,
- Measurement – represents the data stored in the associated fields,
- Tag keys and values – used to store metadata. Tag keys are indexed so that queries that are performed on tag keys are faster,
- Series – stands for the collection of data that share common measurement, tag set and retention policy,
- Field keys and values – store metadata and the actual measurements. These can be of different types: integer, float, string and boolean. Fields are not indexed, and each field value has to be associated with a timestamp. Query on field values has to go through all points that match the specified time range and consequently are not efficient,
- Retention policy – specifies for how long the data will be kept in the database. By default, it is set to infinite (no data are removed).

In AI-PROFICIENT, an instance of InfluxDB will be configured for each plant. As it will be described in Section 4, due to security constraints of each pilot site, there is a need to deploy an instance of the AI-PROFICIENT platform within plant premises.

### 2.2.3 Semantic repository

The semantic repository has 2 purposes:

- publishing the semantic data models used in AI-PROFICIENT
- collecting and storing data according to the semantic data models

For the first, the OSLO toolchain will be deployed. The OSLO toolchain is an Open-Source publication environment using GitHub and free trier online resources. It is the backbone of the data interoperability program of the Flemish Government, Belgium, called Open Standards for Linked Organizations. This toolchain produces a number of artifacts such as RDF [2] vocabularies or JSON-LD [3] context files and SHACL [4] shapes that can be used to create, manage and validate the data produced for the second use case. For this, the necessary publishing platform will be setup. The realized architecture of this will be reported in in the final deliverable on the system integration and deployment (D5.8).

The second use case can be supported with RDF store technology such a Virtuoso Opensource [5]. RDF stores are graph databases, based on the RDF graph model. Over the years, the maturity of these RDF stores has grown and today they are applied by publishers such as the Publication Office of the European Union and Wolters Kluwer Germany. RDF stores excel when complex ad hoc queries have to be posted on the whole knowledge base. For instance: “Provide me all operational production units related to observations made by light sensors that are been manufactured in China for which the accuracy level was below par”. Such complex queries arise often during data analysis to better understand the situation. For this, SPARQL [6] as query language on top of an RDF graph is an excellent choice.

When this use case in not pertinent, there is no need to the setup a whole infrastructure, just copying and storing the data in another representation. Nevertheless, it is important that this infrastructure can be bootstrapped when required. In this way, the day-to-day management of the infrastructure is reduced, while still guaranteeing the ability to create the RDF graph stored in an RDF store and thus queryable via a SPARQL endpoint. In order to achieve this objective, the following actions are foreseen:

- annotating the data that are made available via the Message Broker, so that a conversion of the data can be done to RDF, for instance using JSON-LD
- an infrastructure deployment instruction (following the principle of infrastructure as code) which allows to bootstrap an RDF store hooking itself to the Message Broker and other information sources.

### 2.2.4 Visualization for development – Grafana

It is necessary to provide a user-friendly visualization tool which is capable of showing the data collected by the platform. Such visualization is needed especially during the development of the services, since it allows the platform developers to query and visualize different data stored in platform databases. Grafana [7] is a well-developed software which is used via web browser which fits the requirements. After login, the user is presented with the main screen, as it is shown in Figure 8. The user can configure a number of dashboards, where the specific data from the database (relational or timeseries) are visualized (see Figure 9) .

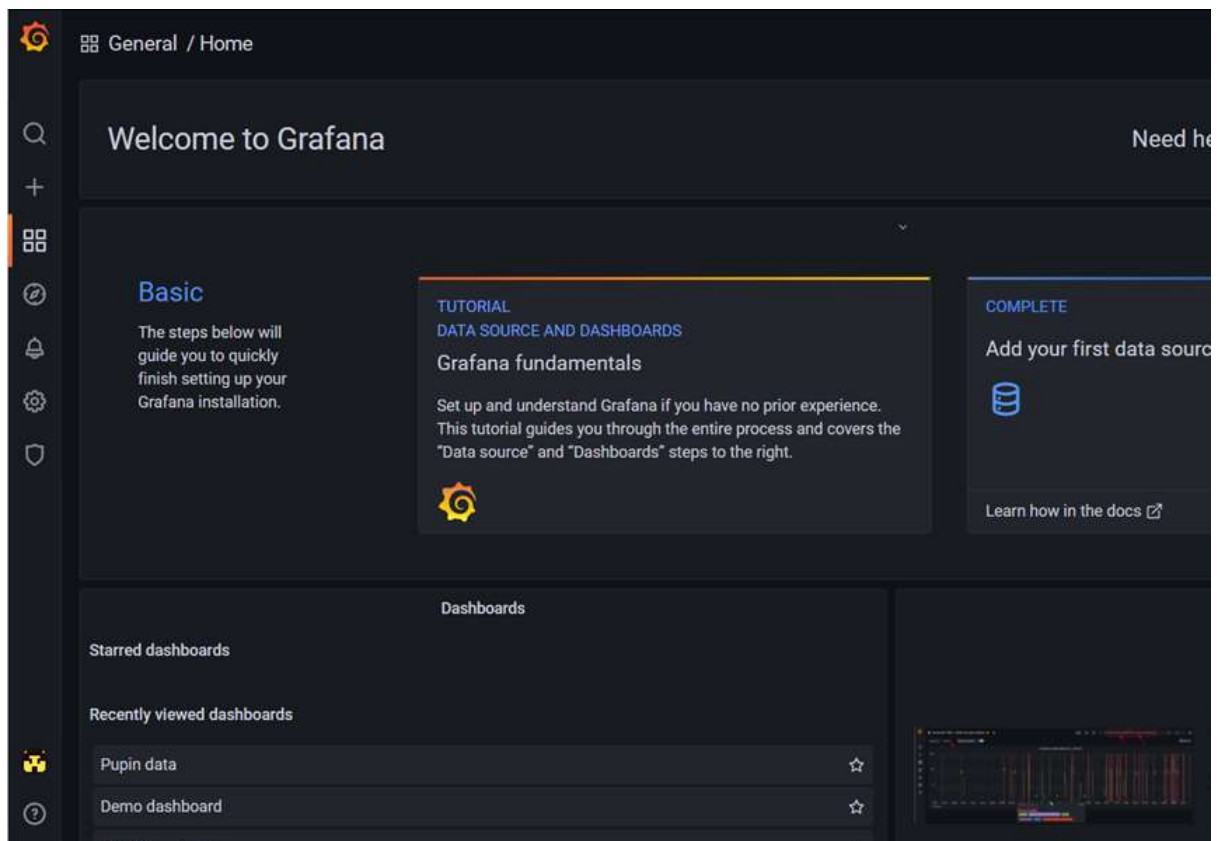


Figure 8: Grafana main screen



Figure 9: Grafana dashboard

Grafana will be installed in all the pilot sites, so that project participants-service developers can easily visualize the data collected from the pilot plants during service development phase.

## 2.3 Orchestration of system components

Orchestration for information systems can be defined<sup>1</sup> as: “Orchestration is the automated *configuration*, coordination, and management of *computer systems*, *applications*, and *software*. Orchestration is used to help streamlining and simplify operation management for information technology (IT) personnel.”

For the purposes of AI-PROFICIENT we define:

*Runtime orchestration* of system components as the (at least partially) automated configuration, coordination and management of data measurements/measurement points (ObservableProperty), measurement results (Observation), sensors (Sensor) and derived results as the outputs of edge or centralized services (Procedure), the items in parentheses being elements of the SSN-SOSA canonical data model (see Section 2.4).

*Design time orchestration* of system components as the (not necessarily automated) off line specification addition removal and/or modification of data measurements/measurement points, measurement results, sensors and derived results.

Runtime orchestration depends on being able to:

- Identify the true availability of sensors. This information is extracted from design time orchestration information and the status of the last executed instance of each measurement point (measured/failed to measure). For vision measurements this information is obtained from the INOS Station which is the runtime component of the INOS software suite<sup>2</sup>. This is a soft real time system that is triggered from a hard real time controller in the production cell, usually the PLC (or alternatively the robot controller if a robot is involved). For non-vision measurements, the information is extracted from the measurement database entries, inserted by the PLCs (CONTI) and DCSs (INEOS).
- Identify and manage the currently active services as well as start, stop and modify settings/parameters (including complete models) of services running on the centralized servers (virtual machines in the factory or external clouds). This can be realized using off-the-shelf orchestration tools such as *Kubernetes* applied to *containerized* services.
- Identify the availability (running/not running status) of INOS Station instances and PLC attached sensors. For INOS Station and complex/smart sensors we envision a simple lifebit (heartbeat) mechanism to be sufficient. For simple sensors, the PLC or DCS status of the sensor will be used if available.
- Identify the availability of databases (relational, time series). This can be implemented via direct querying. Appropriate error handling / restart provisions can be made if necessary.

Design time orchestration depends on being able to:

- Identify the set of available measurement points and edge generated derived measurement points. This is supported by being able to export the currently configured measurements in a format that can be mapped to a subset of the CDM. Currently the names, value types and format of the measurements that have been configured on an INOS station is exportable as a .csv file. We will consider automated support for converting it to a CDM representation.

---

<sup>1</sup> <https://www.webopedia.com/definitions/orchestration/>

<sup>2</sup> Manuals of the software: Station User Manual.pdf (In line component), Maestro User Manual.pdf (Off line configuration) and VisualizationUserManualDesktop.pdf (Visualization) are available upon demand and for internal project use in [https://univlorraine.sharepoint.com/:s/Al-PROFICIENTWP5/EqJL1Guws5VMongkdKr8FfoBzUv89FYS1I\\_tXw8lpNWOnQ?e=iefBXH](https://univlorraine.sharepoint.com/:s/Al-PROFICIENTWP5/EqJL1Guws5VMongkdKr8FfoBzUv89FYS1I_tXw8lpNWOnQ?e=iefBXH)

- Identify the lower-level sensors attached to an INOS Station (composite, smart sensor). This is supported by being able to export the complete hardware setup as an XML file. We will consider automated support for extracting the sensor names and types to a CDM representation.
- Identify any AI models running at the edge on the INOS Station platform. AI model inference in INOS Station is implemented via Tensorflow integration. New capability must be added to export the model into a format that is appropriate for use by AI-PROFICIENT design time model management tools. In this case design time includes periodic updating of edge models from centralized model training.

## 2.4 Canonical data model

The canonical data model (CDM) aims to cover the data exchange between software components and the middleware in a generic manner. It is based on the Semantic Sensor Network (SSN), more specifically the SOSA (Sensor, Observation, Sample and Actuator) ontology [8]. This lightweight ontology makes it possible to model interactions between devices and entities performing sensing and actuation acts, covering a wide range of use cases and possibilities.

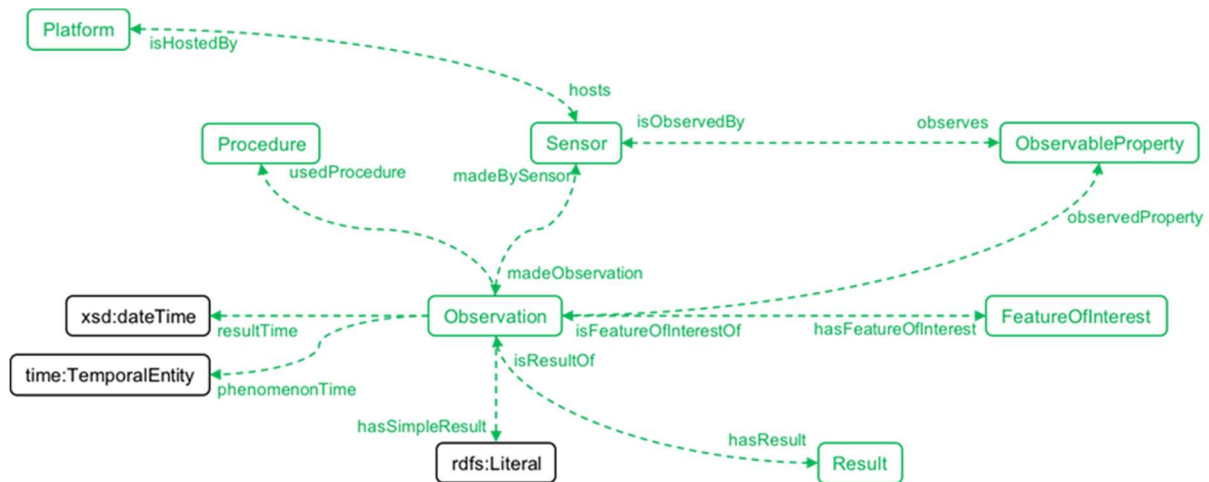


Figure 10: Observation classes in SOSA

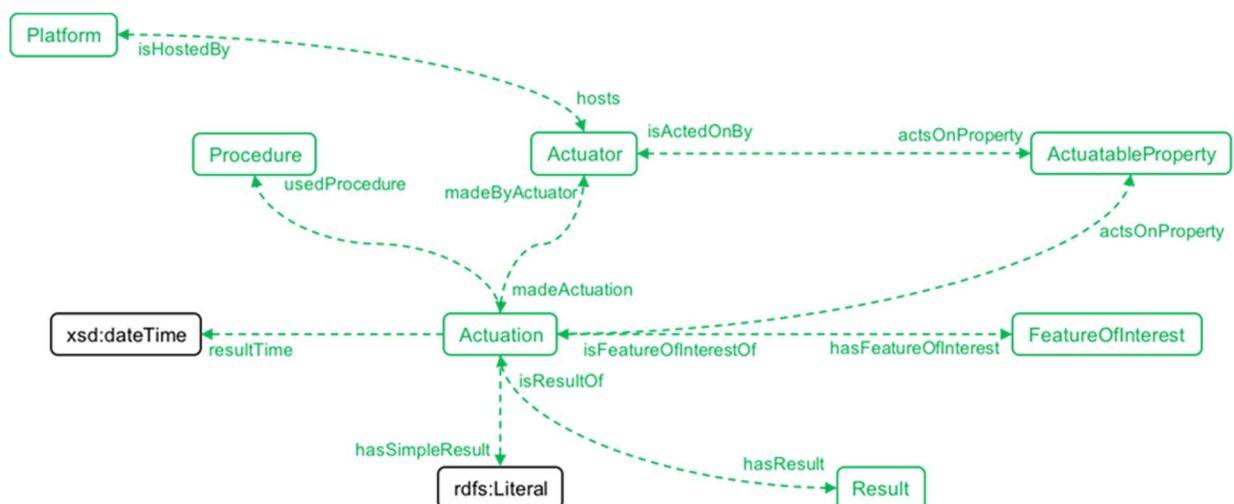


Figure 11: Actuation classes in SOSA

The current version of the CDM is aimed to data exchanges mainly and has left out entities from the SOSA like Procedures and Features of Interest. These exchange events are depicted as Observations (made by sensors) and Actuators (oriented to actuator).



Figure 12: CDM classes

The CDM entities just define the necessary attributes to cover the data exchanges, while the actual implementation can differ depending on the use case: Data could be formatted in with different structured data styles, such as JSON or XML, and different communication protocols such as MQTT or HTTP/REST could be used to deliver them.

## Observation

Observations depict values read by a sensor at a given time. Different properties can be monitored by the same sensor.

Table 1: Fields of Observation object

Field name	Field type	Additional description
resultTime	integer	UNIX time in milliseconds (UTC). Represents when the observation message was created.
phenomenonTime	integer	(Optional) UNIX time in milliseconds (UTC). Represents the specific time of the observed value.
observedProperty	string	Identifier of the monitored variable type.
result	String/Boolean/numeric	Value of the observed event
madeBySensor	string	Identifier of the sensor entity that produced the observation

E.g.: This message tells the current volume output for extruder EXT-001-B

```

{
  "resultTime": 1542899607000,
  "phenomenonTime": 1542899607000,
  "result": 75,
  "madeBySensor": "EXT-001-B",
  "observedProperty": "Extruder_volume"
}

```

## Actuation

Similar to observation, but these messages are aimed as orders to devices capable of interacting with the real world, namely actuators.

Table 2: Fields of Actuation object

Field name	Field type	Additional description
resultTime	integer	UNIX time in milliseconds (UTC). Represents when the actuation order was created.
actsOnProperty	string	Identifier of the configurable var type.
result	String/Boolean/numeric	Value of the configured parameter
madeByActuator	string	Identifier of the actuator entity the message is oriented to

E.g.: This message tells the actuator EXT-001-RS to select extruder recipe R\_122\_CV

```
{
  "resultTime": 1542899607000,
  "result": "R_122_CV",
  "madeByActuator": "EXT-001-RS",
  "actsOnProperty": "Extruder_recipe"
}
```

## Sensor/Actuator

Sensors and actuators can be defined by their id and the list of properties they can observe or act on. They are not limited to equipment and hardware devices: A web service with meteorological data could be used to send observations of predicted weather data. Actuation orders could be oriented to specific human operators.

Table 3: Fields of Sensor object

Field name	Field type	Additional description
id	string	Unique identifier of the sensor
observes	String array	Identifier of the monitored var type.
isHostedBy	string	Identifier of the equipment where the sensor is installed (optional)

Table 4: Fields of Actuator object

Field name	Field type	Additional description
id:	string	Unique identifier of the actuator
actsOnProperty	String array	Identifier of the configurable var type.
isHostedBy	string	Identifier of the equipment where the actuator is installed (optional)

E.g.: Messages depicting a sensor and an actuator on Extruder EXT-001. The sensor has three monitored signals, and the actuator has two properties than can be interacted with.

```
{
  "id": "EXT-001-B",
  "actsOnProperty": ["Extruder_volume", "Extruder_temp", "Extruder_voltage"],
  "isHostedBy": "EXT-001"
}
{
  "id": "EXT-001-RS",
  "actsOnProperty": ["Extruder_recipe", "Extruder_output_on_off"],
  "isHostedBy": "EXT-001"
}
```

### Observable/Actuable Properties

Both observations and actuations use an identifier to refer to the properties involved. These properties can be:

Table 5: Fields of Observable/Actuable Properties

Field name	Field type	Additional description
id	string	Unique identifier of the property
description	string	Description of the property
unit	string	(Optional) Unit of the property. Some properties may not have a specific measuring unit.

An example of the message is given as follows:

```
{
  "id": "Extruder_temp",
  "description": "Temperature of the extruder nozzle",
  "unit": "°C"
}
```

### 3 Interface to smart components and edge AI

#### 3.1 Industrial Internet of Things (IIoT) interoperability

Interoperability is the ability of a system to work with or use the parts of equipment of another system [9]. Most of the interoperability frameworks adopt a layered interoperability model [10] which is shown in Figure 13. As can be seen, it encompasses technical, syntactic and semantic interoperability layers. The aim of the technical interoperability layer is to establish the communication channels between the systems using different communication protocols. Once this is ensured, there is a need to define a common data format for such messages, which is the aim of the syntactic interoperability layer. The semantic interoperability layer provides the meaning of data. In the sequel, we provide more details on each of the aforementioned interoperability layers:

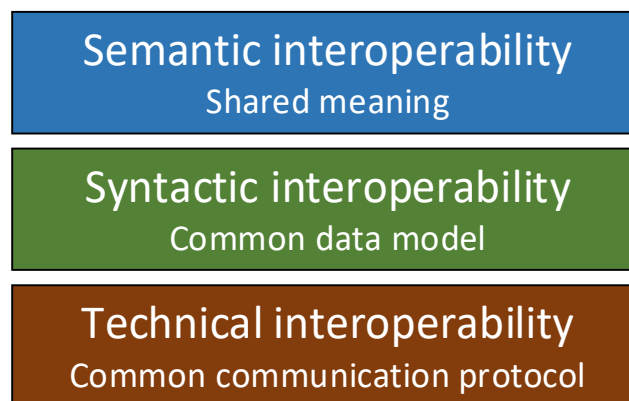


Figure 13: Interoperability layers [10]

##### 3.1.1 Technical interoperability

Whereas technical interoperability is not enough per se, it is necessary to ensure it before other interoperability layers. Technical interoperability includes the software and hardware components that allow the communication within e.g., IIoT network. In particular, this interoperability layer is aimed at the communication protocols and required infrastructure which enables these protocols to operate. Therefore, in order to provide technical interoperability for plant devices and machinery, it is necessary to employ appropriate hardware solutions that enable communication with the rest of the system.

##### 3.1.2 Syntactic interoperability

Syntactic interoperability relates to the common messaging format which forms the basis for messages exchanged among distinct parts of system. This interoperability layer ensures that two or more devices are able to understand the message content of the exchanged data. In practice, there exist different methods which can be used to implement syntactic interoperability. One feasible way is to enable direct communication between the devices by using their native protocols, or by using protocol converters as it is shown in Figure 14a. Another possibility is to employ the protocol converters that unify different protocols of legacy and newly installed equipment by using CDM, as it is shown in Figure 14b. The advantage of the aforementioned approach is that each protocol has to be translated only once into common format and backwards, that requires a linearly growing number of adapters, i.e.,  $2N$ . This is the reason why CDM (described in section 2.4) approach has been adopted in AI-PROFICIENT project.

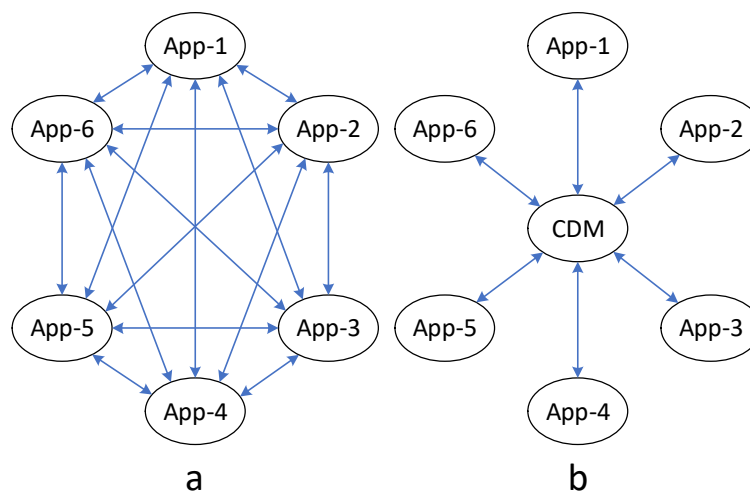


Figure 14: (a) Direct protocol conversion, (b) Canonical data model

### 3.1.3 Semantic interoperability

The semantic interoperability (objective of Task 5.2) aims to ensure that the exact meaning of information exchanged among different parts of the systems is understandable by applications that were not developed for such a purpose. In such a way, systems is capable of combining the information received with other information resources, and additionally, process it in a meaningful way. By using semantic interoperability there exist several benefits:

- The quality of an interoperability specification is enhanced because a systematic process is applied for defining it.
- Interpretation problems can be solved using the resulting specifications as a reference.
- The potential maintenance and extension tasks of the specification are performed more easily.

When aiming to achieve the semantic interoperability, instead of a textual one, a formalized specification is desirable, since it can be used by further tools for validation purposes, and it is easier to make modifications. Among the possible approaches, using ontologies (or Semantic Web vocabularies) is a well adopted approach to formalize specifications. Within AI-PROFICIENT platform, ontologies will be used to express and store the knowledge of different parts of the system, which will be later used by AI services.

The interaction between the layers is a critical challenge to ensure that one layer will not block another. To facilitate the communication between all layers a single representation language for the data models used in them can be used. That allows to see the impact and relationship between them. A representation language suited for this is the semantic web using RDF as syntax.

Using that language, the data models can be expressed and interrelated. Starting from the international data standards for Sensors & Observations. These provide the generic language (vocabulary) to express the information in AI-PROFICIENT. Using these, an AI-PROFICIENT generic core application profile is created. This core application profile expresses the generic semantics and constraints that hold for the whole project. The core application profile is then specialized to use case specific application profiles. Any use case specific application profile can be further specialized into implementation models. An implementation model is a data model that reflects the data in the form as it being used in the implementation. In AI-PROFICIENT, the exploration to find the most appropriate implementation for a use case objective could lead to parallel equivalent implementations using distinct, yet closely related, implementation models. For instance, one could explore the impact from one database technology over another, requiring to use different syntactical representations for the same

data (For instance, one technology can require that the name of a property must be starting with a capital letter).

To illustrate the approach, we list in the following table existing ontologies/standards and application profiles and implementation models to be created:

Type of specification	Name of the specification	Comments
Vocabulary	<a href="#">ISO 19156 Observations and Measurements</a>	International standard
Vocabulary	<a href="#">OGC/W3C Semantic Sensor Network Ontology</a> .	International standard
Application Profile	<a href="#">PURL.eu Core model</a>	A generic application profile integrating previously mentioned vocabularies
Application Profile	<a href="https://data.ai-proficient.eu/doc/applicationprofile/core">https://data.ai-proficient.eu/doc/applicationprofile/core</a>	Core Application Profile AI-PROFICIENT Core Model [To be created]
Application Profile	<a href="https://data.ai-proficient.eu/doc/applicationprofile/ineos-uc10">https://data.ai-proficient.eu/doc/applicationprofile/ineos-uc10</a>	Ineos UC-10 Application Profile [To be created]
Implementation Model	<a href="https://data.ai-proficient.eu/doc/implementationmodel/ineos-uc10">https://data.ai-proficient.eu/doc/implementationmodel/ineos-uc10</a>	Ineos UC-10 implementation model [To be created]

The next table illustrates the notion of the term Concept in the above specifications:

Specification	Concept	Reference
<a href="#">ISO 19156 Observations and Measurements</a>	Observation	<a href="http://def.isotc211.org/iso19156/2011/Observation#OM_Observation">http://def.isotc211.org/iso19156/2011/Observation#OM_Observation</a>
<a href="#">OGC/W3C Semantic Sensor Network Ontology</a> .	Observation	<a href="http://www.w3.org/ns/sosa/Observation">http://www.w3.org/ns/sosa/Observation</a>
<a href="#">PURL.eu Core model</a>	Observation	<a href="https://purl.eu/doc/applicationprofile/AirAndWater/Core/kandidaatstandaard/2021-10-01/index_en.html#Observation">https://purl.eu/doc/applicationprofile/AirAndWater/Core/kandidaatstandaard/2021-10-01/index_en.html#Observation</a>

AI-PROFICIENT Core model	Observation	TBD
--------------------------	-------------	-----

Then approach is as follows. During workshops, the partners discuss whether the term *Observation* in AI-PROFICIENT Core Model is the same as specified in the international data standards. Suppose they agree, then this term is adopted in the ontology.

In the next step, the individual properties are assessed. For instance, the OGC/W3C Semantic Sensor Network Ontology<sup>3</sup> states that an Observation must have a property `resultTime` (and at most 1 value). The PURL.eu model has followed that. During the first discussion for creating the AI-PROFICIENT Core Model, the partners agreed to follow this and include it in the Core Model.

According to the SSN/SOSA specification, the range of this property is `xsd:dateTime`. In the implementation model proposal in section 2.4 the `resultTime` is expressed as UNIX time in milliseconds (UTC). Although a technical conversion is often possible, such low-level differences may hinder the reuse by others. This is the role of the implementation models. They lift the design decisions of the used underlying technology to the same level as the semantic data models. Suppose one finds a software product that implements SSN/SODA, one knows that there must be a conversion module be deployed to turn UNIX time into `xsd:dateTime`.

Observe that detecting this is part of the maintenance for standards as SSN/SOSA. If the practice turns out that this is a too restrictive decision, an issue can be filed to motivate a change based on an implementation experience. Transparency about the decisions connecting international agreements with implementations is the key to address the interoperability challenge.

To make this transparency also visible, the usage of annotating standards like JSON-LD is important. Then also the actual data streams are connected with the data models.

So instead of a plain old json structure like this:

```
{
  "resultTime": 1542899607000,
  "phenomenonTime": 1542899607000,
  "result": 75,
  "madeBySensor": "EXT-001-B",
  "varId": "Extruder_volume"
}
```

Augmenting the json with a ld context

```
{
  "@context": {
    "resultTime": "http://www.w3.org/ns/sosa/resultTime",
  },
  "resultTime": 1542899607000,
  "phenomenonTime": 1542899607000,
  "result": 75,
  "madeBySensor": "EXT-001-B",
  "varId": "Extruder_volume"
}
```

connects the json field with the semantics.

---

<sup>3</sup> <https://www.w3.org/TR/vocab-ssn/>

Observe that this does not lead to the semantics according to the implementation model. This can be achieved by publishing the implementation model as a semantic web document.

```
{
  "@context": {
    "resultTime": "https://data.ai-
    proficient.eu/doc/implementationmodel/uc10#resulttime",
  },
  "resultTime": 1542899607000,
  "phenomenonTime": 1542899607000,
  "result": 75,
  "madeBySensor": "EXT-001-B",
  "varId": "Extruder_volume"
}
```

In this way, the data is always hooked up with the semantics.

Using the above approach and supporting technologies the semantic interoperability can play its role in improving data interoperability among systems. The ontologies, data models and interoperability agreements made for AI-PROFICIENT will be reported in the deliverable D5.2.

## 3.2 Communication protocols and integration

To interface with the additional sensors, mainly vision sensors, which are being used in AI-PROFICIENT and to support AI at the edge, we are using a number of standard industrial networks and communication protocols. Besides, we will also provide support for some more as part of our future exploitation strategy and are developing new capability by implementing MQTT support for smart vision sensor systems built using the reference INOS Station platform.

The INOS Station is a soft real time system specialized for vision and robot guidance applications that has been running in production lines, mainly in the automotive industry, over quite a number of years. The current releases (21.x.x major release) support the protocols below and as part of AI\_PROFICIENT we have added support for the MQTT protocol.

The INOS Station is extensively configurable with a hybrid architecture combining conditionally guarded execution (PLC like) and dataflow execution (dependency driven action execution) with the option to execute an ECMA script (JavaScript) as a primitive action.

### 3.2.1 Profinet, Profinet CIFS

Profinet<sup>4</sup> is a common industrial real time network and communications protocol.

It is commonly supported on PCs and PLCs as well as a number of smart sensors and peripherals. It supports hard real time communications, can be used as a safety network<sup>5</sup> and is claimed to be the most commonly installed industrial network. It is certainly the most common standard in the EU.

In AI-PROFICIENT, Profinet will be used in the CONTI pilot use cases 5 and 7 to communicate between the INOS vision and AI at the edge PC and the plant PLCs (programmable logic controllers). These PLCs provide signals to INOS station for performing measurements.

### 3.2.2 MQTT

MQTT<sup>6</sup> is a standard for IOT messaging. It is used extensively for IOT applications and in the last four years or so has made significant inroads as an enabling standard for the Industrial Internet of Things

---

<sup>4</sup> <https://www.profibus.com/technology/profinet/overview>

<sup>5</sup> <https://www.profibus.com/technology/profisafe/>

<sup>6</sup> <https://mqtt.org/>

(IIOT) and Industry 4.0. As part of AI-PROFICIENT, we have incorporated support for MQTT in the INOS Station platform. We have integrated a library capable of supporting the MQTT 5.0 standard but for simplicity have exported the MQTT 3.1.1 subset of functionality to the system integrator user interface.

We consider MQTT integration critical for the exploitation of the integrated AI-PROFICIENT system. In AI-PROFICIENT we are initially not using MQTT for information (measurement) transfer but can readily do so and we plan to revisit as the project progresses. We will use, most probably, MQTT as an interface to HMIs in the packing and unpacking stations of CONTI Use Case 7.

### 3.2.3 OPC DA (Classic)

INOS station supports the OPC DA<sup>7</sup> protocol and can be used in future exploitation but was not needed as part of the AI-PROFICIENT demo cases. It is a higher-level protocol running over Ethernet or industrial real time networks such as Profinet and EthernetIP.

### 3.2.4 EthernetIP

INOS station supports EthernetIP<sup>8</sup> and can be used in future exploitation but it is not needed as part of the AI-PROFICIENT demo cases. This is the second most common industrial real time network and is very common in North America. Development is managed by ODVA<sup>9</sup>.

### 3.2.5 DeviceNet

INOS station supports the DeviceNet<sup>10</sup> protocol and can be used in future exploitation but it is not needed as part of the AI-PROFICIENT demo cases. A relatively older protocol but with a significant installed base. Currently managed jointly with EthernetIP by ODVA it is interoperable with it as part of the Common Industrial Protocol standard (CIP<sup>11</sup>).

### 3.2.6 INOS Standard Ethernet Socket

This is an internal INOS standard built over standard sockets. While not a real time network, Ethernet and especially Gigabit Ethernet (usually implemented as an IEEE 802.3ab over twisted pair)<sup>12</sup> is extensively used in industry as an inexpensive, fast, high-capacity network (e.g., for image transfer). INOS supports both simple buffer mapping (similar to the real time protocols such as Profinet), XML messaging and customized scriptable output generation to text (e.g., JSON, MODBUS) or binary formats. In AI-PROFICIENT, this protocol will be used to communicate with a number of GiGE<sup>13</sup> sensors used in CONTI use cases and as an alternative to Profinet.

### 3.2.7 Profibus, Modbus (via scripting interface)

INOS station supports Profibus<sup>14</sup> and Modbus<sup>15</sup> via a scripting interface and can be used in future exploitation but they are not currently needed as part of the AI-PROFICIENT demo cases.

---

<sup>7</sup> <https://opcfoundation.org/about/opc-technologies/opc-classic/>

<sup>8</sup> <https://www.odva.org/technology-standards/key-technologies/ethernet-ip/>

<sup>9</sup> <https://www.odva.org/>

<sup>10</sup> <https://www.odva.org/technology-standards/key-technologies/devicenet/>

<sup>11</sup> <https://www.odva.org/technology-standards/key-technologies/common-industrial-protocol-cip/>

<sup>12</sup> [IEEE SA - IEEE 802.3ab-1999](https://www.ieee.org/standards/publications/standards-summaries/ieee-802-3ab-1999)

<sup>13</sup> <https://www.automate.org/a3-content/vision-standards-gige-vision>

<sup>14</sup> <https://www.profibus.com/>

<sup>15</sup> <https://modbus.org/>

### 3.2.8 Database triggering via polling

Because of factory data security policies in the industrial demo user plants (CONTI and INEOS) and the fact that most of the already data being used are exported to existing database, database mirroring, exporting and remapping are expected to be used as part of the AI-PROFICIENT data management strategy at both CONTI and INEOS demo plants. INOS Station is capable of using periodic polling to check the state or relational database as an input dependency to edge computations.

## 4 IT infrastructure at pilot sites to host AI-PROFICIENT platform

### 4.1 Continental

#### 4.1.1 Virtual Machine

As presented in deliverable 1.5, the AI platform will be based on a virtual machine (VM) within Continental, due to strict security measures. Before the virtual machine can be accessed, each partner will have to fulfil a form to apply for a Business Partner Access (BPA). Once the BPA request has been made and approved, each partner will receive an email with their login details and passwords as well as a link to download all the necessary software to connect to the VM.

- Firstly, there is the VPN: Cisco AnyConnect Client.
- Secondly, there is the logger to access the VM: CudaLaunch.
- Finally, there are the application(s) to take remote control: Remote Desktop, LanDESK, UltraVNC etc.

Once all the software's have been downloaded and the virtual machine has been set up, the partner will first be able to establish the connection between the public network and the Continental network using the Cisco Client. To do so, he will have to use the login and password linked to the application.

Once the VPN connection has been established, the connection between the Continental network and the factory network must be established. To do this, the partner will need to use the CudaLaunch application and connect to it with the username and password linked to the application.

At this stage, the partners will be able to access to the VM thanks to the remote viewer application(s).

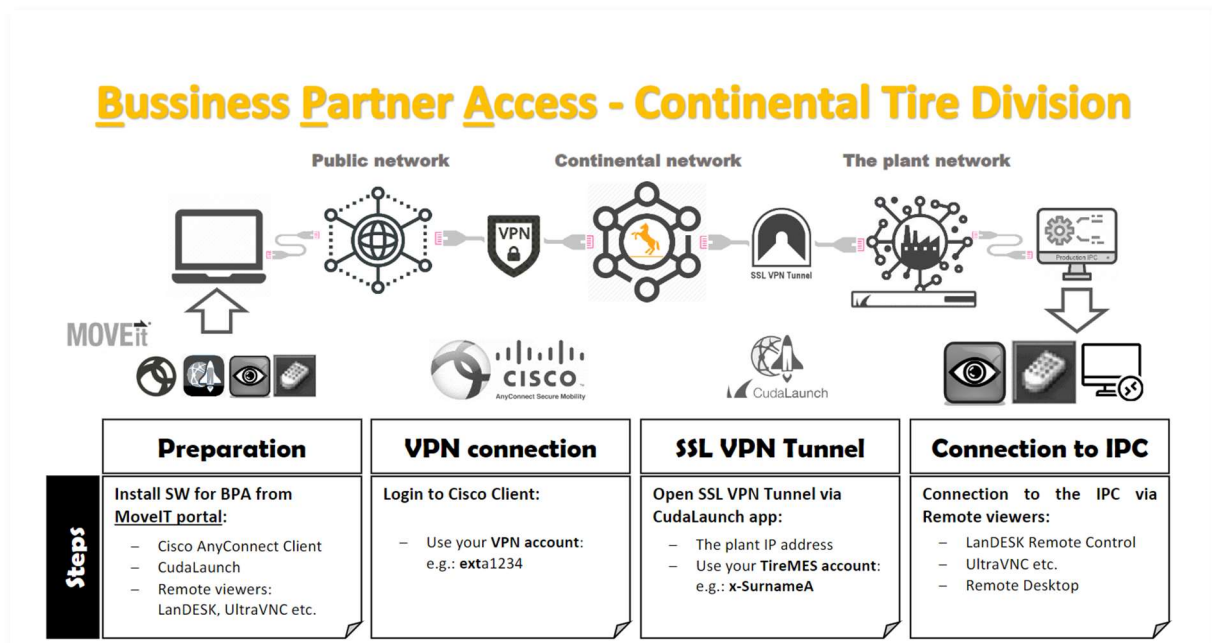


Figure 15: Presentation of the steps to connect to the Virtual Machine at Sarreguemines plant

Concerning the different remote connection protocols, we can propose two:

- The Remote Desktop Protocol:

Thanks to this protocol, partners will be able to connect to the VM with totally independent sessions (see Figure 16). Please note that this is not a duplication of screens. Nevertheless, only 2 partners will be able to connect simultaneously with this protocol.

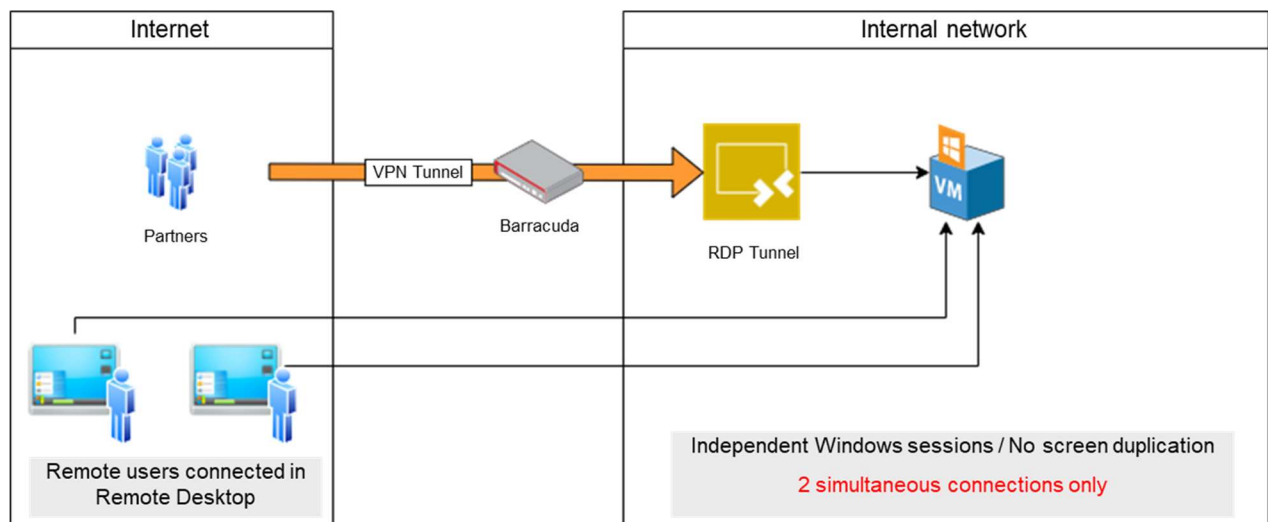


Figure 16: Description of the Remote Desktop Protocol

- The Virtual Network Computing Protocol:

With this protocol, unlike the RDP protocol, the number of simultaneous connections is not limited (see Figure 17). However, the sessions are not independent. The screen is only duplicated for each connected partner and the mouse and keyboard are shared. It should also be noted that this connection protocol will disappear from Continental's connection methods in the future.

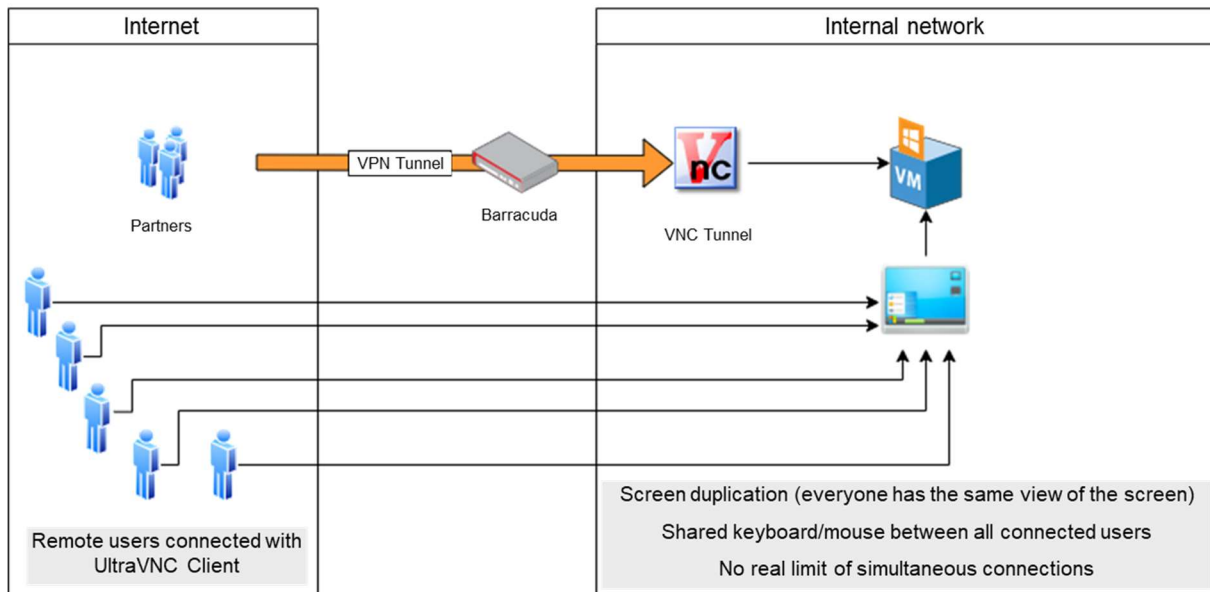


Figure 17: Description of the Virtual Network Computing Protocol

Through the proposed connection methods, it is also possible to perform file transfers from partners directly to the virtual machine. In addition, remote access can be made via the Internet, which can be useful for software's using web management interfaces.

The software to be installed for the partners on the virtual machine has also been identified and is as follows:

- Docker: via this application, all partners will be able to download the applications they need to work on the virtual machine.
- Database software. To date, it is not yet decided what type of database will be used (relational or timeseries). Nevertheless, if the duplication of the databases on the virtual machine keeps the same format, the database will be in SQL format.
- PyTorch, software requested by the Université de Lorraine.

## 4.2 INEOS Geel

INEOS Geel site allows access to OT (Operational Technology) in a very strict manner in order to meet high cyber security standards. Figure 18 describes the process. All external users first have to request an account for the office network. The accompanying hardware key generator can be kept at Ineos or sent to the user individually (depending on nr of external users). When this account is established, a VPN (Virtual Private Network) access named InLink is requested. The next step is logging in to the Fudo-PAM portal (Privileged Access Management server). After logging into this portal (password needed), the user can connect to the boxes on the PIN (Process Information Network) that are assigned to his profile by means of RDP/VNC (Remote Desktop Protocol or Virtual Network Computing). Here a request for access to the OT network end device server (AI Proficient Virtual Machine hosted in INEOS plant) can be triggered. The Ineos application owner can then grant access for this request via the same portal. (max 1 day). An RDP file is then created and can be downloaded and started by the customer. Encrypted credentials are then used to logon to the target system. If needed an in-between jump server can also be used.

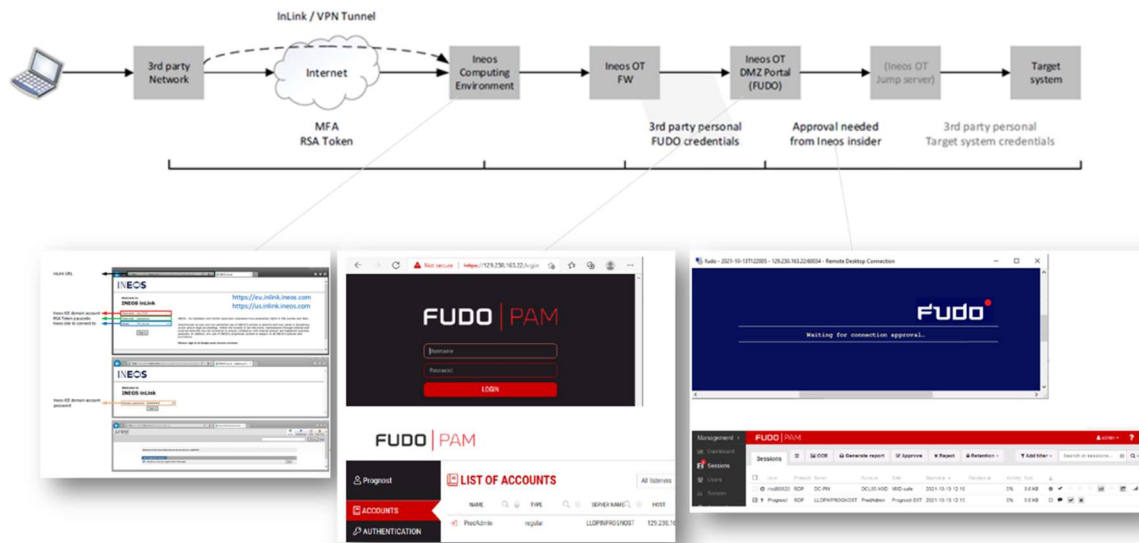


Figure 18: Remote access protocol Ineos Geel site

During the external access session, Ineos personnel is able to follow the session and, moreover all sessions are recorded and can be watched afterwards if needed.

### 4.3 INEOS Cologne

The INEOS Cologne site allows access to OT (Operational Technology) systems similar to the other pilot plants as described above. Figure 19 describes the specific Ineos Cologne process. All external users have to request an account for the office network. When this account is established, Ineos requests a VPN (Virtual Private Network) access, named inLink. The individual inLink profile allows access via RDP (Remote Desktop Protocol) to a Windows Terminal server in the office network (Level 4). From there users sign into a Guacamole Proxy (Apache Foundation) via https, where individual connections via SSH (Secure Shell), VNC (Virtual Network Computing) or RDP to hosts in our PIN (Process Information Network) (Level 3.0), in this case the AI Proficient VM, are defined. For https connections to Level 3.0 a reverse proxy is used.

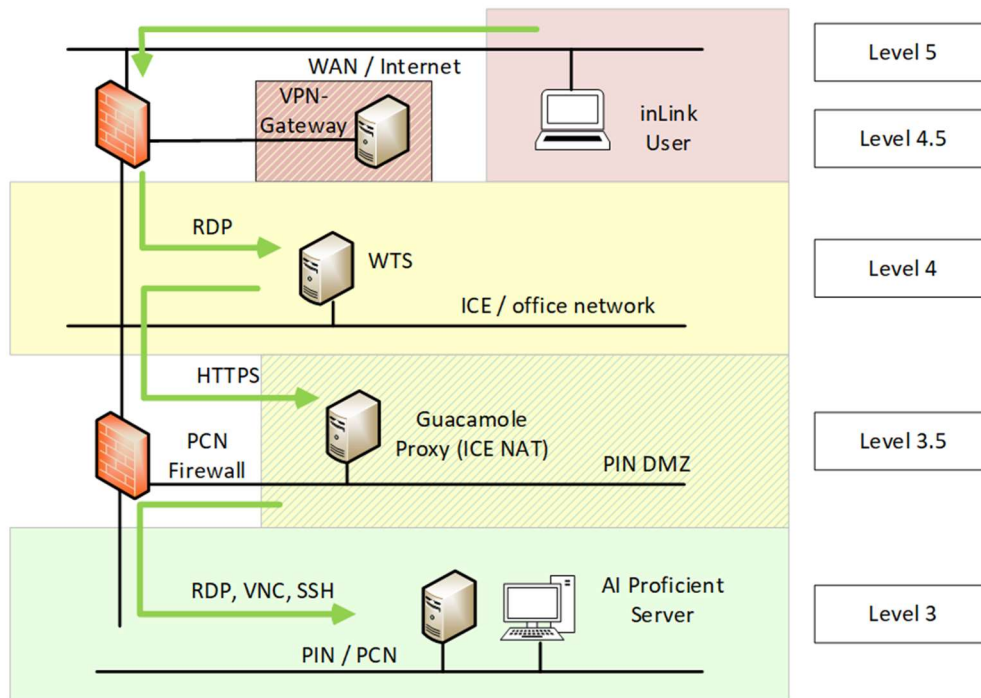


Figure 19: Remote access protocol Ineos Cologne site

Abbreviations used in Figure 19:

WTS: Windows Terminal Server

DMZ: Demilitarized Zone

NAT: Network Address Translation

PCN: Process Control Network

Level 3 to 5 as per industry wide 'Purdue Enterprise Reference Architecture' [11].

## 5 Conclusions

In this document, the aim is to summarize the work performed within Task 5.1 in relation to design and specification of AI-PROFICIENT communication middleware and IIoT interoperability. This document has taken as the input the work performed previously in Task 1.5 on system architecture which was provided in Deliverable 1.5.

First, we have revisited the system architecture provided in D1.5 and provided more details on the envisioned system middleware and its main components: message broker, data layer, orchestration of system components and canonical data model. Next, we described the interfaces to smart components and edge AI. More specifically we describe the IIoT interoperability, different communication protocols and semantic ontology. Finally, we described how middleware will be implemented at pilot sites, having in mind the security constraints imposed by the plant's internal security procedures.

## 6 Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957391.

## 7 References

- [1] "InfluxDB," [Online]. Available: <https://www.influxdata.com/>.
- [2] "RDF," [Online]. Available: <https://www.w3.org/TR/rdf11-concepts/>.
- [3] "JSON-LD," [Online]. Available: <https://json-ld.org/>.
- [4] "SHACL," [Online]. Available: <https://www.w3.org/TR/shacl/>.
- [5] "Virtuoso Universal Server," [Online]. Available: <https://virtuoso.openlinksw.com/>.
- [6] "SPARQL," [Online]. Available: <https://www.w3.org/TR/sparql11-query/>.
- [7] "Grafana," [Online]. Available: <https://grafana.com/>.
- [8] K. Janowicz, A. Haller, S. Cox, D. Phuoc and M. Lefrançois, SOSA: A lightweight ontology for sensors, observations, samples, and actuators, *Journal of Web Semantics*, 2018.06.003..
- [9] "Merriam Webster - Interoperability," [Online]. Available: <https://www.merriam-webster.com/dictionary/interoperability>.
- [10] H. Kubicek, R. Cimander and H. J. Scholl, " Interoperability in Government," in *Organizational Interoperability in E-Government*, Springer, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-642-22502-4\\_2](https://doi.org/10.1007/978-3-642-22502-4_2), 2011.
- [11] "Purdue model," [Online]. Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.194.6112&rep=rep1&type=pdf>.