SEcure Decentralised Intelligent Data MARKetplace

D4.4 Edge data processing and service certification – Final version

Document Identification	
Contractual delivery date:	31/07/2025
Actual delivery date:	28/08/2025
Responsible beneficiary:	EGM
Contributing beneficiaries:	NUID UCD, SURREY
Dissemination level:	PU
Version:	1.0
Status:	Final

Keywords:
Tools, Data processing

This document is issued within the frame and for the purpose of the SEDIMARK project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No.101070074. and is also partly funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission or UKRI.

Document Information

Document Identification			
Related WP	WP4	**Related Deliverables(s):**	D3.2, D3.4
Document reference:	SEDIMARK_D4.4	**Total number of pages:**	32

List of Contributors	
Name	**Partner**
Gilles Orazi, Franck Le Gall, Léa Robert, Iheb Khelifi, Thomas Bousselin	EGM
Diarmuid O'Reilly Morgan, Erika Duriakova, Honghui Du Elias Tragos, Qinqin Wang, Aonghus Lawlor, Neil Hurley	NUID UCD
Tarek Elsaleh	SURREY

Document History			
Version	**Date**	**Change editors**	**Change**
0.1	19/03/2025	EGM	First version of document structure (table of content)
0.2	25/06/2025	NUID UCD, SURREY, EGM	Contributions in different sections
0.3	15/07/2025	EGM	Finalisation and preparation for review.
0.4	31/07/2025	EGM, SURREY	Integrated reviewers comments
0.9	08/08/2025	ATOS	Quality Format Review
1.0	28/08/2025	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval date**
Reviewer 1	Luis Sanchez (UC)	29/07/2025
Reviewer 2	Stefan Jarcau, Gabriel Danciu (SIE)	17/07/2025
Quality manager	María Guadalupe Rodríguez (ATOS)	28/08/2025
Project Coordinator	Miguel Ángel Esbrí (ATOS)	28/08/2025

Table of Contents

List of tables

List of figures

List of Acronyms

Abbreviation / acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under ROC curve
CPU	Central Processing Unit
Dx.y	Deliverable number y belonging to WP x
ETSI	European Telecommunications Standards Institute
EU	European Union
HTTP(S)	HyperText Transfer Protocol (Secure)
HTML	HyperText Markup Language
IoT	Internet of Things
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
JSON-LD	JavaScript Object Notation for Linked Data
KPI	Key Performance Indicator
MAE	Mean Absolute Error
MCU	MicroController Unit
ML	Machine Learning
MLOps	Machine Learning Operations
NiFi	NiagaraFiles (Apache Software Foundation)
NIST	National Institute of Standards and Technology
NGSI-LD	Next Generation Service Interface – Linked Data
OECD	Organisation for Economic Co-operation and Development
ONNX	Open Neural Network Exchange
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	RDF Schema
RMSE	Root Mean Square Error

Abbreviation / acronym	Description
ROC	Receiver Operating Characteristic
RTC	Real-Time Clock
SHACL	Shapes Constraint Language
SQL	Structured Query Language
UI	User Interface
URI/URN	Uniform Resource Identifier / Uniform Resource Name
UART	Universal Asynchronous Receiver-Transmitter:
W3C	World Wide Web Consortium
WP	Work Package
WASM	WebAssembly
YAML	YAML Ain't Markup Language

Executive Summary

This report details the final architecture and tools developed within the SEDIMARK project for edge processing and services certification. The work focuses on providing the foundational components for managing the entire lifecycle of data and AI assets, from their creation and processing at the edge to their certification and exchange in the marketplace. The key contributions establish a framework for AI-driven modules that can be deployed at the data source, adhering to MLOps principles while managing complex edge-cloud interactions.

A primary achievement of this work is the development of an edge processing framework designed for resource-constrained environments. Key innovations include:

- WebAssembly (WASM) on MCUs: A secure and sandboxed architecture was implemented to allow user-defined code to run on low-power microcontrollers, enabling flexible and frequent updates without compromising the core firmware's stability.
- Fine Timestamping and Energy Optimization: A novel algorithm was developed to calibrate the on-device Real-Time Clock (RTC) by compensating for temperature-induced drift. This significantly improves data timestamp accuracy and dramatically reduces the need for energy-intensive network synchronizations, extending the operational battery life of edge devices to meet a target of over four years.
- Edge-Cloud Orchestration: An analysis of open-source tools led to the selection and deployment of platforms like Mage.ai and Apache NiFi to manage distributed data processing flows between edge devices and the cloud.

To address the challenges of managing AI models in a diverse ecosystem, the project has established a comprehensive MLOps strategy and a solution for model interoperability. By adopting MLFlow, SEDIMARK provides a standardized framework for the entire machine learning lifecycle. A critical innovation is the use of Keras Core to create framework-agnostic model descriptions. This allows models to be defined once and then seamlessly trained or used for inference across different backends like TensorFlow, PyTorch, and JAX, which is essential for fostering collaboration in federated learning scenarios where participants may use different tools.

Finally, to build a foundation of trust within the marketplace, a multi-faceted conformity evaluation Service has been designed. This service provides validation for all marketplace assets:

- Data Assets are certified for conformance with standards like NGSI-LD and Smart Data Models, ensuring interoperability.
- Service Assets are validated for API compliance, leveraging and contributing to the official ETSI NGSI-LD Test Suite.
- AI Model Assets undergo a two-fold assessment, verifying not only their performance against quantitative KPIs but also their trustworthiness based on principles of fairness, transparency, and security, in alignment with emerging regulations like the EU AI Act.

Together, these advancements in edge computing, MLOps, and certification provide the core technical infrastructure for a robust, transparent, and efficient decentralized data marketplace.

1 Introduction

1.1 Purpose of the document

This document is the final version of a reporting of the work in WP4 about tools and processes for enabling data processing and sharing in an interoperable way at the data sources.

1.2 Relation to another project work

The work in SEDIMARK_WP2, reported in SEDIMARK_D2.1 [1] and D2.3 [2] so far, showed that the topics of edge computing for data quality, ML models and MLOps especially linked to federated learning are of special interest in this project. This has driven the work of SEDIMARK_T4.2, reported in this deliverable.

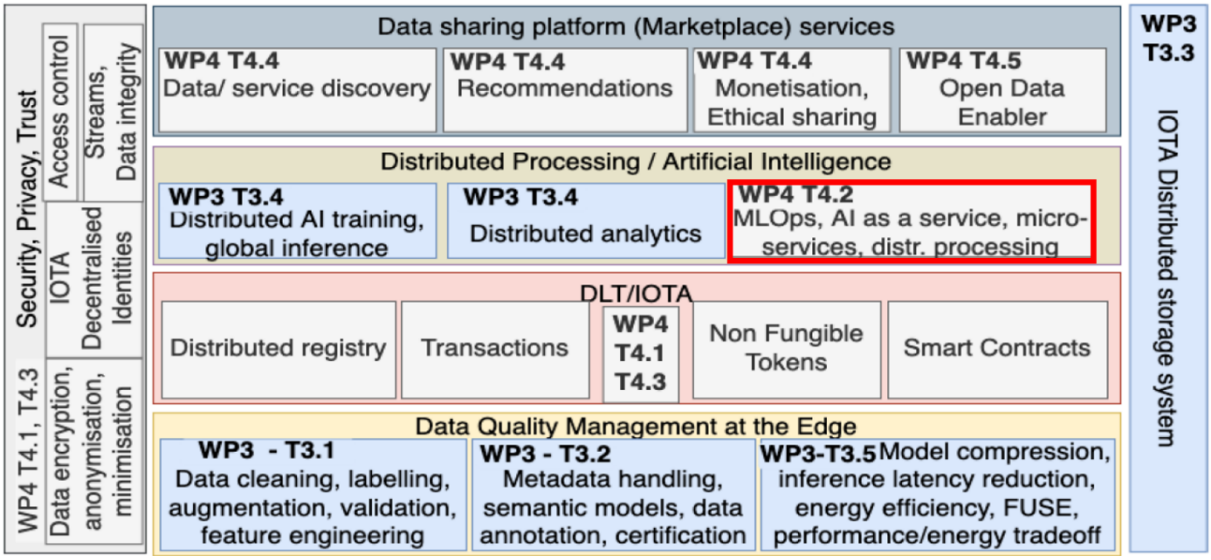

Figure 1: Positioning of distributed processing in SEDIMARK architecture

As one can see in the Figure 1, this task is at the heart of the SEDIMARK platform, in the layer dedicated to distributed processing and artificial intelligence. Its primary objective is to create a framework for the deployment of AI-driven modules that process and share data at edge data sources. It considers the interactions between edge and cloud systems while adhering to MLOps principles. It is thus in tight relationship with SEDIMARK_WP3.

1.3 Structure of the document

After a short introduction to the document (the current part) and a quick overview of the SEDIMARK platform (chapter 2), this document explores two aspects of the management of these data. In chapter 3, the aspects related to the Edge/Cloud interactions are explored by first exposing the challenges and requirements for Edge computing (3.2) and then by looking at how they can be managed by using specific orchestration tools (3.3.1), or how dynamic processing can be implemented even at far edge (3.3.2), where networking and computing resources are very limited. Section 4 quickly review the options investigated for managing interoperability of Machine learning frameworks. Finally, chapter 5 describes a core contribution of the WP4 in relation to the delivery of certification services within the SEDIMARK marketplace components and services.

2 Report contributions to SEDIMARK environment

This deliverable presents the work achieved in Task 4.2, presenting the architecture that has been developed within SEDIMARK to enable data processing and sharing at the edge. SEDIMARK is a decentralised system that allows data providers to collect, clean and process their data at various stages, including at edge devices. This is helpful in cases where large amounts of data are gathered at edge devices, so that once they are cleaned and processed, the communication and storage costs at the provider server is significantly reduced. Additionally, clean data at the edge allows a more efficient machine learning technique, both for training and for inference purposes. Techniques for data anonymisation at the edge are also exploited to hide or remove sensitive information, thus either creating anonymised datasets that can be shared in the marketplace without privacy issues or training ML models that don't reveal or leak private data.

Considering that there are many available frameworks used for training ML models, SEDIMARK also provides a framework for ML model interoperability, exploiting existing well-known platforms. This helps providers to continue to use the frameworks they are familiar with, while at the same time being able to download/purchase models from the SEDIMARK marketplace and use them converting them into their preferred format/framework.

Figure 1 presents the SEDIMARK functional architecture that is described in deliverable SEDIMARK_D2.3 [2] in detail. With orange highlights are the functional components that are part of this deliverable. These components are part of three different layers of SEDIMARK, security, data and intelligence layer

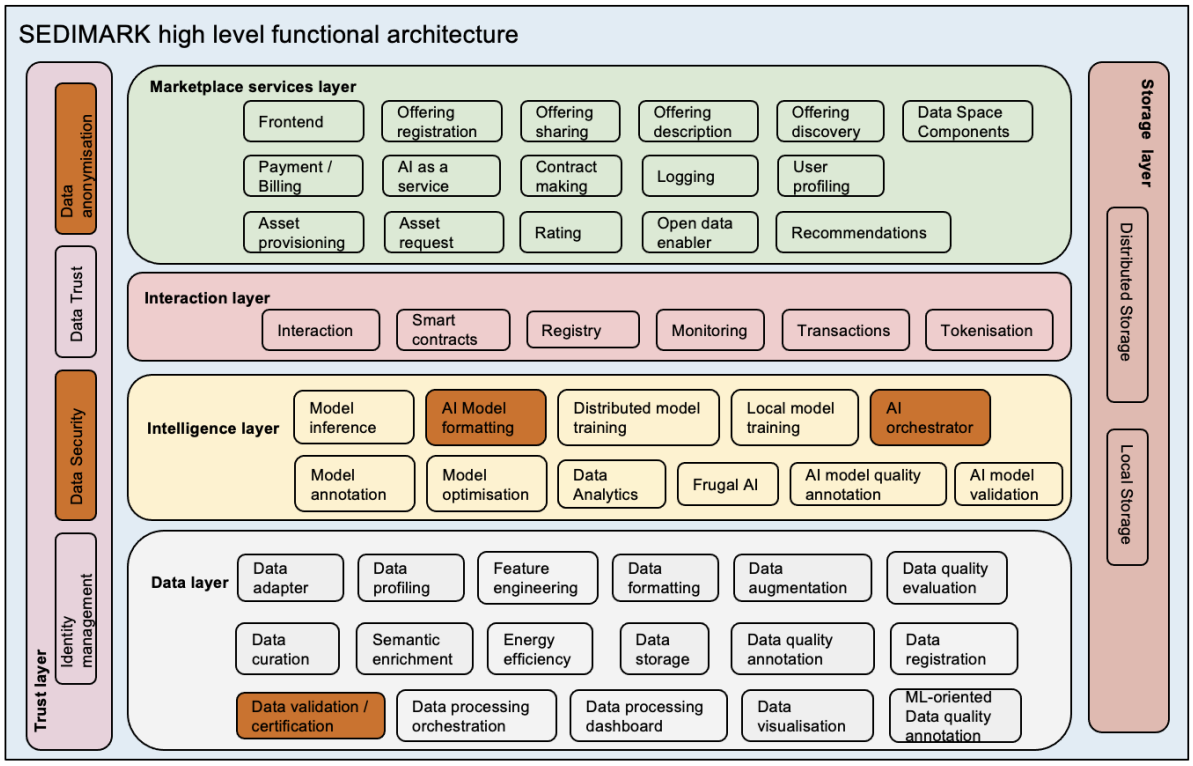

Figure 2: The SEDIMARK functional architecture. Orange highlights functional components that are being part of this deliverable.

3 Architecture for Edge data processing

3.1 Introduction

Edge processing, also known as edge computing, refers to the practice of processing data near the source of generation rather than relying on a centralized cloud-based system. In traditional computing models, data is sent to a remote datacentre or cloud for processing and analysis. However, edge processing brings computational capabilities closer to the "edge" of the network, which is typically where data is generated. In the context of SEDIMARK, this paradigm is in use for distributed AI as well as for potential delocalization of some of the data processing pipeline processes.

The environment in which Edge data processing is performed raises multiple constraints that need to be handled, such as privacy preservation, response time, throughput, and resource consumption (e.g., CPU, memory, energy, bandwidth), while the latter may influence the monetary cost. In the following, some of these requirements are to be considered for the SEDIMARK assets (e.g., Artificial intelligence (AI) model and service assets).

- Bandwidth: While edge data processing reduces the need for transmitting all data to the cloud (federated learning) or between nodes (gossip learning), there is still a need for network connectivity. Limited bandwidth can affect data and synopsis transmission to and from the edge.
- Computing resources: Edge devices often have constrained processing capabilities (e.g., memory and storage). Therefore, running complex and massive data processing tasks on such devices can be challenging.
- Privacy: In the SEDIMARK decentralized environment, privacy naturally arises since personal and sensitive data will be processed, from which real insights about individual behaviour, health, or relationships can be inferred.
- Data quality: The data provided within SEDIMARK can be noisy, duplicated, or incomplete. Ensuring data quality and extracting knowledge from potentially imperfect data is a challenge that needs to be handled. To do so, SEDIMARK provides curation techniques to address imperfect data and improve its quality.

These aspects depend on the framework used to handle the processing distribution as well as the way processes are implemented. In this report, focus is on the tooling which is investigated in the following section. This updated version reports on the evaluation of the tools made in the context of ultra-low power edge devices, as well as algorithms tested to increase data quality on the edge.

3.2 Edge-Cloud Orchestration tools

There exist many edge-cloud orchestration platforms. Identified open-source platforms have been analysed to evaluate how they could support the handling of a data pipeline distributed over cloud and edge. They are the following:

- FogFlow: FogFlow is a FIWARE enabler to orchestrate data processing flows between cloud and edge. It uses intent-based programming. For example, for service consumers, they can specify which type of results are expected under which type of Quality of Service (QoS) within which geo-scope; for data providers, they can specify how their data should be utilized by whom. In FogFlow, orchestration decisions are made to meet those user-

definable objectives during the runtime. It became NGSI-LD compliant in September 2022.

- Apache Airflow: Apache Airflow is a platform to programmatically author, schedule and monitor workflows. It uses Python language to batch processing workflow run at regular intervals. It does not aim at processing event streams. However, coupling with a service bus having storage capabilities such as Apache Kafka allows for periodic processing of stream fragments. Airflow interface is mainly provided for workflows activation and monitoring. Coding in python remain mandatory for workflows definition. It builds on Kubernetes to provide auto-scaling. Apache Airflow 3.0 was released in April 2025, introducing significant architectural changes:
  - Service-Oriented Architecture with a new Task Execution API enabling task execution in remote environments
  - Edge Executor supporting distributed, event-driven, and edge-compute workflows
  - Asset-Based Scheduling with redesigned dataset model for event-driven DAG definition
  - Enhanced ML and AI Workflow Support with logical_date=None capability for model inference and hyperparameter tuning
  - Modern React UI with complete overhaul built on React and FastAPI
- Mage AI: Mage AI aims at simplifying the Apache Airflow experience. It remains coding based, allowing Python, R and SQL in the same data pipeline while the User Interface (UI) focuses on monitoring workflows execution. Both batch and stream processing are allowed. Pipelines can be configured through the set of global variables. Distributed processing is part of the roadmap, considering Ray as a distributed execution framework layer for parallel processing and Dask as Python parallel computing library. Mage AI has expanded its capabilities during the second period of the project:
  - AI-Powered Pipeline Generation from simple prompts, handling setup, code, and configuration automatically
  - Hybrid and Private Cloud Deployment options with cloud control plane and private data processing
  - Enhanced Data Integration with embedded Python logic directly in syncs for data cleaning and enrichment
- Apache NiFi: Apache NiFi also aims at implementing workflow defined as DAG. However, in contrast to Airflow, it provides aa highly configurable web-based interface to define the workflow which can consider either stream or batch processing. Hundreds of existing connectors enable the ingesting of data from almost any kind of source. External scripts or executables can be called thus making Apache NiFi completely customizable. NiFi continues to evolve with recent improvements:
  - NiFi 2.0+ Features including stateless mode support and Python custom processor capabilities
  - Enhanced Cloud-Friendly Architecture with stateless flows easier to scale and deploy in containers
  - Improved Real-Time Processing capabilities for streaming data from IoT devices and AI applications

- MiNiFi: Apache MiNiFi is a sub project of Apache Apache NiFi meant to collect data and process data on the edge. Java (heavier) and C++ (lighter) flavours are provided. Both are however too large to be executed on a low power, microcontroller based far edge device.
- StarlingX: StarlingX is an edge cloud infrastructure targeting security, ultra-low latency, and extremely high service uptime which are requirements from the industrial Internet of Things (IoT). The underlying hardware layer is expected to run Yocto Linux, whereas scalability and orchestration are managed by Kubernetes and OpenStack frameworks, making StarlingX an heavy player.
- OpenNebula: OpenNebula is an open-source framework made to create multi-provider hybrid & edge clouds. It focuses on the virtual infrastructure layer and while deployment of containers and microVMs, it does not address the data processing layer.
- EdgeXFoundry: EdgeXFoundry focuses on IoT related use cases. It abstracts IoT protocols (sensors, actuators and others) and provides device management (administration and maintenance of IoT devices deployed on the field) capabilities. While there are still developments on-going, the number of tested devices and protocol adapters is relatively limited.

Based on this thorough analysis, two main options have been used within the SEDIMARK project:

- Mage AI, with the identified need to provide an additional customised user interface to ease management and configuration of pipeline.
- NiFi, deployed as part of the water use case to handle data driven orchestration of the data service processing flows.

3.3 WebAssembly on MCU

WebAssembly (Wasm) is a binary instruction format that acts as a portable compilation target for high-level programming languages, enabling efficient execution across a wide range of platforms, including web browsers and embedded systems. When applied to microcontroller units (MCUs) WebAssembly unlocks new opportunities for modular, portable, and secure edge computing because it allows to upload some “user code” to be run and controlled by the validated firmware. The user code can then be less secured and robust and changed frequently while the firmware is kept unchanged, robust and fully validated. Using WebAssembly for this task allows for safe and sandboxed execution of algorithms, simplifies updates and integration across different MCU-based devices, and promotes code reuse across projects. This makes it a compelling solution for scalable, low-power edge computing in sensor networks.

We have designed and tested a firmware architecture to support such a scheme. It is depicted in Figure 3.

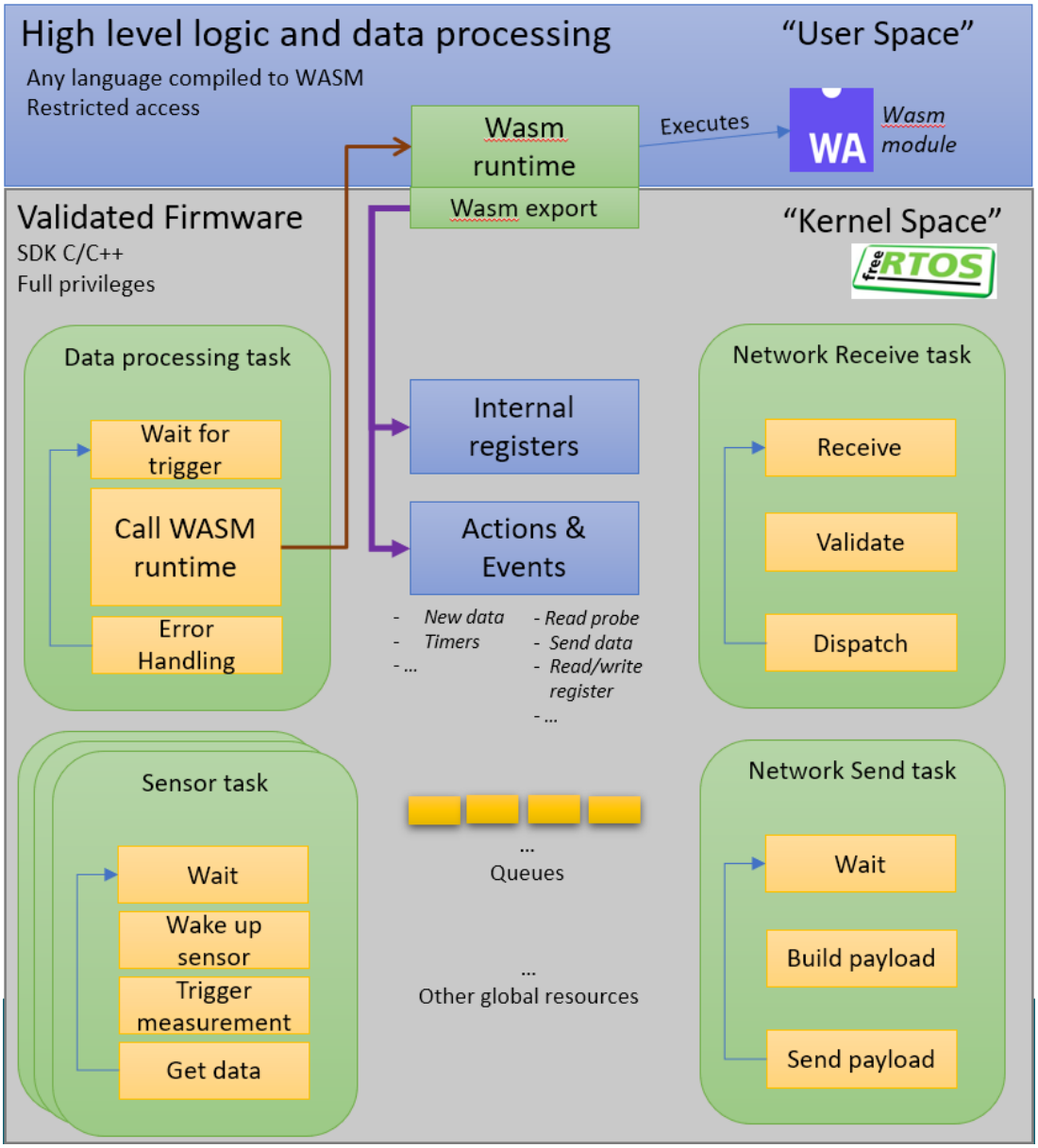

Figure 3: The architecture of the EdgeSpot software architecture to support some user scripts in WASM.

3.4 Fine timestamping on the edge

In many data-driven applications, such as MCU-based monitoring systems, transmitting data with a precise timestamp is crucial. Accurate timestamps ensure that events can be properly sequenced, correlated, and analysed (especially when data is collected from multiple sources). Without precise timing, it becomes difficult to synchronize actions, detect anomalies, or maintain data integrity. To achieve this level of precision, a calibrated Real-Time Clock (RTC) is essential. The RTC maintains the system's internal clock and provides consistent, reliable timekeeping. However, RTCs can drift over time due to various factors, including manufacturing tolerances and environmental influences, particularly temperature. Changes in external temperature can affect the oscillator inside the RTC, causing it to run slightly faster or slower. This drift, if left uncorrected, results in increasingly inaccurate timestamps.

For example, a one-year outdoor deployment of a STM32 L4 based IoT box revealed a drift of approximately 15 minutes. Calibration helps compensate for these variations, aligning the RTC with actual time standards and ensuring that data is logged and transmitted with high temporal accuracy.

EGM has developed a *Low-Power Pulse Counter* from its EdgeSpot platform. It is designed to detect and count pulses at one-minute intervals. The primary goal of this sensor is to estimate flow rate by counting pulses each minute. The device is made from the minimal design of the EdgeSpot hardware platform, paired with a LoRa-E5 module, a LoRaWAN antenna, and a high-capacity battery. To minimize power consumption, pulse counting is synchronized using the MCU's RTC. Every minute, the RTC triggers an alarm indicating the end of the counting period, prompting the system to store the pulse data. If pulses have been detected during a 15-minute window, a packet containing 15 minutes of pulse data, along with a timestamp, is transmitted to the cloud via LoRaWAN. The accuracy of the *Low-Power Pulse Counter* depends on the precision of the RTC. Although the RTC can be synchronized with the LoRaWAN server to maintain accurate timing, this operation results in a temporary increase in power consumption and introduces a dependency on network connectivity, both of which we aim to minimize.

A study was conducted on the temperature-induced drift of the RTC in EdgeSpot. The EdgeSpot is equipped with an HTU21D sensor, which provides internal temperature measurements of the board. During the study, the EdgeSpot's timestamps were recorded alongside temperature data and transmitted in real time to a computer via UART serial communication. This allowed a direct comparison with the computer's reference time.

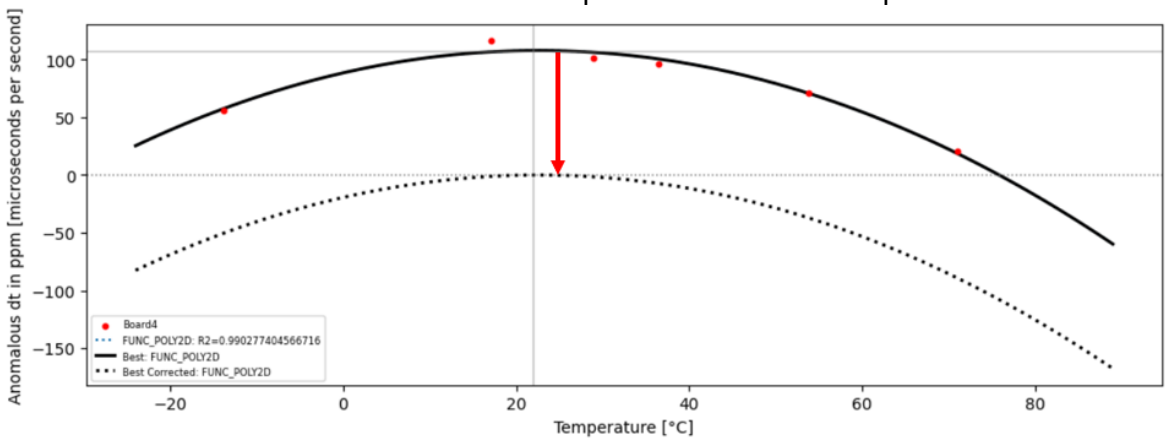

Figure 4: Time drift per minute vs temperature

Several tests were conducted at stable, controlled temperatures to estimate the time drift and generate a curve resembling the temperature-frequency characteristic of a crystal oscillator. In an RTC, the crystal oscillator serves as the precise timekeeping element, generating a consistent frequency (typically 32.768 kHz) to count seconds accurately. However, the frequency of the quartz crystals is sensitive to temperature changes, exhibiting a parabolic drift centred around their turnover point usually near 25°C. As temperature deviates from this point, the oscillator's frequency shifts, causing time to drift in the RTC.

Using the measured data, we fitted a quadratic model which allows us to calculate a temperature-dependent offset to correct the RTC timestamp in real time. The quadratic model provides a derivative of the bias as a function of time (in ppm or $\mu s.s^{-1}$):

$$\frac{dB}{dt} = a + bT + cT^2 \qquad \text{(eq.1)}$$

Where a, b and c are the parameters of the model, T is the temperature of the oscillator and B is the estimate of the bias between the RTC time and the real time t.

Since the measurement of the temperature is provided with a given precision by the sensor, we have also derived uncertainty on the computed value of the bias. So, the estimated bias value can be used to fix the value given by the RTC and the uncertainty can be used as a trigger to initiate a network query to reset the internal clock.

The algorithm for computing the bias is based on the numerical integration of this model, where a measurement of the temperature is provided at regular intervals (typically 1 hour).

We applied this algorithm in an EdgeSpot deployed in an open field. It was thus exposed to the sun, wind and some cold nights (Figure 5).

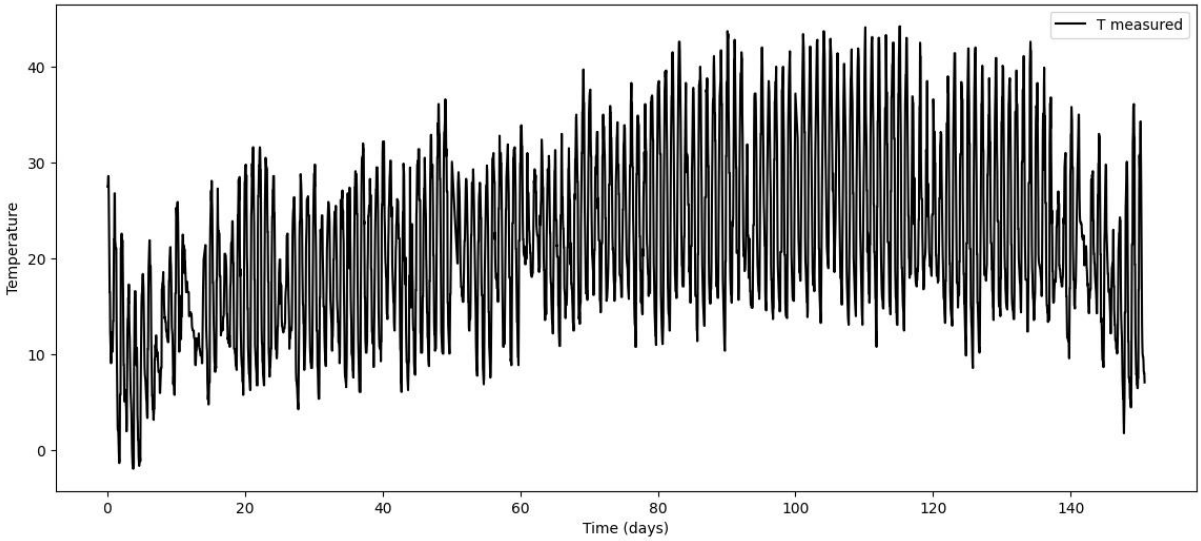

Figure 5: Temperature of the EdgeSpot deployed in a field for RTC bias algorithm testing.

The Figure 6 show the bias due to temperature fluctuations computed during the experiment (integration of equation (1) without *a* term. Additionally (not shown on picture), there is a continuous drift of 9.0 sec/day. One can see that, without temperature correction, the bias can fluctuate within a range of few seconds.

The grey area, on the Figure 6, represents the uncertainty of this correction. It equals 0.05 seconds after 150 days of experiment. This shows that using this simple temperature measurement and edge computing strategy, that consumes a very low power, we are thus able to maintain a very good time keeping accuracy, suitable for autonomous IoT operations. This contributes to a power-saving strategy, as analysed in the following section.

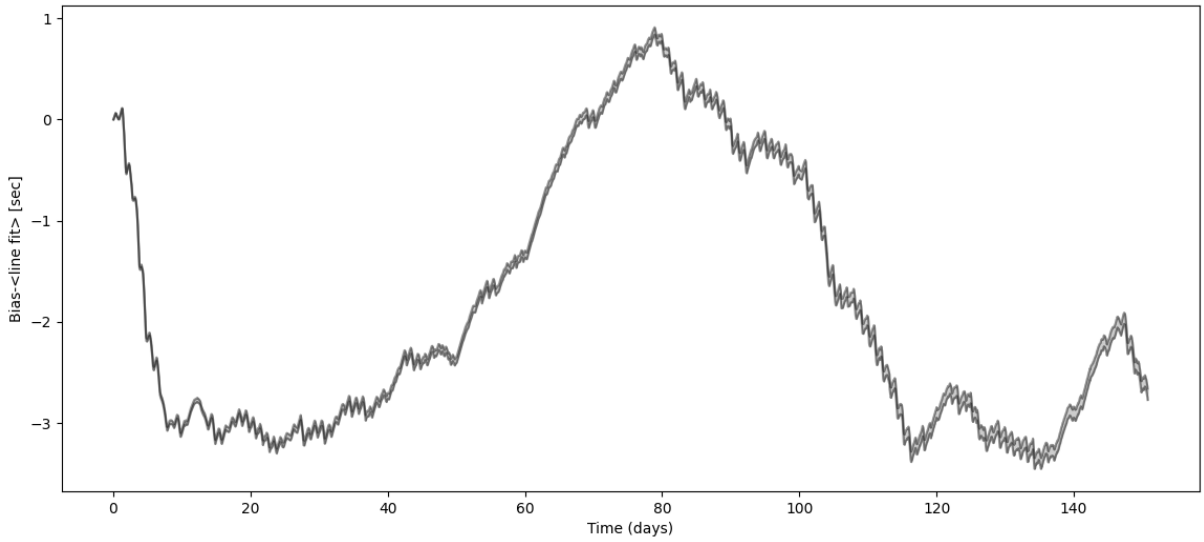

Figure 6: The "temperature part" of the bias, with the computed uncertainty.

3.5 Energy optimisation on the edge

In the case of the Low-Power Pulse Counter, we analysed the device's power consumption with the goal of enabling operation for at least 4 years. To achieve this, the sensor's firmware was carefully designed to minimize energy usage. The STM32L4 microcontroller can operate in a low-power mode, where most peripherals are disabled except for RTC. In this mode, comparable to a sleep state, the MCU halts execution and remains inactive until it is awakened, typically by the RTC. While in this state, the Pulse Counter consumes approximately 3µA. The sensor is only awakened to count pulses via a fast interrupt and to package data once per minute. Pulse detection raises the consumption to 0.1mA during several milliseconds. The device transmits the data to the server only if there is enough information to justify a transmission (every 15 minutes if pulses were detected).

The RTC is responsible for waking up the MCU at one-minute intervals to count pulses. When active, the MCU consumes at least 1mA, although this value may fluctuate depending on processing load and peripheral activity. The most energy-intensive operations involve the LoRaWAN module. For example, establishing a connection with the LoRaWAN server (a Join request), which must be performed at least once during boot, can cause a current spike of up to 80mA for about one second depending on the signal quality. A similar spike occurs during RTC time synchronization, which is performed once per week to prevent significant clock drift. We selected a weekly synchronization interval to minimize the impact on average power consumption and consequently, battery life. More frequent corrections could drastically reduce battery longevity. For example, performing a daily synchronization (lasting up to 20 seconds in poor signal condition) would raise average current draw to 21.5 µA, while weekly correction maintains it at 5.6 µA (Table 1).

Table 1: Impact of correction periodicity over battery life

Number of corrections	Average consumption	Battery life
Once per day	21.5 µA	100 years
Every other day	12.2 µA	176 years

Number of corrections	Average consumption	Battery life
Once per week	5.6 µA	383 years
Once per day	21.5 µA	100 years
Number of corrections	Average consumption	Battery life

However, when accounting for pulse detection and data transmission, the actual battery life is more realistically estimated between 4 to 10 years. As previously mentioned, enabling pulse detection increases the current consumption to approximately 0.1 mA. According to STMicroelectronics specifications, the MCU requires a maximum of 13.3 µs to wake from low-power mode. Following this, it executes processes to count and store pulses, which we have experimentally determined to take approximately 5 ms. During data transmission, the device consumes around 20 mA for at least 20 seconds under poor network conditions. Table 2 provides the computation of the average current consumption.

Table 2: Average current consumption depending on number of detected pulses

Number of pulses per minute	Average consumption	Battery life
400	443.2 µA	4.9 years
150	441.3 µA	4.9 years
50	440.5 µA	4.9 years

As shown, the number of detected pulses has a minimal impact on battery life. The primary factor influencing power consumption is data transmission, which occurs every 15 minutes, but only if pulses have been detected during that interval. Therefore, it is network communication that significantly affects battery longevity and should be optimized wherever possible.

To further reduce the frequency of energy-intensive time synchronization, the RTC calibration module offers a promising solution. By leveraging temperature data to estimate and correct drift locally, the system can maintain accurate timing with fewer network interactions. In comparison, the energy required for a temperature measurement is negligible. The HTU21D sensor, for instance, consumed just 0.02 µA in idle mode and up to 450 µA during a brief measurement. Integrating local time correction based on temperature could significantly improve pulse timing accuracy while reducing overall power consumption.

4 Framework-agnostic ML model description

Interoperability of neural network models between frameworks is a key development area within the ML research community. A plethora of frameworks exist for defining and training neural network models, with PyTorch [3], Tensorflow [4], and JAX [5] being the most popular from a research and development perspective. There are two broad modes of framework interoperability. On the one hand, a user might wish to deploy a model defined and trained in a framework optimized for inference at scale. On the other hand, it might be preferable to distribute a model definition in a common format that enables continued training within a framework of choice. This is especially the case within distributed or federated learning, where the sharing of raw Python code can present a security risk.

In general code written using the syntax and abstractions of one framework cannot be easily ported to another framework. As shown in [6], there are many converters between individual frameworks, but still the picture is incomplete, since there are many cases where no converter exists between two frameworks (i.e. between Theano [7] and caffe2 [8]). Additionally, there can be converters from i.e. framework 1 to framework 2, but no converters for the opposite conversion from framework 2 to framework 1, as in the case of ONNX to torch using onnx2torch, but no converter from torch to ONNX. Given the rapid pace of development, maintaining converters is a problem, and many frameworks may lack equivalent operators, and thus they will have to be re-implemented by the converter developer [9]. Small differences in the implementation of neural network components between frameworks might also result in differing model behaviour when models are ported from one framework to another, while it is noted by [9] that these converters can often introduce subtle bugs and security problems.

In the case of model deployment and inference, most of the popular frameworks contain a module or method for porting code to Open Neural Network Exchange (ONNX) [10], a common intermediary depiction which represents the network as a language agnostic graph, that can then be compiled and deployed in one of several inference run-times. SEDIMARK will allow for the export of models to ONNX format for the purpose of inference. However, ONNX does not fully support the retraining of models.

As seen in the table from [6], most conversions between frameworks are based on the MMdNN project [11], which is an attempt to define a “Universal Converter” for deep learning models to allow both inference and re-training of ML models across different frameworks. MMdNN converts model formats to an “Intermediate Representation”, and from that, converts the model to the target platform format. However, MMdNN is only focusing on a subset of deep learning models and currently is not maintained on GitHub, with its latest commit more than 3 years ago.

There are ongoing attempts to remedy this problem in a more holistic manner - for instance the commercial effort Ivy [12] from unify.ai aims to offer code transpiration between frameworks, though requires an Application Programming Interface (API) key for use and did not appear to work out of the box when tested. There is currently not fully inclusive, open-source solution to the problem of transferring models between frameworks for continued training.

A recent update to the popular framework Keras [13] will support code written in Keras being run with either Jax, Pytorch or Tensorflow as a backend. Both inference and continued model training are supported. Though the Keras API imposes limitations on the class of models that can be defined within it, for most general use-cases it proves sufficient.

SEDIMARK builds upon the newly released Keras Core Python package to offer a degree of interoperability for the purpose of further training models in distributed settings (see Figure 7). Models defined in this format can be seamlessly exported to either JAX, Tensorflow or Pytorch, though the reverse is not true. While allowing for a choice of back-end and providing a common format for participants in the SEDIMARK distributed learning ecosystem, this does effectively restrict users to the Keras syntax and abstractions when defining their models.

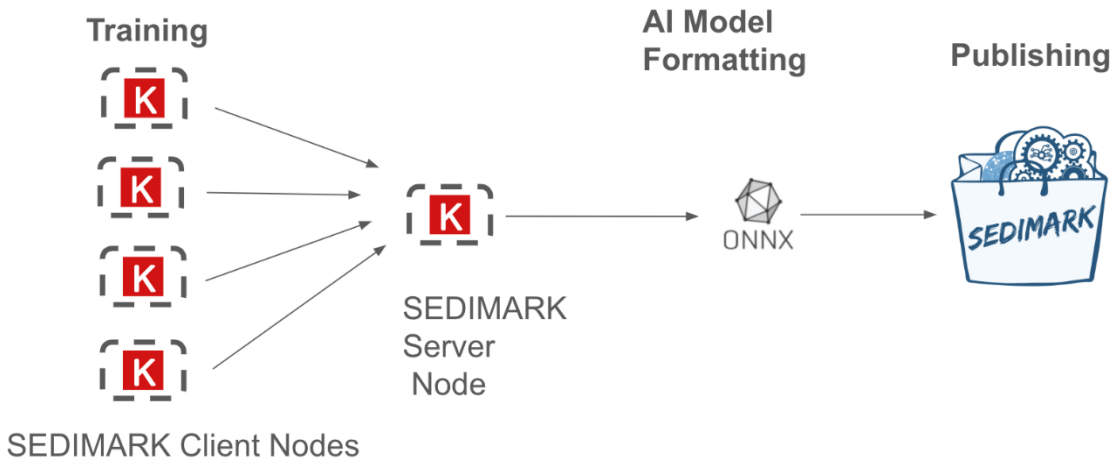

Figure 7: Model formatting process within SEDIMARK.

Going forward SEDIMARK explored alternatives, such as the ONNX training-runtime, which is still undergoing development. Keras-core has already been implemented as the model format within both SEDIMARK distributed training components (Fleviden and deFlight), and performance has been tested with nodes running each of the three underlying frameworks it supports.

5 Certification Services

Assets advertised through offerings made available through the Marketplace vary in terms of quality and performance. Although, it is mandatory to pass a minimum set of requirements for them to be minimally viable and exchangeable assets. This involves checking for compliance:

- Assets with standards specified by the Marketplace.
- Connectors with the minimal set of operations.
- Use of Assets in accordance with license or policy restrictions.

Figure 8 illustrates the categorization of certification services for a particular type of Asset.

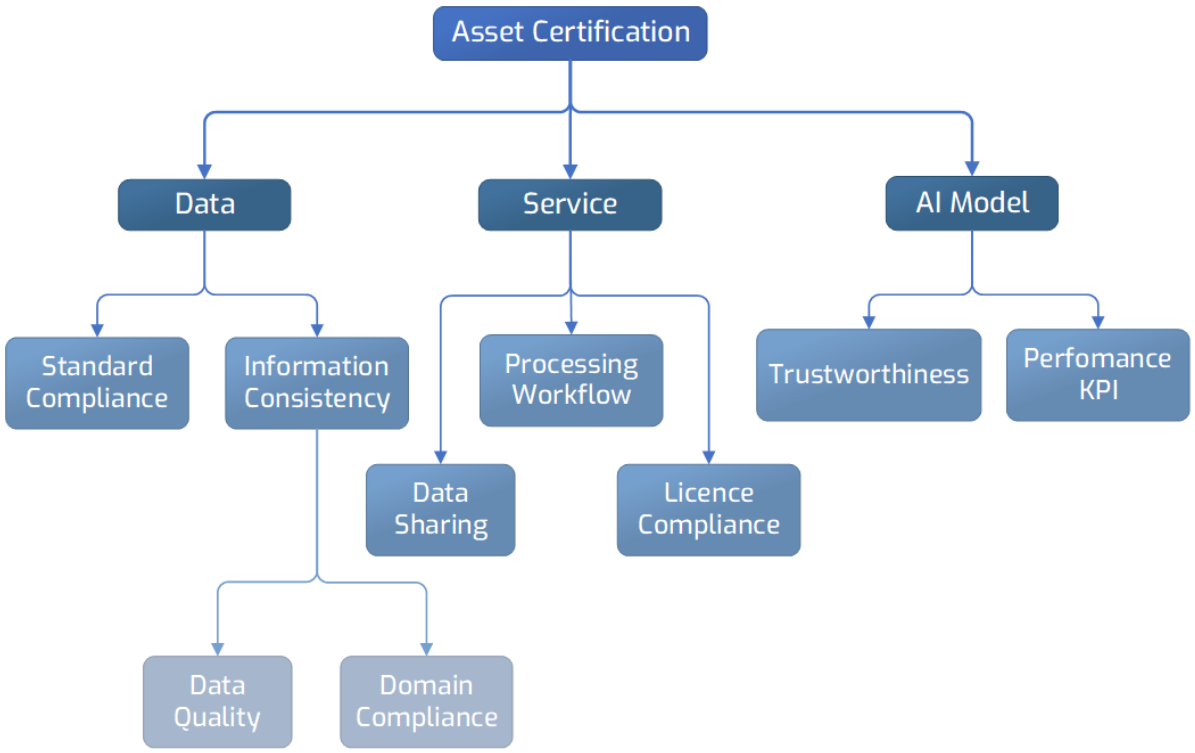

Figure 8: Taxonomy of Certification services for Assets within SEDIMARK.

The services for certification are contained within a dedicate suite service exposed via RESTful interface [14].

5.1 Data Assets

For data assets, their formatting, annotation, and enrichment need to comply with the information model standards specified by SEDIMARK. Information regarding both metadata and data needs to be assessed for consistency. This involves checks for quality and compliance within domain-specific parameters. With regards to information consistency, a subset of generic data quality metrics defined in SEDIMARK deliverable D3.2 are used, as well as metric governed by domain-specific restrictions or ranges.

Data Assets are certified based on Standard Conformance and Information Consistency

Standard Conformance

The SEDIMARK Marketplace allows Participants to provide Data Assets at different conformance, although SEDIMARK recommends that Participants to provide the highest level to maximize its streamlining with Consumer data and AI workflows. The different levels are as follows:

- L1 Conformance = Proprietary
- L3 Conformance = Proprietary + NGSI-LD Mapping (Entity, Specific Properties, Generic Properties)
- L3 Conformance = NGSI-LD + JSON-LD
- L4 Conformance = NGSI-LD + JSON-LD + Smart Data Model

Level 1 is the lowest level of conformance, in which the Data Asset format is purely proprietary. Most Data Assets in this format will not be generated through the generic workflows supported by the SEDIMARK Orchestrator and will be provisioned as is from the original data source.

Level 2 involves providing a mapping configuration whereby entities in the proprietary Data Asset are mapped to NGSI-LD model concepts.

Level 3 ensure Data Asset are modelled in NGSI-LD and serialised in the JSON-LD format.

Level 4 is the highest level of conformance, whereby a Data Asset complies also with a standard that provides a thematic vocabulary in relation to domains of interest, entity types and special properties such as units of measurement. The reference theme vocabulary that is used in SEDIMARK is the Smart Data Model standard which is detailed in SEDIMARK deliverable D3.4 [15].

Information Consistency

Consistency is determined by the quality of the Data Asset, and its compliance with domain-specific boundaries. This stage of certification is secondary to the Standard Conformance stage.

Data Assets also have metadata that captures data quality metrics, mainly based on those identified in SEDIMARK deliverable D3.2. This information is generated as a by-product of the Asset generation process within the Orchestrator and represented as a DataQualityAssessment Entity in NGSI-LD, which is also a model originating from the Smart Data Model standard. This will be retrieved by the certification suite when a Data Asset is being assessed.

In relation to domain-specific assessment, Data Assets covering a particular theme require that values for thematic properties are within the boundaries or limits relative to that domain. For examples, temperature values in a Weather-based use case will have different range of values, as opposed to the temperature values captured from a Vehicle's engine.

5.2 Service Assets

For Service Assets, the set of operations available through their corresponding interfaces are checked to ensure that they conform to the standards adopted for data sharing interfaces. For example, NGSI-LD is currently the main interface for Data Sharing, therefore the technical specification on validating NGSI-LD platforms is used as a reference. For that purpose, contributions have been made to the on-going development at ETSI of a test specification [16] and a testing environment [17] to ensure that various NGSI-LD implementations comply with the ETSI NGSI-LD API specification. The tool is developed with Robot Framework, which is

an open-source test automation framework. Written in Python, Robot Framework uses versatile syntax that helps in creating readable and maintainable test cases. It also provides embedded servers for HTTP and Mock requests.

The Test Suite focuses on verifying the compliance of different key functionalities defined in the NGSI-LD specification such as:

- Context Information Provision: which tests the creation, update and deletion of context information
- Context Information Consumption: which tests the retrieval and query of context information
- Context Information Subscription: which tests subscribing to context information changes and receiving notifications
- Context Source Registration: which tests registration and discovery of context information
- Distributed Operations: which tests interoperability between different implementations.

The Test Suite is organized into several modules corresponding to each key functionality described above. In addition to these, it includes modules for configuration files, reusable test data, libraries, and scripts used to generate documentation of the test cases. This modular organization ensures that the Test Suite is maintainable, easy to navigate, and scalable as the NGSI-LD specification evolves.

Each test case begins by creating the necessary test entities in a Suite Setup. After execution, the Test Suite performs a clean-up in Suite Teardown. The test case includes clear documentation and tags referencing the NGSI-LD specification section. The Figure 9 shows the template used for defining test cases.

Figure 10 shows an example of a test case execution from the NGSI-LD test suite.

To execute the Test Suite, the environment must be set up by downloading a configuration file and executing the scripts inside. These scripts install different software requirements and create a Python Virtual Environment where the test cases can be launched.

The Test Suite is available on GitLab [18] and it includes comprehensive documentation covering installation, configuration, and usage instructions.

```
*** Settings ***
Documentation       {Describe the behavior that is being checked by this Test Case}

Resource            ${EXECDIR}/resources/any/necessary/resource/file

# For both setup and teardown steps, try as possible to use Keyword names already declared in doc/ana
Suite Setup         {An optional setup step to create data necessary for the Test Case}
Suite Teardown      {An optional teardown step to delete data created in the setup step}
Test Template       {If the Test Case uses permutations, the name of the Keyword that is called for e


*** Variables ***
${my_variable}=             my_variable


*** Test Cases ***    PARAM_1    PARAM_2
XXX_YY_01 Purpose of the first permutation
    [Tags]    {resource_request_reference}    {section_reference_1}    {section_reference_2}
    param_1_value    param_2_value
XXX_YY_02 Purpose of the second permutation
    [Tags]    {resource_request_reference}    {section_reference_1}    {section_reference_2}
    param_1_value    param_2_value


*** Keywords ***
{Keyword name describing what the Test Case is doing}
    [Documentation]    {Not sure this documentation brings something?}
    [Arguments]    ${param_1}    ${param_2}

    # Call operation that is being tested, passing any needed argument

    # Perform checks on the response that has been received from the operation

    # Optionally call another endpoint to do extra checks (e.g. check an entity has really been creat

# Add there keywords for setup and teardown steps
# Setup step typically creates data necessary for the Test Case and set some variables that will be u
# using Set Suite Variable or Set Test Variable keywords
# Teardown step must delete any data that has been created in setup step
```

Figure 9: Test Case Template

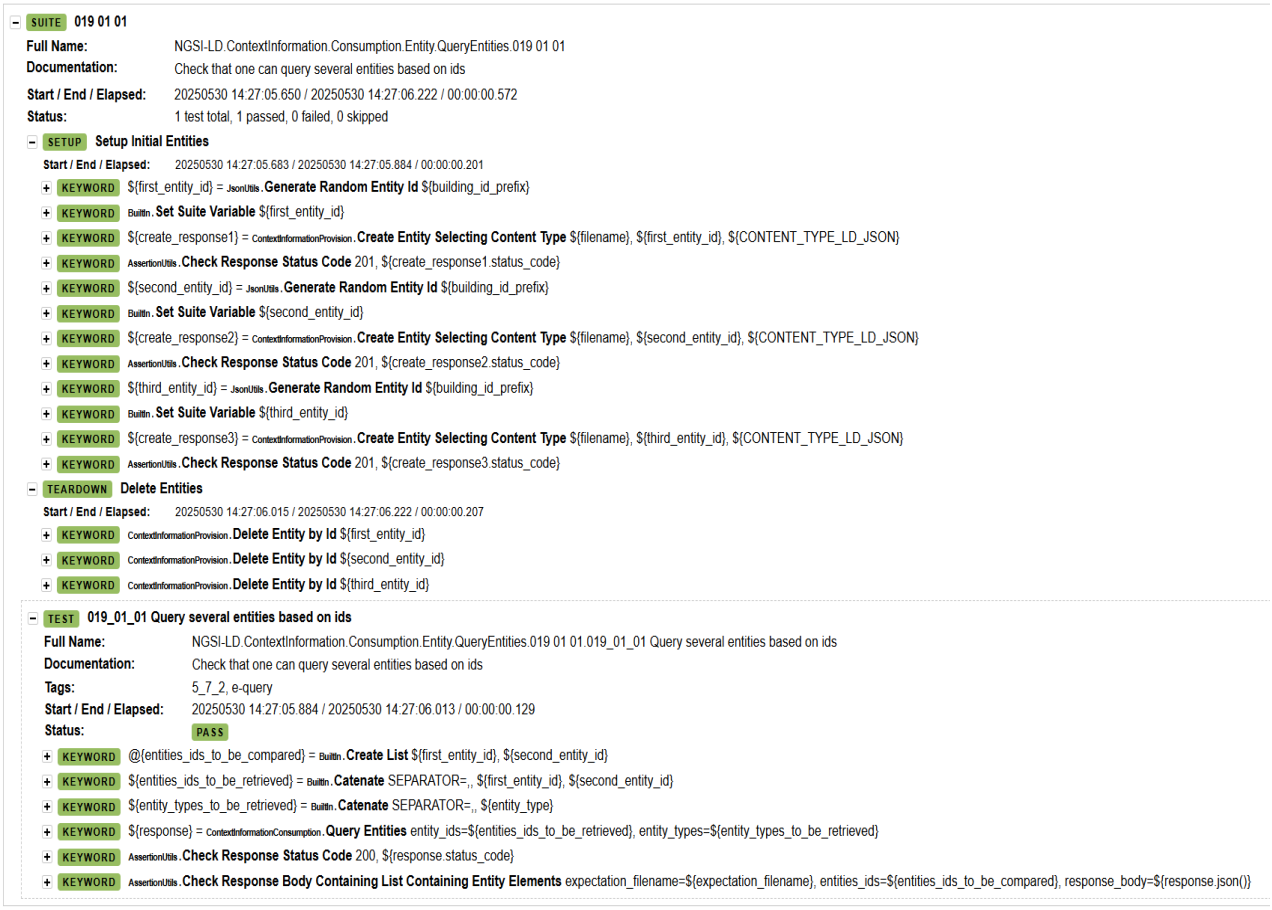

Figure 10: Execution log of a query test case in the NGSI-LD test suite

The NGSI-LD Test Suite generates a detailed HTML report after execution, summarizing the outcome of each test case. The Figure 11 shows a sample report highlighting:

- The total number of test cases executed
- How many passed, failed, or were skipped
- Execution time per test group
- A breakdown of results grouped by specification tag (e.g., 4_5_5), referring to the NGSI-LD specification chapters.

Summary Information

Status:	**2 tests failed**
Start Time:	20250306 14:14:02.413
End Time:	20250306 14:19:36.487
Elapsed Time:	00:05:34.074
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	566	564	2	0	00:04:25	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
4_10	16	16	0	0	00:00:01	
4_15	6	6	0	0	00:00:00	
4_16	3	3	0	0	00:00:01	
4_18	11	11	0	0	00:00:02	
4_19	6	6	0	0	00:00:00	
4_5_18	19	19	0	0	00:00:07	
4_5_19	8	8	0	0	00:00:01	
4_5_20	5	5	0	0	00:00:01	
4_5_23	22	22	0	0	00:00:10	
4_5_24	6	6	0	0	00:00:01	
4_5_5	21	19	2	0	00:00:06	
4_5_7	3	3	0	0	00:00:00	
4_5_9	3	3	0	0	00:00:00	
4_7	10	10	0	0	00:00:01	
5_2_2	4	4	0	0	00:00:01	

Figure 11: Summary view of NGSI-LD Test Suite results

We also contributed to the Plug Test event organized in Sophia Antipolis, Nice by ETSI, which brought together multiple teams and implementations of the NGSI-LD standard. This event served as a practical interoperability test, where different NGSI-LD-compatible platforms and components were integrated and tested in real-time scenarios. The main objective was to verify that these implementations can exchange data, perform context operations (such as queries and updates), and comply with the NGSI-LD specification. Interoperability testing plays a critical role in ensuring that all implementations behave consistently and predictably when communicating with each other. The following Figure 12 illustrates a representative configuration used during the Plug Test.

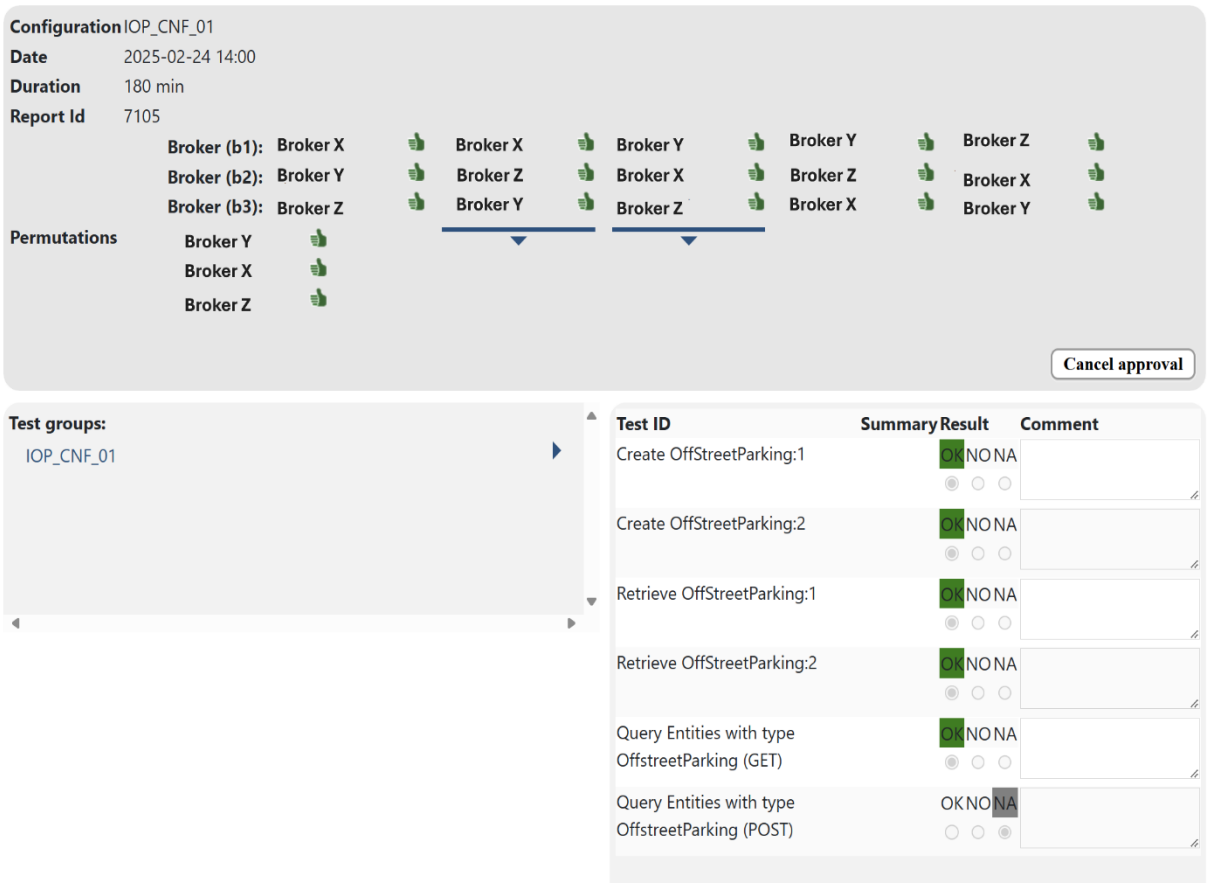

Figure 12: Example of configuration from the NGSI-LD Plug Test event

5.3 AI Model Assets

In the context of the SEDIMARK platform, AI model assets are critical functional components that deliver predictive insights across multiple domains, including energy forecasting and customer churn. As such, their inclusion in the SEDIMARK Marketplace demands rigorous certification processes to ensure that they meet operational performance standards and exhibit trustworthy behaviour. The certification of AI model assets is two-fold; verification of performance-related KPIs, and evaluation of the trustworthiness of the AI model in alignment with international standards and ethical AI guidelines.

AI models must demonstrate acceptable levels of accuracy and reliability in the tasks for which they are certified. As part of their onboarding and validation into the SEDIMARK ecosystem, models are required to include a performance profile derived from test datasets (either synthetic or real, if available and permitted). The following quantitative KPIs are assessed:

Classification Models (e.g., for churn prediction):

- Accuracy: Percentage of correct predictions over total predictions.
- Precision/Recall/F1-score: Measures of relevance and completeness.
- Confusion Matrix: Breakdown of true/false positives/negatives.
- Area Under ROC Curve (AUC): Sensitivity-specificity trade-off.

AI/ML models Regression Models (e.g., for energy forecasting):

- Mean Absolute Error (MAE): Average magnitude of errors.

- Root Mean Square Error (RMSE): Penalizes larger errors.
- R^2 (Coefficient of Determination): How well data fits the model.

These metrics are expected to meet minimum quality thresholds depending on the domain of use and training context. For example, a churn model may be expected to achieve at least 85% F1-score on test datasets, while an energy forecasting model may require an RMSE below a sector-defined baseline.

In practice, performance metrics are collected and attached as metadata to each AI model asset within the Toolbox registry or Marketplace catalogue, supporting transparency and reuse.

Furthermore, AI model assets promote the inclusion of trustworthy AI practices, which are necessary to ensure ethical use, user confidence, and regulatory compliance. The certification process therefore includes a trust assessment, inspired by frameworks from NIST, OECD, and ISO/IEC 24028. Key pillars of AI trustworthiness include:

- Validity and Reliability: The model produces consistent and correct outputs under varying conditions.
- Transparency & Provenance: Certification requires traceability of the training dataset origin, data quality assessment, and clear documentation of model structure and hyperparameters.
- Fairness and Bias Mitigation: Providers must submit fairness testing results across protected variables (e.g., gender, age) and describe any bias mitigation techniques applied.
- Security and Resilience: Models must be tested against adversarial vulnerabilities (e.g., via robustness testing) and demonstrate fallback mechanisms.
- Privacy Compliance: Models trained on personal data must document anonymization techniques and comply with GDPR obligations. Edge deployments must ensure local processing where necessary.

This certification framework ensures that only responsible, performant, and transparent AI models are made available as assets in the SEDIMARK platform. It fosters user confidence, supports the reusability and auditability of AI components, and aligns with emerging EU AI regulatory frameworks such as the AI Act.

5.4 Other Artefacts

The certification suite also contains the validation of Offerings as detailed in SEDIMARK deliverable D3.4. The assessment of Offering is based on graph-based data representation is SHACL (Shapes Constraint Language) [19] which is a W3C specification aimed at validating the compliance of a graph by checking its “shape” comply to the expectation.

6 Conclusions

This document details the architecture, tools, and processes developed to enable intelligent and distributed data processing within the SEDIMARK platform. The work provides the foundational components for managing the entire lifecycle of data and AI assets, from their creation at the edge to their certification and exchange in the marketplace.

A contribution has been the evaluation of edge data processing approaches for low power (far edge) devices. By investigating and implementing solutions for resource-constrained environments, the project has demonstrated the feasibility of deploying advanced functionalities on low-power devices. The use of WebAssembly (WASM) on MCUs allows for secure and flexible execution of user code but lacks developer support in some MCUs while new frameworks such as μpython seem to offer wider appropriation. The development of a RTC calibration algorithm ensures precise data timestamping by compensating for environmental factors like temperature. These innovations, combined with energy optimization techniques, enable long-term, autonomous operation of edge devices, which is critical for the project's use cases. The analysis and selection of orchestration tools like Mage.ai and Apache NiFi provide the necessary mechanisms to manage these distributed data flows between the edge and the cloud.

The establishment of a multi-faceted Certification Service represents a cornerstone of the SEDIMARK platform's commitment to trust and quality. This service provides a systematic methodology for validating all marketplace assets:

- Data Assets are certified against multiple conformance levels, ensuring they comply with the NGSI-LD standard and domain-specific information models.
- Service Assets are validated for API compliance, leveraging and contributing to standardized test suites like the ETSI NGSI-LD Test Suite to guarantee interoperability.
- AI Model Assets undergo a two-fold assessment, verifying not only their performance through quantitative KPIs but also their trustworthiness against key principles such as fairness, transparency, and security.

Collectively, these advancements provide the essential technical infrastructure for a truly decentralized and intelligent data marketplace. By enabling interoperable data processing at the edge and ensuring all assets are certified for quality and compliance, this work lays the groundwork for a trusted ecosystem where data and AI models can be shared and monetized securely and efficiently.

7 References

[1] SEDIMARK, “D2.1 Use cases definition and initial requirement analysis,” 2023, June.

[2] SEDIMARK, “D2.3 SEDIMARK Architecture and Interfaces. Final version,” 2024, September.

[3] [Online]. Available: https://pytorch.org/.

[4] [Online]. Available: https://www.tensorflow.org/.

[5] [Online]. Available: https://jax.readthedocs.io/en/latest/.

[6] ysh329, “Deep learning model convertors,” [Online]. Available: https://github.com/ysh329/deep-learning-model-convertor.

[7] [Online]. Available: https://github.com/Theano/Theano.

[8] [Online]. Available: https://caffe2.ai.

[9] Z. M. G. C. K. L. T. X. L. &. C. C. Deng, “Differential Testing of Cross Deep Learning Framework {APIs}: Revealing Inconsistencies and Vulnerabilities,” in *32nd USENIX Security Symposium*, 2023.

[10] [Online]. Available: https://onnx.ai/.

[11] Y. C. C. Z. R. Q. T. J. X. L. H. &. Y. M. Liu, “Enhancing the interoperability between deep learning frameworks by model conversion. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering,” in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020.

[12] D. P. F. F. F. J. S. &. C. R. I. Lenton, “Templated deep learning for inter-framework portability,” [Online]. Available: https://doi.org/10.48550/arXiv.2102.02886.

[13] [Online]. Available: https://keras.io/.

[14] SEDIMARK, “Certification suite source code,” [Online]. Available: https://github.com/Sedimark/certification-suite.

[15] SEDIMARK, “D3.4 - Enabling tools for data interoperability, distributed data storage and training distributed AI models. Final version,” 2025.

[16] ETSI, “ETSI GS CIM 014 V3.1.1 - NGSI-LD Test Suite,” 07 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/014/03.01.01_60/gs_CIM014v030101p.pdf .

[17] ETSI, “NGSI-LD Test Suite,” [Online]. Available: https://forge.etsi.org/rep/cim/ngsi-ld-test-suite.

[18] ETSI, “NGSI-LD Test Suite,” [Online]. Available: https://forge.etsi.org/rep/cim/ngsi-ld-test-suite.

[19] W3C, “Shapes Constraint Language (SHACL),” [Online]. Available: https://www.w3.org/TR/shacl/.