Improved Robotic Platform to perform Maintenance and Upgrading Roadworks: The HERON Approach

Grant Agreement Number: 955356

D6.2: Middleware and DF services

Work package	WP6: Communication and Networking Solutions, DSS, IMS and CoP
Activity	Task 6.2 HERON Middleware and Data Fusion Services
Deliverable	D6.2: Middleware and DF services (Other)
Authors	Dimitrios Bilionis, Theodora Karali, Günter Becker, Leonidas Camarinopoulos, Inke Hussels (RISA)
Status	Final (F)
Version	1.0
Dissemination Level	Public (PU)
Document date	29/06/2024
Delivery due date	31/03/2024
Actual delivery date	29/06/2024
Internal Reviewers	Nikos Frangakis (IKH), Nikos Bakalos (ICCS)
	This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 955356.

Document Control Sheet

Version history table			
Version	**Date**	**Modification reason**	**Modifier**
0.1	08/01/2024	Initial Table of Contents and initial content	RISA
0.2	21/02/2024	Input to several Chapters	RISA
0.3	31/02/2024	Input to several Chapters	RISA
0.4	14/03/2024	Input to several Chapters	RISA
0.5	24/03/2024	Input to several Chapters	RISA
0.6	26/03/2024	Complete draft submitted for review	RISA
1.0	29/06/2024	Complete (after review)	RISA

Legal Disclaimer

Table of Contents

List of Tables

List of Figures

Abbreviation List

Abbreviation	Definition
API	Application Programming Interface
AI	Artificial Intelligence
AR	Augmented Reality
CoP	Common Operational Picture
CUD	Chaussée Urbaine Démontable
Dx.x	Deliverable x.x
DF	Data Fusion
DMP	Data Management Plan
DRACO	Data Routing & Contextualization
DSS	Decision Support System
EM	Event Management
EP	Evolutionary Prototyping
GA	Grant Agreement
IMS	Incident Management System
IoT	Internet of Things
IS	Information System
RI	Road Infrastructures
Tx.x	Task x.x
UAV	Unmanned Aerial Vehicle
V2I/X	Vehicle-to-Infrastructure/-Everything
WPx	Work Package x

Executive Summary

This deliverable is of type other and is an accompanying report to the actual software developed within WP6: Communication and Networking Solutions, DSS, IMS and CoP of HERON project under Grant Agreement No. 955356. This deliverable offers an in-depth overview of the middleware serving as the core system for data acquisition, storage, and management within the HERON project. Operating on a scalable architecture, the middleware encompasses:

- A high-performance communication API and messaging services tailored to streamline data exchange with other components, accommodating both synchronous and asynchronous modes.
- Efficient coordination of information delivery between modules and services, ensuring seamless and scalable service assurance.
- Serving as a broker across network communication interfaces, ensuring that each connected service and module presents its information in the appropriate format for the HERON platform or other information systems.
- Incorporating capabilities for processing and fusion to enhance data integration and analysis within the system.

The Deliverable is the outcome of Task 6.2 and has been authored with contribution from all technical partners of the HERON project.

1 Introduction

1.1 Purpose of the Document

This document is the outcome of Task 6.2 that is responsible for the development of HERON Middleware and DF services. Due to the variation and quantity of components necessary for the HERON middleware and DF services to operate, this task is subdivided into the following subtasks:

- Sub-Task 6.2.1 - HERON Middleware Layer 1 - Preprocessing: The subtask involves the development of the necessary services and the integrated gateway capabilities for all those different data sources so as the middleware will be capable of exchanging information with external resources. It will incorporate storage capabilities for the identified data sources.
- Sub-Task 6.2.2 - HERON Middleware Layer 2 - Main-processing: The subtask involves the development of a data processing, storage and mining module designed for temporal data storage in order to be aggregated (information is gathered and expressed in a summary form, for purposes such as statistical analysis) and for data mining process (discovering patterns in the retrieved data sets). In order to achieve better heterogeneous data management and mapping (uniform information formatting along with similar contextual information), an event filtering and contextual information management module will be developed along with the ontologies and semantic representation layer. This layer will also include the development of a CEP module in order to establish rules and reasoning on sensor data that will permit the early detection of anomalies and alerts that cannot be detected by the individual systems.
- Sub-Task 6.2.3 - HERON Middleware APIs: This subtask involves the development of the standardised APIs necessary for other components to interact with the HERON middleware. Through this process, a common interaction layer will be designed, in the form of an API specification, to which modules and data providers should adhere in order for the data exchange to be feasible.
- Sub-Task 6.2.4 - HERON Middleware Semantic Services: This subtask is crucial not only for the interoperability of the different (internal) components of the system; as well as external systems; but also necessary for the completion and correct operation of subtasks 6.2.1 and 6.2.3.

This report's objective is to offer a comprehensive overview of the middleware from a technological standpoint, encompassing crucial architectural design aspects, technical develpoments, and capabilities. The Deliverable is structured as follows:

- Section 1. Describes HERON's aim as well as this document's purpose, intended audience and structure.
- Section 2. Presents an overview of HERON platform and middleware, as well as its development methodology.
- Section 3. Describes the overall middleware and DF services system architecture, as well as its layers' functionalities and tools deployed.
- Section 4. Concludes the Deliverable by summarising the main outcomes and referring to future work.

1.2 Intended Audience

As Deliverable 6.2 is public, it will be openly available to all stakeholders, such as public authorities, infrastructure owners and operators, researchers and technology providers, as well as decision and policy makers interested in a report presenting the design and development of a middleware system. This Deliverable is also of great interest to all HERON partners as it analyses the middleware from a technological standpoint, encompassing crucial architectural design aspects, technical develpoments, and capabilities.

1.3 Interrelations

This deliverable interacts with all other technical WPs, namely WP3-WP5, while it is also based on the outcomes of the Data Management Plan (DMP), as presented in D1.2; the end-user needs and requirements (as described in D2.1); and the architecture specification (as outlined in D2.2).

2 HERON Platform and Middleware / DF services Overview

HERON endeavors to create an integrated automated system for executing road maintenance and enhancement tasks, such as crack sealing, pothole patching, asphalt rejuvenation, autonomous replacement of CUD (Chaussée Urbaine Démontable / Demountable Urban Roadway) elements, and painting markings. It also supports pre- and post-intervention phases, including automated visual inspections and controlled dispensing and removal of traffic cones. The HERON system comprises autonomous ground robotic vehicles, bolstered by autonomous drones for coordinating maintenance works and pre-/post-intervention activities. Various robotic equipment, including sensors and actuators mounted on the primary vehicle, are employed. These vehicles feature a sensing interface connected to both the robotic platform and Road Infrastructures (RI) to enhance monitoring (situational awareness) of structural and functional conditions, as well as RI and markings. A control software links the sensing interface with the actuating robotic equipment. Augmented Reality (AR) visualization tools, combined with Artificial Intelligence (AI) based toolkits, enable maintenance crews to meticulously examine surface defects and markings under survey.

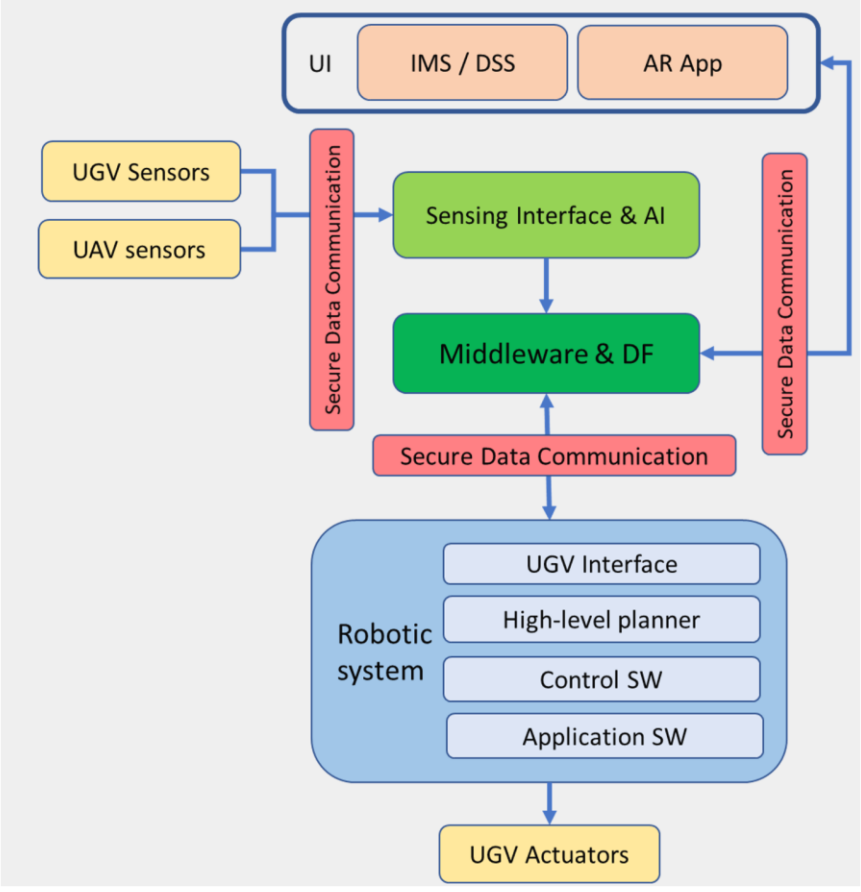

Figure 1. HERON high-level architecture

Furthermore, a middleware is developed to optimally coordinate the road maintenance/upgrading workflows and intelligently process data from sensors and infrastructure datasets to ensure integration and seamless data exchange. All data are integrated

into an enhanced visualization user interface supporting decision-making and communication modules, allowing for Vehicle-to-Infrastructure/-Everything (V2I/X) data exchange for predictive maintenance and enhanced user safety. HERON features a modular design for system operation, maximizing adaptability for various transport infrastructures while reducing fatal accidents, maintenance costs, and traffic disruptions, ultimately increasing network capacity and efficiency.

The middleware works as the fundamental system for acquiring, storing, and managing data within the HERON project. In the following Sections the methodology applied and used for its design and development, as well as its architecture are presented.

2.1 Middleware Development Methodology

Evolutionary Prototyping (EP) stands as a dynamic approach applied in the development of middleware to establish a resilient and adaptable system. The EP process commences with the construction of an initial prototype, emphasizing core functionalities. This prototype serves as a tangible representation, facilitating the exploration and clarification of requirements. A notable advantage of employing EP in middleware development lies in its ability to adapt to evolving needs and changing constraints.

As development advances, new features and requirements seamlessly integrate into the existing prototype. This iterative process empowers developers to refine and enhance the middleware in response to emerging insights and evolving project dynamics. EP fosters a continuous feedback loop, fostering effective communication and collaboration among development teams. Furthermore, EP in middleware development allows for the creation of prototype versions, each building upon insights gained from previous iterations. Stakeholders, including partners and end-users, can test and evaluate these versions, providing valuable input for further refinement.

The evolutionary essence of EP harmonizes well with the complexity typically associated with middleware systems, enabling a more adaptive and responsive development process. This methodology ensures that the middleware system evolves in sync with changing requirements and the technological landscape, ultimately resulting in the creation of a tailored and effective middleware solution.

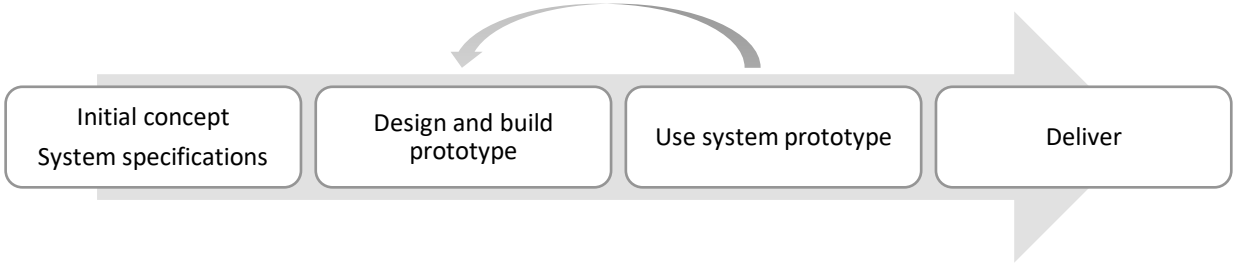

Figure 2. Steps in the development process of HERON middleware

3 Middleware and DF services architecture

HERON middleware and DF services system incorporates information from various HERON components and sensors and interacts with the application layer in an appropriate format while ensuring high availability and scalability. HERON middleware and DF services system leverages on the proven FIWARE platform, to build historic data via a time-series database and to manage (storing and retrieving) historical context information as raw and aggregated time series. Through FIWARE components complex events are managed, as well as the information from various sources are consolidated in common smart data models keeping a unified data scheme.

To meet the needs of HERON project, the Middleware and Data Fusion services architecture consists of five layers, namely Pre-processing layer (employs a KAFKA broker, FTP server and DRACO (Data Routing & Contextualization) component to retrieve data from several internal and external sources); Data Storage layer (employs MongoDB and PostgreSQL to support storage sevices); Interaction layer (build on Core API, Subscription API, Temporal API and Orion Context Broker (NGSI API) to enhance interoperability); Processing / Ontologies and Semantic Representation (uses several processing tools); and Security layer (based on Keyrock and Wilma PEP proxy), as they will be analysed in the following Sections.

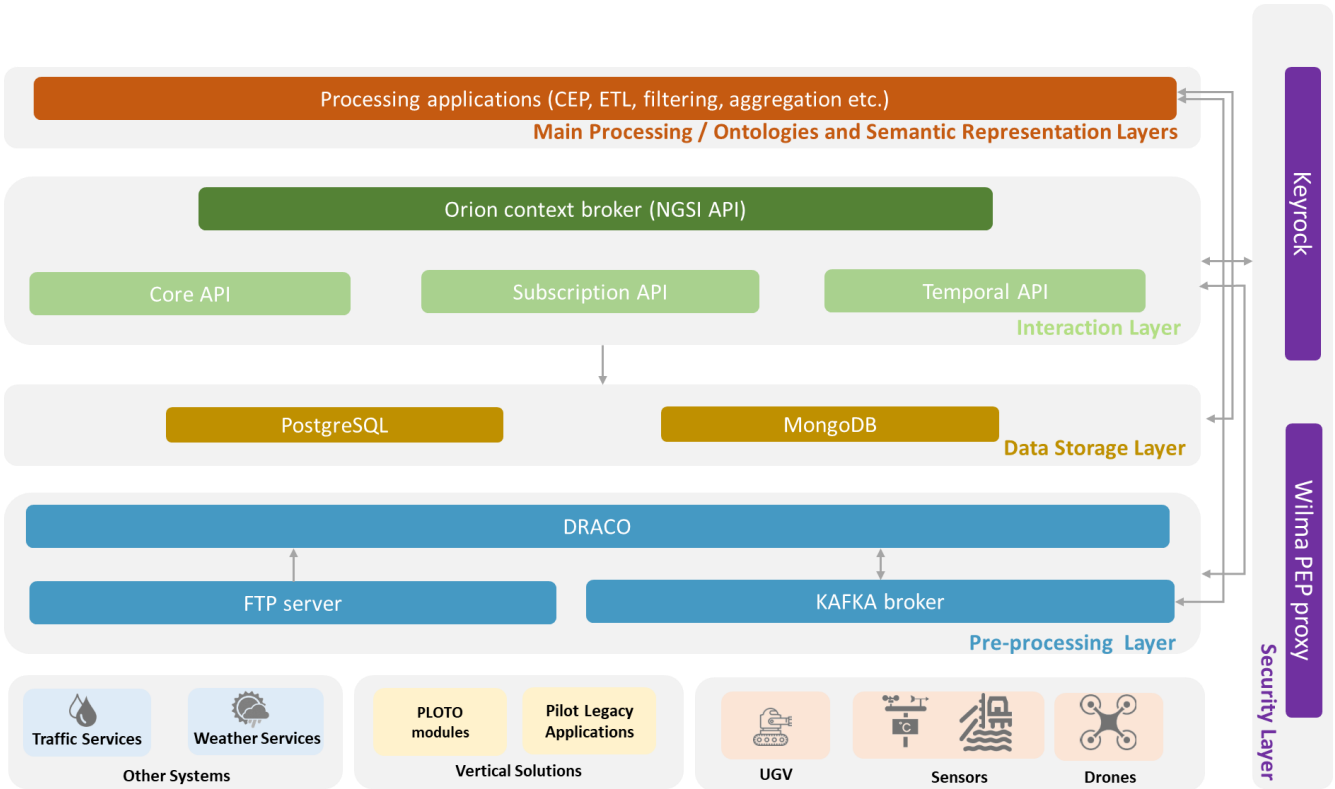

Figure 3. Schematic diagram of HERON middleware

Overall, the architecture is designed to promote modularity, scalability, and efficiency, allowing for flexible expansion and seamless integration of new functionalities and services. In the following paragraphs the key architectural components and their interactions per layer are analysed.

3.1 Pre-processing Layer

The pre-processing layer supports the integration of various data sources and enable seamless information exchange. In doing this, a Kafka broker and an FTP storage is deployed and integrated. It also includes the activities such as normalization and virtualization, ensuring that the data is structured and prepared for storage in a way that facilitates efficient retrieval and analysis, as they will be analysed in the following paragraphs. In the respect of HERON, the middleware utilizes the DRACO component of the FIWARE ecosystem that is based on Apache Nifi component and is configured to build data pipelines that facilitate data virtualization by pulling data from various sources like FTP servers and Kafka. DRACO facilitates the routing of data streams between different components of a FIWARE-based solution. It is a powerful tool for data integration and flow management, offering extensive capabilities for real-time data processing, transformation, and secure data handling. Its role in HERON middleware, particularly in facilitating data virtualization, underscores its value in creating a unified and dynamic data view from diverse sources.

By abstracting the complexities of data integration and providing a user-friendly interface, NiFi streamlines the process of building and managing data pipelines, making it an indispensable component for modern data-driven applications. To meet the aim of the pre-processing layer, the diverse data sources (e.g. sensors, inventory etc.) that the middleware will interact with are first listed and categorized. In the respect of HERON, the following data sets were collected, stored to the middleware, processed and shared with other HERON components.

Road infrastructures data

Data for road infrastructures were gathered and assigned-where available-the geographic location (GeoJSON), encompassing the following information:

- **Road network data:** A road network dataset is a comprehensive collection of data that represents the layout, connectivity, and attributes of roads and highways within a specific geographic area. The dataset includes details about the arrangement and interconnection of roads and highways, providing information on individual road segments. This encompassed specifics such as road type (e.g., highway, local road), the number of lanes, barriers description and other pertinent attributes.
- **Traffic signals and signs data:** The traffic signals and signs dataset contains information about the geospatial location, characteristics, and attributes of traffic signals, road signs, and other traffic control devices.
- **Real time traffic data:** A text delimited file (VDSData) with traffic data recorded by the loops is collected through an FTP- server on real-time. The file is geolocated and timestamped, gets converted into geospatial datasets (.gjson), and encompasses the following information:
  - Date And Time *(* This field represents the date and time when the traffic data was recorded.*)
  - Loop Name *(*This field identifies the specific loop responsible for recording the traffic data.)*
  - Loop description *(*This field describes the specific loop responsible for recording the traffic data.)*
  - Direction (**This field identifies the specific loop's direction)*
  - Lane (**This field likely indicates the lane in which the traffic data was recorded.)*
  - Occupancy *(* This field represents the lane occupancy in percentage.)*

- Average speed *(* This field represents the average speed of vehicles recorded by the loop.)*
- Total Volume *(* Total Volume= Volume for CAT 1 + Volume for CAT 2 + Volume for CAT 3…)*
- Volume for CAT1-8 *(CAT 1 – Motorcycle, CAT 2 – Car, CAT 3 - Van, CAT 4 - Car trailer, CAT 5 – Bus, CAT 6 – Truck, CAT 7 - Truck Trailer, CAT 8 - Articulated Vehicle, CAT 9 - Unknown)*

- **Defects detection data:** The GPS location of the defect will be on the center of mass of the segmentation mask translated in world coordinates using the UAVs GPS location, the current altitude (both stored in the files metadata), and some parameters of the camera (focal length etc). Data is collected via the Kafka broker.

In summary, the dataset captures essential information about traffic conditions recorded by loops, including temporal, spatial, and quantitative aspects. Analyzing this dataset can offer valuable insights into traffic patterns, congestion levels, and the overall performance of the transportation system at specific locations and times.

Other useful data

Other useful for the HERON platform data that will be collected are presented below:

- **Mission planning** bounding box that is defined by GPS WGS 84 locations.
- The **speed limit dataset** contains information about the maximum allowed speeds on different road segments within a specific area. Analyzing this dataset is crucial for understanding and managing traffic regulations, ensuring road safety, and optimizing transportation systems. This dataset is a valuable resource for transportation authorities, law enforcement, and city planners in enhancing overall traffic safety and efficiency.
- The **safety barriers dataset** contains information about the characteristics, and attributes of safety barriers installed along roads and highways. Analyzing this dataset is essential for understanding and managing road safety measures, identifying potential hazards, and improving overall transportation infrastructure. This dataset is valuable for transportation authorities, road maintenance crews, and safety engineers in enhancing overall road safety measures.
- The **road conditions and maintenance dataset** contains information about the condition of roads, including details about maintenance activities and interventions. Analyzing this dataset is essential for ensuring the proper functioning of road infrastructure, addressing safety concerns, and planning effective maintenance strategies.
- **The meteorological conditions dataset** contains information related to weather conditions, atmospheric variables, and climate observations.

This layer also encompasses data processing functions, such as normalization, virtualization, augmentation, as explained below.

3.1.1 Data normalisation and virtualisation

Data normalization is a process that involves mapping and transforming data from various sources into a common schema, ensuring a unified representation for normalized data by aligning data structures, field names, and data types. This also includes standardizing data values and formats, such as date formats and units of measurement, to maintain consistency. Additionally, data normalization identifies and rectifies errors, inconsistencies, or missing

values in the data, creating mappings for categorical or coded values to ensure a consistent representation across datasets. The ultimate goal is to curate datasets for consistency and standardization across different sources or systems, enabling effective comparison, analysis, and reporting of data.

This is supported by leveraging FIWARE smart data models relevant to HERON, such as Transportation, Traffic Flow Observed and WeatherObserved that enhance standardization, interoperability, and community collaboration, establishing a robust foundation for representing and interpreting data across diverse contexts. In the respect of HERON, the middleware receives for instance weather data and transforms them in a standardised format (based on the WeatherObserved smart data model), as displayed below. Such standardisation supports the seamless integration and exchange of data within the services of the middleware, as well as with third-party applications (internal or external to HERON platform).


```
{
    "id": "urn:ngsi-ld:WeatherObserved:Akrata-Meteo",
    "type": "WeatherObserved",
    "refDevice": {
        "value": "urn:ngsi-ld:Device:Akrata-Meteo",
        "type": "Relationship"
    },
    "precipitateType": {
        "value": {
            "value": 8,
            "error": 0,
            "description": "Snow"
        },
        "type": "StructuredValue"
    },
    "location": {
        "value": {
            "coordinates": [
                22.347681,
                38.147275
            ],
            "type": "Point"
        },
        "type": "GeoProperty"
    },
    "dateObserved": {
        "value": "2023-07-13 00:00:00.0000000",
        "type": "Property"
    }
}
```


In a similar line, the traffic data received from the middleware get transformed based on the Traffic Flow Observed smart data model where possible, as displayed in the example below.


```
{
    "id": "urn:ngsi-ld:TrafficData:VDS-2155-2-EPT-T-1",
    "type": "TrafficData",
    "loopName": {
        "value": "VDS-2155-2",
        "type": "Property"
```

```
    },
    "refDevice": {
        "value": "urn:ngsi-ld:Device:VDS-2155-2-EPT-T-1",
        "type": "Relationship"
    },
    "direction": {
        "value": "EPT-T",
        "type": "Property"
    },
    "lane": {
        "value": 1,
        "type": "Property"
    },
    "occupancy": {
        "value": 0,
        "type": "Property"
    },
    "averageSpeed": {
        "value": 0,
        "type": "Property"
    },
    "totalVolume": {
        "value": 2,
        "type": "Property"
    },
    "timestamp": {
        "value": "2023-03-21 00:00:01",
        "type": "Property"
    }
}
```


Data virtualization via the DRACO component enables the seamless creation of a unified view by integrating data from various databases, file systems, or applications in real-time, fostering a dynamic and comprehensive representation. This process involves the creation of virtual layers that effectively abstract the complexities of the underlying physical data sources, allowing users to interact with and query data in a unified manner. Data virtualization streamlines the access to and management of disparate data, providing a unified and simplified experience for users, irrespective of the diverse origins of the underlying data. Together, these functionalities contribute to more efficient and flexible data management and integration within middleware systems.

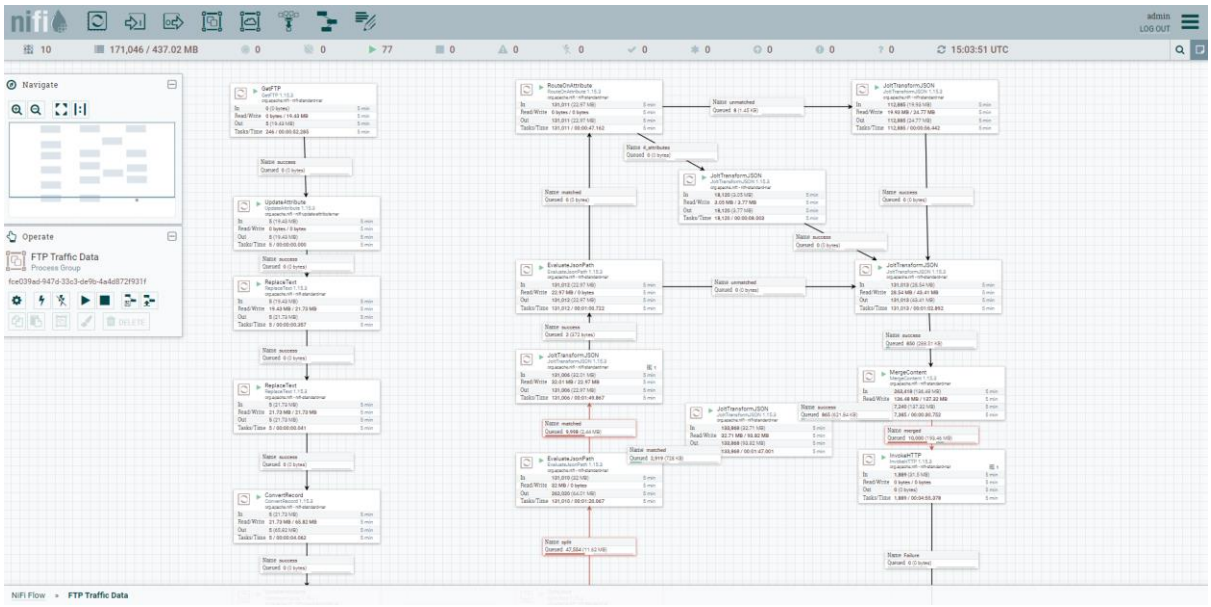

Figure 4: Dataflow chain for traffic data

3.1.2 Metadata augmentation

Metadata augmentation enriches the metadata associated with digital assets, such as documents or datasets. This solution enhances the metadata of datasets by adding common properties such as the provider's information, the time of ingestion, and other relevant details. This augmentation process ensures that the data becomes more easily discoverable and accessible to other tools within the application layer. The primary goal of metadata augmentation is to enhance the comprehensiveness, accuracy, and relevance of metadata associated with digital assets. By providing additional context and information about the content, metadata augmentation improves the overall quality of the dataset. This enhanced metadata not only helps in organizing and categorizing data but also supports efficient querying and retrieval processes. For instance, metadata augmentation can add descriptive tags, keywords, or categorizations to documents or datasets, making it easier for users to search and locate specific information. Additionally, it can standardize metadata formats across different datasets, ensuring consistency and interoperability across various systems and applications.

As explained in Section 3.5, data received get standardised based on smart data models. Along with this process, metadata of the received properties are added to support interpretation and understanding of the received values, as presented in the snippet below.

```
{
      ...
        "precipitateType": {
            "value": 3,
            "error": 0,
            "description": "No precipitation"
        }
        ...
}
```

Overall, metadata augmentation plays a crucial role in improving the usability and effectiveness of digital assets by enhancing their metadata, ultimately facilitating better data management, discovery, and utilization.

3.2 Data storage Layer

The storage component within the middleware and Data Fusion services system is not just a passive element; it is the backbone that supports the entire data ecosystem. Its role extends beyond simple storage and retrieval, encompassing critical functions such as data organization, accessibility, and resilience.

At the core of the middleware there is a meticulously designed repository capable of accommodating a wide spectrum of data types and sizes. This repository serves as a centralized hub where structured, semi-structured, and unstructured data, along with their corresponding metadata from both the pre-processing and processing layers, are stored. By handling data in both original and processed formats, the middleware ensures versatility in managing diverse data sources and their evolving requirements. At its core, the middleware incorporates a sophisticated repository designed to handle diverse and extensive datasets, encompassing robust backup and recovery processes.

This component securely stores structured and semi-structured data, along with associated metadata, in their native or processed formats. Continuous integration of real-time data streams into the repository, demanding scale and flexibility, is managed through data migration services described earlier. Simultaneously, batch integration stores data based on its nature. In the respect of HERON, several storage systems are employed, such as MongoDB and PostgreSQL that supports the historically accumulated datasets.

3.3 Main-processing Layer

The main-processing layer is responsible for processing data in preparation for further use. This layer typically involves the use of processing pipelines that can operate in the batch and streaming paradigm. One of the main challenges that HERON faces is the data fusion of multimodal information in highly heterogeneous datasets, as well as the different granularity levels in time and in space. HERON datasets exhibit diversity arising from multiple sources, characterized by differences in spatial and temporal granularity, formatting, and data quality. The middleware's and Data Fusion services' crucial capabilities lie in the fusion and harmonization of these diverse datasets, addressing challenges related to their heterogeneity.

In this layer, the main-processing functionality is focusing on data processing, and features such as real-time traffic dataset analysis, incident management, speed limits dataset analysis, safety barriers dataset analysis, road conditions, and maintenance dataset analysis, as well as gap filling, underscore the middleware's capacity to derive valuable insights and perform in-depth analysis on varied datasets, as presented below:

- **Gap filling:** Filling the missing values at spatial and temporal dimension.
- **Analyse real time traffic datasets:** Data analysis helps in understanding the flow of traffic over time, identifying peak hours, and determining average traffic speeds. Data analysis allows for the identification of congestion points and the assessment of its severity. Road operators can evaluate the effectiveness of implemented traffic management strategies.

- **Time-of-Day Patterns:** Analyzing historical traffic patterns based on time of day, day of the week, or season helps operators predict and manage regular traffic fluctuations. This allows for better optimization of signal timings and traffic flow.
- **Traffic Incident History:** Data analysis helps in detecting abnormal traffic patterns caused by accidents, breakdowns, or other incidents. Data analysis can identify high-risk zones for traffic incidents.
- **Analyse speed limits datasets:** Create a frequency distribution to visualize the distribution of average speed across different road segments. Explore potential correlations between average speeds and other factors, such as road type, location, or speed limit.
- **Mission planning:** The middleware retrieves the mission planning details and breaks it down in to corresponding subtasks. Each subtask is sent through Kafka to the HLP and stored under the same mission.
- **Road Construction and Maintenance:** Integrating information on ongoing road construction or maintenance activities with traffic fluctuations, allows operators to plan maintenance activities and traffic management measures. This minimizes disruptions and ensures smooth traffic flow.

3.3.1 Resource Management (RM)

Normalization and virtualization are closely linked, particularly in their association with Resource Management (RM). RM provides a framework for storing, retrieving, and managing metadata and data, as well as handles filtering, aggregation, and fusion of data. This integration enhances the overall capabilities of the middleware, allowing for effective management of information resources and optimizing data processing. The utilization of the FIWARE subscriptions mechanism is significant. This mechanism is employed to build historic data through a time-series database. It enables the middleware to capture and store data over time, facilitating the analysis of historical trends and patterns. HERON employs the Draco[1] component for persisting context data and creating a historical view of the context. The script below displays an example of the subscription of Draco to updates on TrafficData that are subsequently sinked to a dedicated table of a PostgreSQL relational database.

```
url = "http://{{orion-context-broker}}/v2/subscriptions/"

payload = json.dumps({
  "description": "Traffic Data Subscription for Draco",
  "subject": {
    "entities": [
      {
        "idPattern": ".*",
        "type": "TrafficData"
      }
    ]
  },
  "notification": {
```

[1] https://fiware-draco.readthedocs.io/en/latest/

```
    "http": {
      "url": "http://draco:5050/v2/notify"
    },
    "attrs": [
      "loopName",
      "direction",
      "lane",
      "occupancy",
      'averageSpeed',
      'totalVolume',
      'volumeCat0'
      'volumeCat1',
      'volumeCat2',
      'volumeCat3',
      'volumeCat4',
      'volumeCat5',
      'volumeCat6',
      'volumeCat7',
      'volumeCat8',
    ]
  }
})
headers = {
  'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)
```

Figure 5: Subscription mechanism

3.3.2 Scalability and Performance

The middleware employs batch processing tools, leveraging the Extract-Transform-Load (ETL), querying, and aggregation capabilities of Apache Flink. This open-source distributed processing engine enhances scalability, allowing the middleware to efficiently handle substantial datasets in a distributed computing environment. Apache Flink is designed to read and write data from different storage systems as well as to consume data from streaming systems such as Kafka and MongoDB.

In HERON, the middleware leverages the available Flink connectors and creates custom business logic to listen for context data subscription events and then process the flow of the context data. This integration with Apache Flink enables the middleware to efficiently extract, transform, and load data, as well as conduct queries and aggregations on large-scale datasets in a distributed computing environment. Apache Flink's robust features contribute to the middleware's ability to handle substantial data processing tasks with scalability and performance.

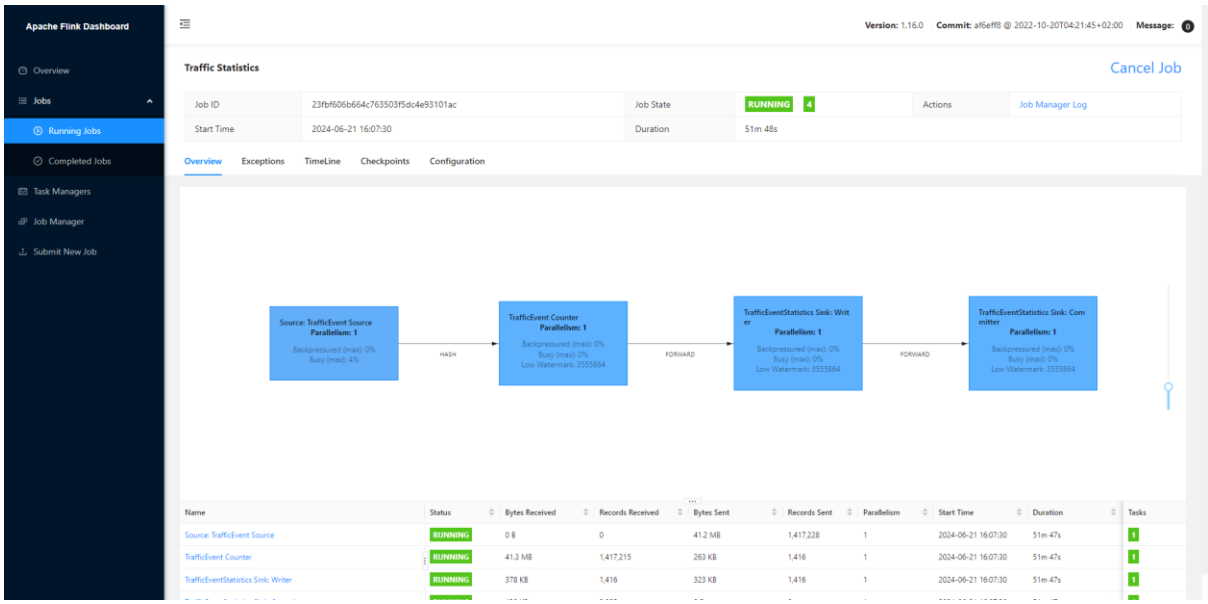

Figure 6: Flink's web UI to monitor the status of the cluster and running jobs

3.3.3 Event Management (EM) and Complex Event Processing (CEP)

The middleware's support for extracting datasets for analysis algorithms establishes a significant relationship with Event Management (EM) and Complex Event Processing (CEP), contributing to a comprehensive and dynamic data analysis framework.

Event Management (EM)

Event Management (EM) is a crucial element within the HERON platform, overseeing the handling of events and contextual information. Its responsibilities include managing retrieved events by categorizing them into relevant classifications, ultimately enhancing comprehension and processing capabilities. This categorization, based on unified schemata, contributes to faster data retrieval and processing, ensuring consistency in the interpretation and structuring of events and information within the middleware.

Transformed and analyzed events lead to the generation of additional events and data aggregations, enriching the overall information available in the system. This iterative process of event enrichment ensures that the platform remains adaptable and effective in addressing evolving business challenges and opportunities.

Complex Event Processing (CEP)

The middleware utilizes its streaming processing capabilities and employs Complex Event Processing (CEP) for real-time sensor data alert generation. CEP involves the instantaneous analysis and response to data streams, enabling swift detection of specific conditions or events necessitating immediate attention. This approach facilitates the identification of intricate patterns, triggering alerts based on predefined rules for data aggregation, and subsequently disseminates these alerts to a Kafka broker for external application subscription.

By integrating CEP, the middleware enhances its ability to offer real-time insights, enabling the immediate identification of significant events through continuous dataset analysis. This approach harnesses contextual understanding derived from extracted datasets to make informed decisions in real-time, bolstering the middleware's efficacy in real-time monitoring and response.

The integration of Apache Flink and CEP within the HERON middleware significantly enhances its data processing and analysis capabilities. By addressing the challenges of diverse datasets and enabling real-time processing and analysis, the middleware supports a comprehensive and dynamic data management framework. This framework not only ensures the consistency and reliability of the data but also empowers the platform to deliver timely and actionable insights, ultimately driving better decision-making and operational efficiency.

In summary, the HERON platform's middleware and DF services leverage advanced technologies like Apache Flink and CEP to manage and process diverse datasets effectively. Through real-time data streaming, batch processing, and dynamic event handling, the platform provides a robust and scalable solution for comprehensive data analysis and real-time monitoring, catering to the complex needs of modern data-driven environments.

Rule Editor for Dynamic Event Processing

Administrators use a web application rule editor to dynamically create, update, or delete processing rules, establishing a flexible and adaptive system for event processing. The rule editor allows administrators to define triggering conditions, which are essential for the CEP framework to function effectively.

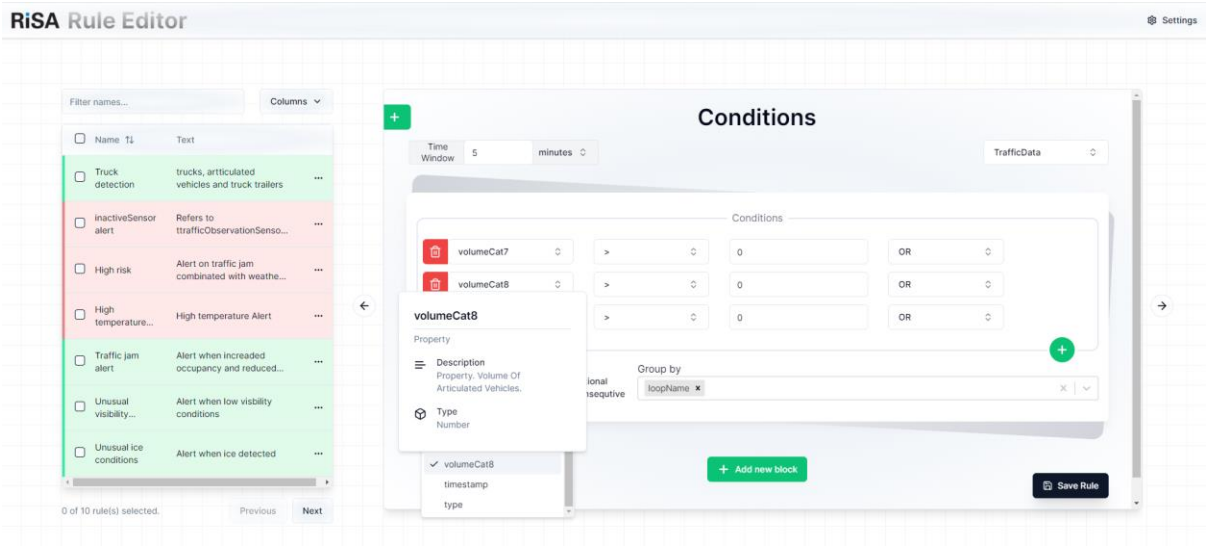

Figure 7: Rule editor for alert generation

There are three main sections per rule, presented in the next table that can be edited through the rule editor.

Table 1: Sections of the rules definition

Section	Description	Properties
Information	General information about the rule.	• name of the rule • description • creation time • time of the last change

Section	Description	Properties
Pattern	The pattern of expected sequences of events/instances that should trigger an alarm generation.	• individual and combining patterns (see below for more details)
Alert	Definition of the alert that will be generated.	• ID • Description • priority (0-10)

Patterns are categorized into two primary types: individual patterns and combining patterns. Each pattern, whether individual or combining, may include one or more conditions. These conditions play a crucial role in determining whether a single event or multiple events will be chosen to trigger the generation of an alert. The flexibility in defining conditions allows for a nuanced approach to alert generation based on specific criteria associated with individual or combined events within the pattern.

Simple conditions: This type of condition determines whether to validate entity data based on inherent properties of the entity itself. A comprehensive list of properties associated with the entity data object is at the user's disposal, facilitating the definition of pattern constraints. This approach enables a nuanced and customizable assessment of alerts, taking into account specific characteristics or attributes associated with the detection process.

Combining conditions: Multiple properties can be incorporated as part of the pattern constraints, and in such instances, the user is required to specify the logical relationship (AND/OR) between these constraints. This customization allows for a flexible and tailored definition of pattern criteria based on the logical connections between various properties.

Looping conditions: For each individual pattern, the following looping patterns are available:

- oneOrMore, for patterns that expect one or more occurrences of a given event
- times(#ofTimes), for patterns that expect a specific number of occurrences of a given type of event; in this case the expected number of occurrences should be provided
- times(#fromTimes, #toTimes) for patterns that expect a specific minimum number of occurrences and a maximum number of occurrences of a given type of event. The minimum and maximum thresholds are also parameters to be defined.

The optional pattern is applicable to all aforementioned quantifiers and specifies that an occurrence may not occur at all. Additionally, there is the option to impose strict contiguity between matching events with the consecutive pattern in conjunction with oneOrMore and times. With this option, any non-matching input data breaks the match.

Iterative conditions: For each pattern, an iterative condition can be specified that accepts subsequent events based on properties of the previously accepted event. The properties are a subset of the properties of the entity data and are specified per each individual pattern.

Time constraints: A time constraint can be defined. It states that all events need to happen within a specific time span to have a match.

Combining patterns can be defined by appending patterns to a sequence of individual patterns and specifying the desired contiguity conditions between them. The following options of contiguity between events are available:

- next, for strict contiguity that expects all matching events to appear strictly one after the other, without any non-matching events in-between
- followedBy, for relaxed contiguity that ignores non-matching events appearing in-between the matching ones
- followedByAny, for allowing additional matches that ignore some matching events.

Second step is the processing: Recognizing the criticality of time in the alert generation process, the middleware has been specifically crafted to handle real-time events delivered through the Orion Context broker. To accomplish this, the middleware harnesses the capabilities of the open-source Apache Flink. This technology operates on a reactive and asynchronous principle, enabling the detection of patterns within an uninterrupted stream of input data. This approach ensures the timely identification of significant information, aligning with the imperative need for swift and responsive alerting in dynamic environments.

As a final step the generated alerts are enriched with valuable information and provide them for further use again to the Kafka and the Orion context broker.

In summary, the middleware's extraction of datasets creates a robust and dynamic framework, enabling real-time analysis, immediate response to events, and adaptive decision-making based on the insights, patterns, and correlations uncovered through the analysis. The fused information resulting from processing and analysis is made accessible through APIs and messaging services. This allows external systems, such as the Decision Support System (DSS) and other modules requiring additional data, to retrieve meaningful information for decision-making and analysis. Raw data is also made available to high-level modules and applications, enabling further application-driven processing. This flexibility ensures that different stakeholders, such as road managers and relevant authorities, can access and utilize data in a manner that aligns with their specific needs and objectives.

In summary, the second layer of middleware is a comprehensive processing hub that not only handles the transformation and storage of data but also integrates seamlessly with resource management, event management, and historical context management. This integration ensures that the HERON platform can effectively manage, process, and provide meaningful insights from the diverse data generated in the initial layer.

3.4 Interaction Layer

Interaction layer’s focus is on providing standardized APIs that are essential for facilitating interaction between the platform components and the HERON middleware and DF services system. Its objective is to establish a uniform interaction layer through the creation of an API specification. The API specification defines the protocols, methods, and data formats that components within the platform should adhere to when communicating with the middleware. This adherence ensures that each component communicates in a standardized manner, promoting a cohesive ecosystem within the platform. Consistency in API usage simplifies integration efforts and enhances overall system reliability.

The development of standardized APIs enhances interoperability by providing a common language for communication. This is particularly important in a multi-component system, such as HERON, where seamless interaction is critical. The standardized APIs act as bridges between different elements of the platform, fostering a more integrated and collaborative environment.The API is accessible at heron-middleware.risa.eu and is aligned with the ETSI

NGSI-v2 and NGSI-LD standards, leveraging on the Orion-LD context broker that is the cornerstone of this "Powered by FIWARE" platform. As the platform evolved, new modules and components were added without disrupting existing functionalities. The standardized APIs (https://swagger.lab.fiware.org/) contribute to the flexibility and adaptability of the system architecture. FIWARE NGSI is the API exported by the Context Broker, used for the integration of platform components within the middleware and by other applications to update or consume context information. Moreover, HERON middleware extends the Orion's Context broker default API by endpoints that allow the retrieval of historical datasets (https://app.swaggerhub.com/apis-docs/i.kourentzis/TemporalAPI/1.0.0-oas3.1).

Indicatively, to get all traffic data from a specific device, client applications may use:

```
curl --location 'heron-middleware.risa.eu/temporal/livedata/?id=urn%3Angsi-
ld%3ATrafficData%3AVDS-5110-2-EPT-E-2' --header 'X-Auth-Token: 1YYBc3pboIuBwcB'
```

To get traffic data from a device within specific time period, use:

```
curl --location 'heron-middleware.risa.eu/temporal/livedata/?id=urn%3Angsi-
ld%3ATrafficData%3AVDS-5110-2-EPT-E-2&timerel=between&time=2023-01-
21%2010%3A53%3A00&endtime=2023-03-21%2010%3A54%3A00' --header 'X-Auth-Token:
1YYBc3pboIuBwcB'
```

In addition, REST APIs play a pivotal role in facilitating data integration, particularly for **HERON data regularly pushed in synchronous batch mode (e.g. sensors data, real-time traffic data),** while ensuring comprehensive data protection across layers, as analysed below:

- Sensor registration is streamlined via OGC-compliant SensorML or NGSI service embedded in the available REST API, establishing a seamless connection between sensors and the middleware.
- Real-time data (e.g. on traffic) collection involves acquiring text-delimited files (VDSData) via an FTP server, which are then geolocated, timestamped, and transformed into geospatial datasets (.gjson). These datasets encompass vital information such as the road network, traffic signals and signs, bridges and tunnels, as well as pilot-related raw files including safety barriers, road conditions and maintenance, accident and incident data, and environmental factors.
- Collected observations from sensors or other static information are pushed via appropriate NGSI requests to the Orion Context broker.

In essence, the development of standardized APIs for HERON middleware interaction is instrumental in creating a robust, interoperable, and scalable platform. This initiative lays the groundwork for consistent communication between components, fostering a cohesive and efficient ecosystem for data exchange within the research infrastructure.

3.5 Ontologies and Semantic Representation Layer

Ontologies and Semantic Representation Layer serves as a critical bridge in ensuring the optimal functioning of RIs. The primary goal of this layer is to enhance interoperability among diverse system components. By establishing common ontologies and semantic representations, it creates a shared understanding of data across different RIs. This shared knowledge is essential for seamless communication and collaboration among various components within the

system. The layer plays a pivotal role in identifying and organizing data collected from various components and information systems. By doing so, it sets the foundation for coherent and structured data management. This organized approach is crucial for efficient utilization and retrieval of information within the RIs.

The Ontologies and Semantic Representation layer contributes to the creation of a foundational level of shared knowledge. This shared understanding is essential for harmonizing data and ensuring consistency in how information is represented and interpreted across different components of the system. HERON leverages linked data capabilities, which involve establishing relationships between entities. This interconnectedness enhances the comprehensibility and context of data. By utilizing linked data, the system can establish meaningful connections between different pieces of information, providing a more holistic view for users.

A key challenge in integrating data from diverse sources is dealing with ambiguity. The Ontologies and Semantic Representation layer addresses this challenge by creating common data models. These models serve as a standardized framework, reducing ambiguity and ensuring a consistent interpretation of data, which is vital for accurate analysis and decision-making.

FIWARE employs an annotation mechanism attribute that highlights an additional strategy for achieving a unified schema. This mechanism, capable of referencing semantic taxonomy model definitions, ensures that a standardized schema is adopted across the system. This approach simplifies the integration of data from different sources and promotes a cohesive data structure.

In essence, the fourth layer acts as a critical enabler for ensuring efficiency and functionality within the research infrastructure. By providing a common understanding of data and establishing a standardized approach, it facilitates smooth communication and interaction among the various components, ultimately contributing to the overall effectiveness of the system.

The Ontologies and Semantic Representation Layer involves the creation, refinement, and management of concepts and ontologies. This plays a crucial role in representing knowledge in a structured and semantically rich manner. It includes features such as Conceptual Modeling, where conceptual models are defined using ontologies to represent key concepts, entities, relationships, and attributes within a specific domain; Semantic Annotation, which involves applying explicit meanings and relationships to data elements for enhanced interpretability; and Ontology Development and Management, facilitating the creation, modification, and management of ontologies defining vocabulary, taxonomy, and semantics in a specific domain.

Additionally, in this Layer, the alignment with Smart Data Models signifies the integration and compatibility of the layer with Smart Data Models - standardized, domain-specific data models designed for information representation and exchange within specific industries or application domains. The incorporation of FIWARE smart data models, including data models relevant to HERON like Transportation, WeatherObserved and Traffic Flow Observed, enhances standardization, interoperability, and community collaboration within the Ontologies and Semantic Representation Layer, establishing a robust foundation for representing and interpreting data across diverse contexts.

Table 2: Data models relevant to HERON

Data models relevant to HERON	Entity types relevant to HERON
Transportation	**Road.** This entity contains a harmonised geographic and contextual description of a road. **CityWork.** The Data Model is a contextual description of urban works carried out on a road axis and which can impact individual (cars, motorcycle, bicycles, ….) or common transport (Tram, Bus, subway).
WeatherObserved	An observation of weather conditions at a certain place and time. This data model has been developed in cooperation with mobile operators and the GSMA.
TrafficFlowObserved	This entity describes the traffic flow conditions at a certain place and time.

3.6 Security Layer

To ensure security in the middleware, confidentiality, integrity and information availability are applied in combination. Confidentiality intends to prevent unauthorised access to information. Each user has access only to the resources allowed, according to permissions and the resources are accessible only to authorised users. Any component of the system that wants to publish or receive data via the middleware API must first successfully complete the authentication process by logging in to the system with its credentials.

The middleware uses the FIWARE Wilma PEP Proxy combined with FIWARE Keyrock to secure access to FIWARE Orion-LD endpoints exposed by FIWARE generic enablers. Users (or other actors) must log-in and use a OAuth token to gain access to the secured access points. Configuration at the proxy level control permissions according to which users are authorised to access resources.

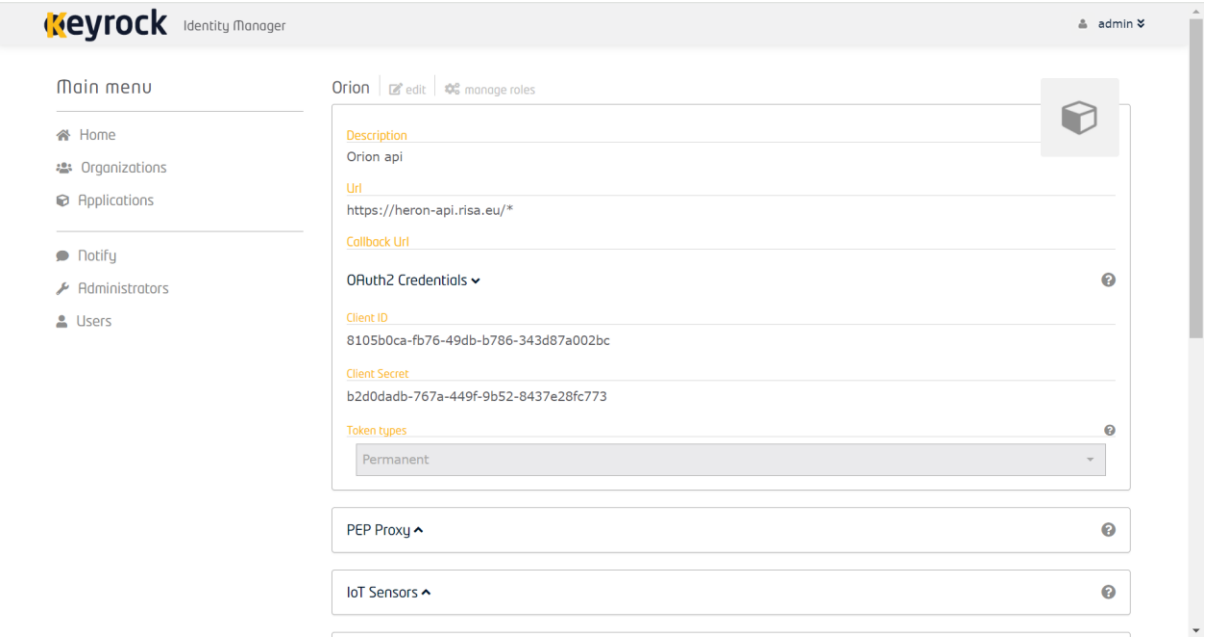

Figure 8. Configuration defined in Keyrock

At the same time, integrity aims to prevent unauthorised modification of information. A user will not be able to modify data that permissions do not allow it. The communication via the REST-API will be over an encrypted communication channel (TLS) to secure traffic, thus ensuring reasonable protection from eavesdroppers and man-in-the-middle attacks and the integrity of the data.

In summary, the middleware uses a robust security framework that includes measures for confidentiality, integrity, and availability. By implementing stringent authentication processes, access control mechanisms, and encrypted communication channels, it ensures that sensitive information is protected from unauthorized access and manipulation.

4 Conclusions

The current report falls under the scope of WP6 within the HERON project, focusing on Communication and Networking Solutions, DSS, IMS, and CoP, under Grant Agreement No. 955356. It provides a comprehensive overview of the middleware, which serves as the central system for data acquisition, storage, and management within the HERON project. The middleware operates on a scalable architecture, offering high-performance communication APIs and messaging services designed to streamline data exchange with other components, accommodating both synchronous and asynchronous modes. Efficient coordination of information delivery between modules and services ensures seamless and scalable service assurance, enhancing overall system performance. Acting as a broker across network communication interfaces, the middleware ensures that connected services and modules present information in the appropriate format for the HERON platform or other information systems, facilitating interoperability. Moreover, it incorporates capabilities for processing and fusion to enhance data integration and analysis within the system, improving decision-making processes. This deliverable which is the outcome of Task 6.2, presents the architecture of the middleware, as well as its functionalities and technical solutions provided; and has been authored with contributions from all technical partners of the HERON project, indicating collaborative effort and expertise in its development.