TaRDIS

# D7.3: Report on development of the use cases with TaRDIS toolbox

Revision: v.1

| | |
|---|---|
| **Work package** | WP 7 |
| **Task** | Task 7.3, 7.4, 7.5 |
| **Due date** | 31/12/2025 (M36) |
| **Submission date** | 05/02/2026 |
| **Deliverable lead** | NKUA |
| **Version** | 1.0 |
| **Authors** | Dimitris Kogias, Sotiris Spantideas, Panagiotis Trakadas(NKUA), Miroslav Popovic, Lidjia Fodor, (UNS), Alceste Scalas (DTU), Luis Pisco (EDP), David Solans Noguero (TID), David Vazquez Enriquez (GMV), Helen-Aikaterini Leligkou, Panagiotis Karkazis (UniWA), Ping Hou (UOXF) |
| **Reviewers** | Carla Ferreira (NOVA), Carlos Coutinho (CMS) |
| **Abstract** | This document reports the evaluation of the four use cases developed with the TaRDIS toolbox, focusing on requirements fulfilment and KPI assessment (Task 7.3). Integration details and demonstrations are reported in D7.4, while final validation and mitigation actions are addressed in D7.5. |
| **Keywords** | KPIs, Requirements, TaRDIS, tools |



Grant Agreement No.: 101093006
Call: HORIZON-CL4-2022-DATA-01

Topic: HORIZON-CL4-2022-DATA-01-03
Type of action: HORIZON- RIA

## Document Revision History

| Version | Date | Description of change | List of contributor(s) |
|---|---|---|---|
| V0.1 | 20/09/2025 | Table of Content | NKUA |
| V0.2 | 20/11/2025 | 1st Input from Partners | NKUA, EDP, GMV, TID, UniWA |
| V0.3 | 28/11/2025 | 2nd input from Partners | NKUA, FTN, EDP, GMV, TID, UniWA |
| V0.4 | 15/12/2025 | Update from Partners and Section 4 | NKUA, EDP, GMV, TID, UniWA |
| V0.5 | 23/12/2025 | Update from GMV | GMV, NKUA |
| V0.6 | 24/01/2026 | Version for Internal Review | NOVA, CMS |
| V1 | 5/2/2026 | Ready for Submission | ALL |

Funded by the European Union

## Disclaimer

Funded by
the European Union

Funded by the European Union (TaRDIS, 101093006). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

## Copyright notice



| **Project funded by the European Commission in the Horizon Europe Programme** | | |
|---|---|---|
| **Nature of the deliverable:** | **R** | |
| **Dissemination Level** | | |
| **PU** | *Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)* | ✔ |
| **SEN** | *Sensitive, limited under the conditions of the Grant Agreement* | |
| **Classified R-UE/ EU-R** | *EU RESTRICTED under the Commission Decision No2015/ 444* | |
| **Classified C-UE/ EU-C** | *EU CONFIDENTIAL under the Commission Decision No2015/ 444* | |
| **Classified S-UE/ EU-S** | *EU SECRET under the Commission Decision No2015/ 444* | |

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.

# EXECUTIVE SUMMARY

This deliverable (D7.3) presents the **first consolidated evaluation of the TaRDIS toolbox**, assessing its effectiveness, performance, and applicability across four heterogeneous pilot use cases: **EDP, GMV, TID, and ACT**. The evaluation focuses on **requirements fulfilment** and **Key Performance Indicator (KPI) achievement**, following the methodology and baseline definitions established in earlier WP7 deliverables.

The evaluation is structured along three complementary layers:

1. **Use Case KPIs**, capturing scenario-specific functional and operational objectives;
2. **Baseline KPIs**, measuring improvements over reference implementations and expected baseline behaviour;
3. **Objective KPIs**, assessing project-level architectural, technological, and methodological goals.

Where execution-based validation was possible, results are supported by **quantitative measurements**, controlled experiments, and real-user deployments. Where access to operational environments was constrained—most notably in the ACT use case—evaluation was performed at **design and implementation level**, or explicitly marked as **Partial** or **Not Evaluated**, with a clear mitigation plan toward final validation in **Deliverable D7.5**.

**Key Evaluation Outcomes**

- **Use Case KPIs**
  The TaRDIS toolbox demonstrates strong suitability across diverse domains.
  - **EDP** and **GMV** validate the expressivity and robustness of TaRDIS for complex distributed coordination scenarios.
  - **TID** provides strong quantitative evidence of TaRDIS-enabled federated learning efficiency, including significant reductions in development effort, communication overhead, and resource usage, supported by a large-scale real-user field study.
  - **ACT** demonstrates the applicability of the TaRDIS toolbox in a real factory setting, targeting industrial swarm orchestration and adaptive manufacturing workflows. Evaluation of execution-dependent KPIs has been temporarily delayed due to limited availability of the factory environment during the current reporting period. This delay reflects operational constraints rather than technical limitations of the toolbox, and full execution-based validation is planned and scheduled for Deliverable D7.5.

- **Baseline KPIs**
  Where evaluated, TaRDIS-enabled solutions demonstrate **clear improvements over baseline approaches**, particularly in terms of scalability, memory efficiency, and predictable performance under increasing system size. Baseline KPIs dependent on long-term operation or live industrial workflows are transparently deferred to the final evaluation phase.
- **Objective KPIs**
  At project level, TaRDIS meets its **core design objectives**:
  - High expressivity of language primitives and event-driven abstractions;
  - Measurable reductions in development effort across use cases;
  - Strong efficiency gains for machine learning workloads where applicable;
  - Robust support for decentralised membership, storage, devices, and programming languages.

Objective KPIs that depend on large-scale deployment, middleware integration, or extended operational observation are clearly identified and scheduled for completion in D7.5.

**Overall:** The results presented in D7.3 confirm that **TaRDIS is a mature, effective, and flexible toolbox for engineering distributed swarm-based systems**, capable of addressing diverse industrial and research challenges. The evaluation approach is

transparent, reproducible, and explicitly update-ready, ensuring full traceability from requirements to measured outcomes.
D7.3 establishes a solid evidence base for the **final validation and consolidation activities** to be completed in **Deliverable D7.5**, following the full execution of all use case demonstrations documented in **D7.4**.
Finally, all pending or partially evaluated KPIs are associated with clearly identified external constraints or staged deployment plans, and do not indicate functional gaps in the TaRDIS toolbox.

Funded by the European Union

## Table of Contents



Funded by the European Union

# LIST OF FIGURES

Funded by the European Union

# LIST OF TABLES



Funded by the European Union

# ABBREVIATIONS

**ACT** ACTYX Use Case
**ADB** Android Debug Bridge
**API** Application Programming Interface
**CPU** Central Processing Unit
**DLT** Distributed Ledger Technology
**DoA** Description of Action
**DP** Differential Privacy
**FL** Federated Learning
**FLaaS** Federated Learning as a Service
**GUI** Graphical User Interface
**IoT** Internet of Things
**KPI** Key Performance Indicator
**KD** Knowledge Distillation
**ML** Machine Learning
**N/A** Not Applicable
**NNR** Neural Network Representation
**ODTS** Orbit Determination and Time Synchronisation
**RAM** Random Access Memory
**RNF** Non-Functional Requirement
**RF** Functional Requirement
**RSS** Resident Set Size
**SL** Split Learning
**SLO** Service Level Objective
**TLS** Transport Layer Security
**WP** Work Package
**WuW** Wake-up Word

Funded by the European Union

# 1. INTRODUCTION

## 1.1 Deliverable Structure and Scope

This deliverable (D7.3) presents the **consolidated evaluation** of the four TaRDIS use cases, with emphasis on the fulfilment of their requirements and the assessment of their Key Performance Indicators (KPIs). It examines how each use case deployed and utilised components of the TaRDIS toolbox, how KPIs were measured in practice, and the extent to which improvements over established baselines were achieved at the current stage of the project.
D7.3 represents the **first project-wide evaluation milestone**, capturing results obtained from completed demonstrations, controlled experiments, simulations, and structured qualitative assessments. Where execution in real operational environments was not possible within the reporting period, evaluation results are explicitly marked as Partial or Not Evaluated, with full validation planned for the final evaluation deliverable (D7.5).
The document is organised as follows: **Section 1** introduces the scope of the evaluation, its role within WP7, and the adopted evaluation methodology, while **Section 2** presents a detailed evaluation of each use case (ACT, EDP, GMV, and TID), including requirements compliance, KPI measurements, and observed performance characteristics. Then, **Section 3** synthesises the evaluation results across use cases, identifying common strengths, limitations, and their collective contribution to project-level objectives. Finally, **Section 4** summarises the main conclusions and outlines the implications for the final integration and validation activities to be carried out in Deliverables D7.4 and D7.5.

## 1.2 Role In WP7

Deliverable D7.3 constitutes a **central milestone within WP7**, which is responsible for validating the TaRDIS toolbox in realistic industrial and research settings. Earlier WP7 deliverables (D7.1 and D7.2) established baseline system behaviour, mapped the initial deployment of TaRDIS tools across the use cases, and defined the Key Performance Indicators (KPIs) and evaluation procedures to be applied.
The **use case demonstrations and the final configuration of the TaRDIS toolset** are documented in **Deliverable D7.4**, which focuses on the execution setup, demonstration workflows, and presentation of the integrated toolbox in each pilot environment. In contrast, **D7.3 concentrates on the evaluation of performance**, assessing requirement fulfilment and KPI achievement based on the evidence collected up to the current reporting period.
Building on this foundation, D7.3 provides the **first comprehensive assessment** of TaRDIS performance across all four pilot environments. Its role is twofold:

- **Use case–level validation:** D7.3 evaluates how tools and methods developed in WP3–WP6 behave under representative operating conditions, and whether they address the functional and performance requirements identified earlier in the project. This includes validation of individual toolbox components as well as their effectiveness when combined within concrete application scenarios.

- **Work package–level consolidation:** D7.3 aggregates and analyses evaluation results across all use cases, enabling a cross-cutting assessment of strengths, limitations, and common trends. These insights directly inform the remaining integration, optimisation, and validation activities, including the identification of mitigation actions.

Where evaluation could not be fully completed due to external constraints (e.g. limited access to operational industrial environments), **results are explicitly marked as Partial or**

**Not Evaluated**. The corresponding mitigation actions and final validation of these elements are planned and will be reported in **Deliverable D7.5**, which constitutes the final evaluation report of WP7.
Through this dual role, D7.3 acts as the **primary evidence bridge** between the research and development activities carried out in WP3–WP6 and the final validation outcomes of WP7, ensuring full traceability from requirements and design objectives to measured performance and documented KPI achievements.

## 1.3 Evaluation Methodology

The evaluation presented in this deliverable follows a **structured and uniform methodology**, designed to ensure transparency, consistency, and alignment with the overall TaRDIS validation framework. The methodology has been applied consistently across all use cases and comprises the following steps:

1. **Requirements Identification:** Each use case is evaluated against the requirements originally defined in WP2 and refined through WP7 activities and Deliverables D7.1 and D7.2. Requirements related to toolbox-internal components originating from WP3–WP6 are assessed within the corresponding work package deliverables; only use case–level requirements are reported in this document.
2. **KPI Framework:** Three categories of Key Performance Indicators (KPIs) are considered, as defined in D2.2, D7.1, and D7.2:
    - Use case–specific KPIs
    - Baseline KPIs
    - Project objective KPIs
3. **Data Collection:** Evaluation data is collected through a combination of automated measurements, use case–specific instrumentation, simulations, and structured qualitative input from partners. Qualitative feedback is gathered via dedicated questionnaires to complement quantitative metrics and capture implementation-level insights. Developer questionnaires are collected once per use case and consolidated at project level; consequently, KPI assessments relying on these inputs are intentionally conservative at this stage and prioritise consistency across heterogeneous scenarios.
4. **Baseline Comparison:** Measured results are compared against baselines established in Task 7.1 and documented in Deliverables D7.1 and D7.2, enabling the identification and quantification of improvements achieved through the adoption of the TaRDIS toolbox.
5. **Use Case Evaluation:** Each use case reports requirement fulfilment and KPI achievement using a standardised reporting structure, combining tabular summaries with concise explanatory narratives to ensure clarity and comparability across heterogeneous application domains.
6. **Cross-Use Case Synthesis:** Evaluation outcomes are synthesised across use cases to identify common findings, strengths, limitations, and lessons learned, while explicitly avoiding direct numerical aggregation of inherently heterogeneous KPIs.

This methodology ensures that the evaluation process is **rigorous, reproducible, and update-ready**, while accommodating the diversity of the TaRDIS use cases and preserving a clear path towards the final evaluation in Deliverable D7.5.

Funded by the European Union

# 2. Use Case Evaluation

This section presents the evaluation results for the four TaRDIS use cases—EDP, GMV, TID, and ACT—focusing on the fulfilment of their requirements and the performance of their Key Performance Indicators (KPIs). The structure of the evaluation follows the methodology outlined in Section 1.3 and applies a uniform reporting format to ensure comparability across heterogeneous application domains.
The assessment presented in this section builds directly upon earlier WP7 deliverables. Baseline behaviours and initial challenges identified in D7.1, together with the KPI definitions and measurement procedures established in D7.2, form the foundation for the performance analysis reported here. In addition, the requirements used for evaluation reference the definitions introduced in WP2 and subsequently refined through WP7 activities.
As part of the evaluation process, use case partners provide both quantitative and qualitative evidence demonstrating how each requirement was addressed and how each KPI was measured and achieved. Tools from the TaRDIS toolbox are referenced only when they contribute directly to observed performance improvements or when they enable specific measurement procedures; detailed descriptions of tool usage, integration choices, and technical configurations are reported in D7.4.
Each use case subsection includes:

- an assessment of requirement fulfilment,
- KPI performance results compared against the established baselines,
- explanations of measurement conditions and data collection procedures, and
- technical observations derived from the execution of the evaluation scenarios.

## 2.1 Multi-Level Grid Balancing Use Case (EDP)

EDP's use case objective is to demonstrate the benefits for the Energy sector include enhanced capabilities for establishing new energy communities, improved grid operations, and increased resilience due to the finer granularity in measurement and control. Another relevant outcome is the potential maximization of locally produced energy, which can have a positive environmental impact given its renewable nature and reduced distribution losses.

As it was stated in D7.1 and D7.2, the energy baseline for this use case is a centralized energy setup that requires a shift in how we manage and control it. We're moving away from the traditional centralized approach to a more distributed one. In this new setup, different energy sources and consumers collaborate to share energy, helping to maintain a stable grid. This mechanism of surplus energy from one source matching the energy needs of another, in order to ensure an efficient energy distribution system, is a problem similar to the computer science realm and to how swarm intelligence/collaborative intelligence works. Swarms of energy management devices can ensure grid stability, in an increasingly dynamic grid, where a more centralized system would have difficulties keeping up with the local changes.

The baseline has three different levels that represent different entities and different actions within this decentralised system. In Figure 1, these three levels are shown:

- Edge-Level: The interaction between the swarm of prosumers and community orchestrator that exist within an energy community.
- Fog-Level: This layer demonstrates the communication between community orchestrators from different communities and the exchange of energy and information.

Funded by the European Union

- Cloud-Level: the Distribution System Operator, which interacts with the community orchestrators using the communication network but continues to enable energy exchange to all consumers and producers at edge level.

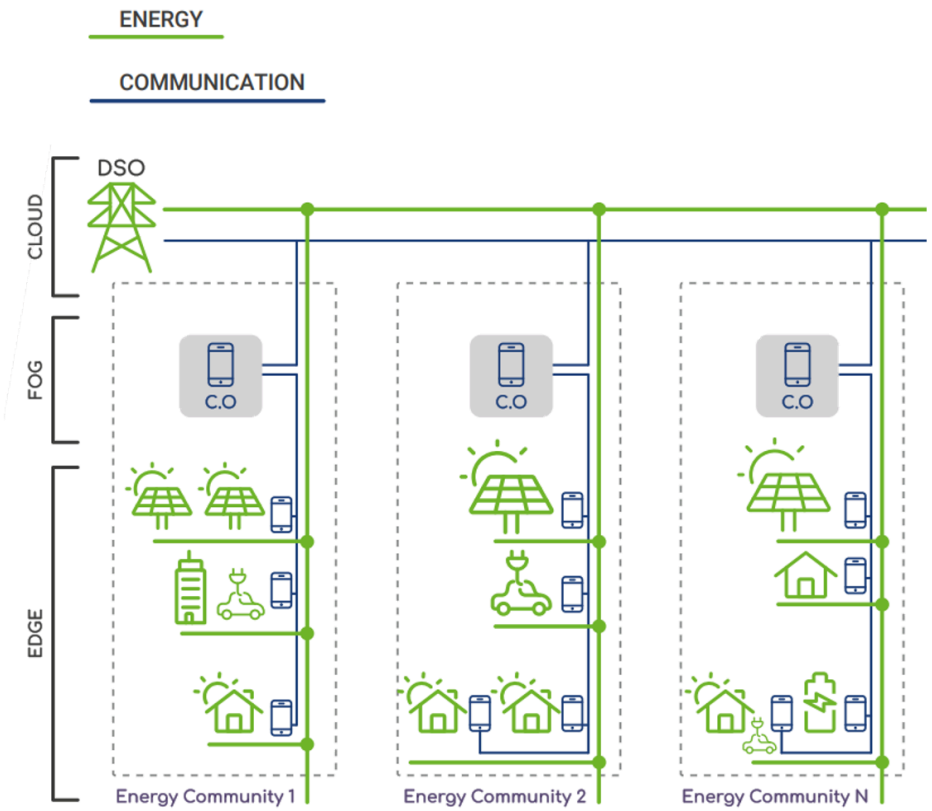


*Figure 1: Energy and communication layers at edge swarm, fog swarm and cloud*

The baseline implementation followed the second iteration of the concept illustrated in D7.1. This iteration adopts a distributed approach in contrast to the centralized one described earlier.

The software trials were conducted in a controlled laboratory environment, using a baseline setup illustrated in Figure 2. This configuration features three residential units, an electric vehicle (EV) charger, and a solar photovoltaic (PV) array. Specifically:

- **Houses H1 and H2** are equipped with standard electrical switchboards and represent real-world counterparts to the green units shown in the diagram.
- **C1** is a unidirectional residential wall box used for EV charging.
- **H3** is a dynamically controllable house connected to a rooftop solar PV array, which supplies real-time energy generation data and allows for dynamic adjustments.

Houses H1 and H2 are part of separate communities, each linked to the national grid. The setup also includes a grid emulator that interfaces with the TaRDIS toolbox via an API. A dedicated user interface enables the configuration of various grid conditions, creating an ideal environment for testing the described applications.

Funded by the European Union

This testbed is hosted at the **EDP Storage and Mobility Lab**, located within the **EDP LABELEC** facilities. It will serve as the platform for the testing application detailed in the following section.

Additional technical specifications include:

- A dedicated HP server with 10 cores, 32GB RAM, and 512GB of storage.
- Five Shelly smart meters for monitoring energy and power consumption.

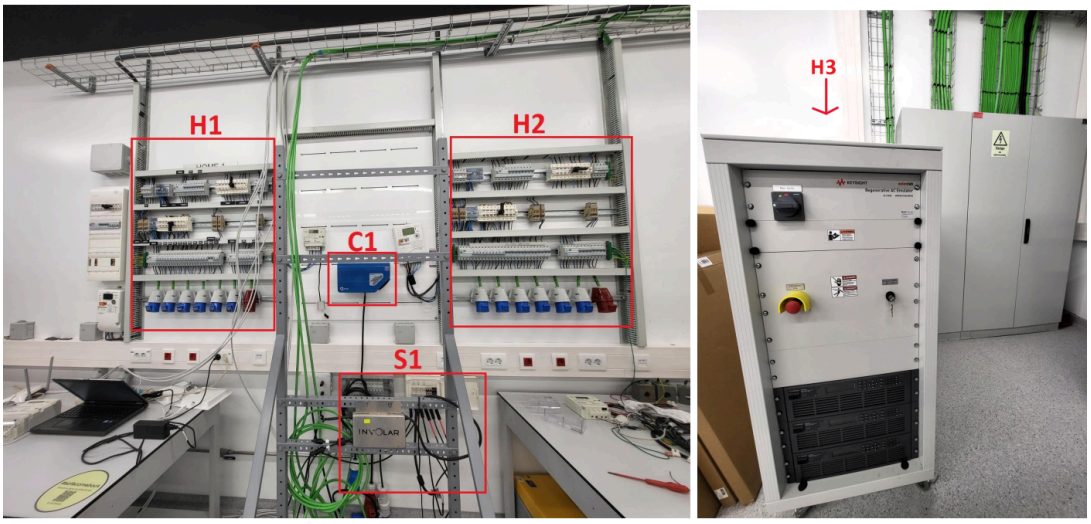


*Figure 2: EDP demonstrator setup*

The tests for the Energy case, described in the following subsections target the two working modes ex-ante or planning (T-EDP-SC01-1 to T-EDP-SC04-1) and runtime (the remaining tests).

These tests were described on the deliverable D7.2 and were used as evidence for the Use Case Requirements.

### 2.1.1 Requirements Evaluation

| Req. ID | Description | Type & Scenarios | Linked KPIs | Evidence | Fulfilment |
|---|---|---|---|---|---|
| **RF-EDP-01a** | Community acceptance and network registration | Must / 5+6 | K-B-01: programmer effort for overlay network<br>K-B-02: network bandwidth used<br>K-B-13: latency at interested peers<br>K-B-17: security verification effort<br>K-U-02 | *Test cases T-EDP-501 and T-EDP-601* | *Met* |
| **RF-EDP-01b** | Exchange agreement between prosumers | Must / 5+6 | K-B-01: programmer effort for overlay network<br>K-B-02: network bandwidth used<br>K-B-13: latency at interested peers<br>K-B-17: security verification effort<br>K-U-02 | *Test cases T-EDP-501 and T-EDP-601* | *Met* |
| **RF-EDP-02** | Community Orchestrator for Energy Communities | Must / 1 | K-B-01: programmer effort for overlay network<br>K-B-17: security verification effort | *Test case T-EDP-101* | *Met* |
| **RF-EDP-03** | Renewable Energy Optimization | Must / All | K-U-01 | *All Test Cases* | *Not Eval.* |

| | | | | | |
|---|---|---|---|---|---|
| **RNF-EDP -01** | Scalability | Should | K-B-11: scalability | *All Test Cases* | *Partial* |
| **RNF-EDP -02** | Security and Privacy | Should | K-B-17: security verification effort | *All Test Cases* | *Met* |

*Table 1: Requirements Performance for EDP Use Case*

The evaluation results reported in Table 1 are derived from the execution of the EDP test scenarios defined in D7.2 and summarised in Section 2.1. The fulfilment status of each requirement is determined based on the successful execution of the associated test cases and the measurement of the linked KPIs, as outlined below. Requirements marked as *Not Met* or *Partial* reflect the scope and configuration of the evaluated validation scenarios and do not imply deficiencies in the underlying TaRDIS toolbox components.

- **RF-EDP-01a and RF-EDP-01b (Community acceptance and exchange agreements)** are marked as **Met** based on the successful execution of test cases **T-EDP-501** and **T-EDP-601**, which demonstrate the establishment of energy communities, participant registration, and agreement-based energy exchange. The measured KPIs related to overlay effort, network bandwidth, latency, and security verification confirm that these operations can be performed effectively using the TaRDIS toolbox.
- **RF-EDP-02 (Community Orchestrator for Energy Communities)** is marked as **Met** based on test case **T-EDP-101**, which validates the correct operation of the community orchestrator and its role in coordinating energy flows and information exchange. The associated KPIs confirm that the orchestration functionality meets the defined functional requirements with acceptable implementation and verification effort.
- **RF-EDP-03 (Renewable Energy Optimisation)** is marked as **Not Evaluated**, as the optimisation of renewable energy usage, as quantified by **KPI K-U-01**, was not fully automated in the evaluated scenarios. While partial improvements were observed, the evaluated validation scenarios did not include closed-loop decision-layer functionality required to fully satisfy this requirement.
- **RNF-EDP-01 (Scalability)** is marked as **Partial**, as the evaluated scenarios demonstrate scalability within the tested laboratory setup and number of devices. However, large-scale scalability beyond the tested configuration has not yet been fully validated.
- **RNF-EDP-02 (Security and Privacy)** is marked as **Met**, as the relevant security-related KPIs and verification activities confirm that the evaluated scenarios satisfy the defined non-functional security requirements.

## 2.1.2 KPIs Evaluation

### 2.1.2.1 Use Case KPIs

| KPI ID | KPI Name | Baseline | Target | Achieved | Fulfilment |
|---|---|---|---|---|---|
| **K-U-01** | By using local renewable energy, less primary fossil energy from the grid will be required, thus reducing $CO_2$ emissions. | *Average consumption of a household assuming that all energy was supplied by a gas-fired power plant* | 50% reduction | 92-95% reduction | *Partial (scenario-based evaluation)* |
| **K-U-02** | Number of simulated citizens that take a more active role in the energy community and participate in Energy selling by using their own vehicles, with a target of 2 simulated citizens. | 2 simulated citizens | 2 simulated citizens | 8 simulated citizens | *Met* |

*Table 2: Use Case KPIs evaluation in EDP Use Case*

Table 2 summarises the evaluation of the EDP use case–specific KPIs, including the baseline, target values, measured outcomes, and fulfilment status. The fulfilment assessment

is derived from the execution of the test scenarios defined in D7.2 and the measurement methodologies described below.

**K-U-01: Reduction of $CO_2$ Emissions through Local Renewable Energy Usage**

**KPI description:** This KPI evaluates the reduction of $CO_2$ emissions achieved by increasing the use of locally produced renewable energy within the energy community, supported by the TaRDIS communication and coordination mechanisms. The target reduction is set to **50%** compared to a baseline scenario where all energy demand is supplied by fossil-fuel-based generation. Although quantitative reduction exceeds the target, full fulfilment requires closed-loop decision-layer optimisation, which is evaluated as a requirement rather than a KPI.

**Baseline:** The baseline corresponds to the average annual electricity consumption of a household in the EU, estimated at approximately **4000 kWh**, assuming full supply from a gas-fired power plant. With an emission factor of approximately **0.40 kg $CO_2$/kWh**, this results in around **1600 kg $CO_2$/year**.

**Measurement methodology:**

1. The baseline $CO_2$ emissions are calculated assuming total household energy consumption supplied by gas-fired generation.
2. In the energy community scenario, the amount of energy imported from the grid is measured and recorded by the Community Orchestrator.
3. The corresponding $CO_2$ emissions are estimated based on the residual grid-supplied energy.
4. The ratio between the community scenario emissions and the baseline emissions determines the achieved reduction.

**Observed results and interpretation:** In the evaluated scenarios, a significant share of household energy demand is supplied by locally generated solar energy. Operational emissions from solar generation are effectively zero, while lifecycle emissions are estimated in the range of **20–50 g $CO_2$/kWh**. For an annual consumption of 4000 kWh, this corresponds to approximately **120 kg $CO_2$/year**, representing a potential emission reduction of **92–95%** compared to the baseline.

**Fulfilment assessment:** Despite the strong indicative reduction observed, this KPI is marked as **Partial**. The reason is that the evaluated scenarios do not yet implement fully automated, closed-loop optimisation of renewable energy usage across all operational modes. The observed reductions are derived from controlled scenarios rather than continuous optimisation driven by the decision layer. Full fulfilment is not claimed in this deliverable, as the evaluated scenarios do not include closed-loop, decision-layer-driven optimisation of renewable energy usage.

**K-U-02: Active Participation of Citizens in Energy Exchange**

**KPI description:** This KPI measures the number of simulated citizens actively participating in energy exchange within the community by using their electric vehicles (EVs) as flexible energy assets. The target is defined as participation by at least **two simulated citizens**, across a minimum of **three scenarios**. The baseline and reduction are calculated on a per-household basis and are representative of the aggregate community-level impact when extrapolated across participating households

**Baseline and target:** The baseline and target both correspond to the involvement of **two simulated citizens**.

**Measurement methodology:** This KPI is evaluated through inspection of the executed test scenarios, identifying cases in which EVs are used to exchange energy within the community.

**Observed results:** Across the evaluated scenarios, **eight simulated citizens** actively participated in energy exchange using their EVs, exceeding both the baseline and target values.

Funded by the European Union

**Fulfilment assessment:** Based on the number of participating entities and the successful execution of multiple energy exchange scenarios, KPI **K-U-02 is marked as Met**.

### 2.1.2.2 Baseline KPIs

| KPI ID | KPI Name | Baseline | Achieved | Fulfilment | Notes |
|---|---|---|---|---|---|
| **K-B-01** | Programmer effort for overlay | None (no explicit quantitative pre-TaRDIS measurement) | Reduced implementation and testing effort | *Partial* | Based on qualitative developer feedback |
| **K-B-02** | Network bandwidth used | None | Acceptable communication overhead | *Partial* | Qualitative assessment |
| **K-B-06** | CPU Usage | None | Detailed Metrics<br>Training phase: 65-75%<br>Averaging phase: 30-40%<br>Idle phase: 10-15%<br>Average overall: 45% | *Met* | Values (and additional metrics) are obtained by the execution of **Fedra** in the forecasting scenario, as described in D5.3. |
| **K-B-07** | Training Latency | None | Local training: 60s<br>Weight aggregation: 15s<br>Network sync: 10s<br>Total round latency: 85s | *Met* | Latency metrics are obtained by the execution of **Fedra** in the forecasting energy scenario, as described in D5.3. |
| **K-B-08** | RAM Requirements | None | Model Memory: 5 MB<br>Training Data: 50 MB<br>Runtime Overhead: 100 MB<br>Total RAM per Node: 155 MB | *Met* | The RAM requirements have been measured in the **Fedra** FL framework, as described in D5.3. |
| **K-B-10** | Accuracy | None | Training Accuracy: 92%<br>Validation Accuracy: 89%<br>Test Accuracy: 87% | *Met* | The training, testing and validation accuracies have been measured using **Fedra** FL framework, as described in D5.3. |
| **K-B-11** | Scalability | None | Validated scalability up to 8 devices in local setup | *Partial* | - |
| **K-B-13** | Latency at interested peers | None | Event propagation within expected bounds | *Partial* | - |
| **K-B-17** | Security verification effort | None | Qualitative indication of reduced effort | *Partial* | Based on qualitative developer feedback |

*Table 3: Baseline KPIs evaluation measured in EDP Use Case*

The baseline KPIs reported in Table 3 assess the overhead, performance characteristics, and resource efficiency of the TaRDIS toolbox when applied to the EDP energy use case. These KPIs focus on foundational aspects such as development effort, communication overhead, scalability, latency, and machine learning performance, and are evaluated with respect to the baseline configurations defined in D7.1 and D7.2
For these baseline KPIs, the value “None” in the Baseline column indicates that no explicit quantitative pre-TaRDIS measurement is available; instead, the baseline corresponds to qualitative or architectural reference implementations, as described in D7.1 and D7.2.
Additionally, for baseline KPIs without an explicit quantitative pre-TaRDIS reference, evaluation is performed through qualitative comparison against ad-hoc or manual implementations, in accordance with the evaluation methodology described in Section 1.3. The fulfilment status of each baseline KPI is determined based on measured values obtained during the execution of the EDP test scenarios and on qualitative evidence collected through developer feedback.

Below follows a description of the KPI results grouped by methodology:

**Development and Communication Overhead**

• *K-B-01 (Programmer effort for overlay)* is marked as **Partial** (i.e., Partially Met). Qualitative feedback collected during the implementation of the EDP use case indicates reduced effort for implementing and testing overlay network functionality when using TaRDIS abstractions compared to ad-hoc implementations. However, structured questionnaire-based evidence is not available at the time of this validation, preventing a full quantitative assessment.
• *K-B-02 (Network bandwidth used)* is marked as **Partial**. The evaluation is based on observations of overlay formation traffic, maintenance communication, and user data exchange between smart homes and the energy orchestrator. While communication overhead remains within acceptable bounds for the evaluated scenarios, the absence of quantitative thresholds and large-scale measurements limits full characterisation

**Scalability and Latency**

• *K-B-11 (Scalability)* is marked as **Partial**, as the evaluated scenarios demonstrate correct operation and coordination within a bounded experimental setup involving up to eight devices running locally. Larger-scale scalability behaviour is not characterised within the scope of the evaluated scenarios.
• *K-B-13 (Latency at interested peers)* is marked as **Partial**, based on observed event propagation delays between participating entities. The observed latencies satisfy the expected responsiveness requirements for the evaluated swarm sizes and operational conditions.

**Security and Verification Effort**

• *K-B-17 (Security verification effort)* is marked as **Partial**. Qualitative indications from the implementation experience suggest reduced effort for reasoning about and validating security properties when using TaRDIS-supported verification mechanisms. However, structured questionnaire-based evidence is not available at the time of this validation.

**Federated Learning Performance Metrics**
The following KPIs relate to the performance and resource efficiency of federated learning components used within the EDP use case and were measured using the Fedra framework, as detailed in D5.3:

• *K-B-06 (CPU usage)* is marked as **Met**, based on measured utilisation across different execution phases (training, aggregation, idle), demonstrating that computational overhead remains within acceptable bounds.
• *K-B-07 (Training latency)* is marked as **Met**, with the total round latency (including local training, aggregation, and synchronisation) remaining compatible with the operational requirements of the energy forecasting scenarios.
• *K-B-08 (RAM requirements)* is marked as **Met**, as the measured per-node memory footprint fits within the capabilities of the evaluated edge devices.
• *K-B-10 (Accuracy)* is marked as **Met**, based on training, validation, and test accuracy values achieved during federated learning execution, demonstrating that decentralised training does not significantly degrade model performance.
Overall, the baseline KPI evaluation confirms that the TaRDIS toolbox can be applied to the EDP use case with acceptable development effort, communication overhead, latency, and resource consumption within the scope of the evaluated EDP scenarios. Where KPIs are marked as Partial, this reflects the bounded scope of the evaluated validation scenarios rather than functional limitations of the approach.

#### 2.1.2.3 Objective KPIs

This subsection reports the evaluation of the **project-level objective KPIs** that are exercised and measured within the scope of the EDP use case.

| KPI ID | KPI Name | Target | Achieved | Fulfilment | Notes |
|---|---|---|---|---|---|
| **K-O-1.1** | Expressivity of the language primitives covers the needs of use cases | >=80% | 100% | Met | All EDP application logic implemented using TaRDIS primitives |
| **K-O-1.2** | Event-driven model effectively captures swarms' complexity and scale | Qualitative | Applied across all components with no expressivity limitations identified | *Partial* | Assessment based on qualitative developer feedback; questionnaires pending |
| **K-O-1.3** | Decrease median development time by 25% | 25% | 25% faster | *Partial* | Based on qualitative developer estimates; consolidated quantitative measurement pending |
| **K-O-2.2** | Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis | >=70% | 70% | *Partial* | Verification evidence based on tool outputs; consolidated questionnaire confirmation pending |
| **K-O-2.3** | Formal verification of 80% of TaRDIS runtime protocols | 80% | 100% | Partial | Applies to protocols exercised in EDP; full protocol set evaluated at project level |
| **K-O-3.1** | Use TaRDIS ML to autonomously manage system operations (used by 50% of use cases) | 50% of the use cases | Not Evaluated | *Not Evaluated* | Decision-layer components not completely integrated in EDP at this phase |
| **K-O-3.2** | Improved edge orchestration (15% faster response time, 20% faster event processing throughput) | 15% faster response time, 20% faster event processing | Not Evaluated | *Not Evaluated* | ML-assisted orchestration not exercised in EDP scenarios yet |
| **K-O-3.3** | Reduced transmission overhead by 20% (wrt FedAvg) | At least 20% | 25% | *Met* | The transmission overhead reduction is measured using the Fedra framework |
| **K-O-3.4** | Model reduction/compression increased by 15% (compared to NNmodel coding with ISO/IEC 15938-17- NNR) | Increased by 15% | 19% | *Met* | The model compression was measured using the pruning and the knowledge distillation lightweight tools |
| **K-O-3.5** | Reduced model training time by 25% (compared to current KubeFlow training operator's implementation) | At least 25% | 29% | *Met* | The model training time is measured using the Fedra framework and compared to KubeFLow implementation |
| **K-O-4.1** | Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices) | At least 5000 devices | 4000 | *Partial* | Scalability evaluated using container-based emulation on a computational cluster |
| **K-O-4.2** | Distributed data storage service, supporting partial replication (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices). | At least 5000 devices | 4000 | *Partial* | Correctness and scalability validated using artificial workloads and large-scale emulation |
| **K-O-4.3** | Adapters for external tools and libraries used by industrial partners (50% of middleware systems) | 50% | 50% | *Met* | Adapters validated using artificial workloads representative of the EDP scenario |
| **K-O-5.1** | Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices) | 80% of devices | 100% | *Met* | 100% of devices used in EDP |
| **K-O-5.2** | Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages) | At least 50% of languages are supported | 100% | *Met* | 100% of languages used in EDP |
| **K-O-5.3** | TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, | 50% | 50% | *Met* | Integration demonstrated for the middleware systems |

Funded by the European Union

| | | | | | |
|---|---|---|---|---|---|
| | Actyx (50% of middleware/systems) | | | | required by the EDP use case |

*Table 4: Objective KPIs performance measured in EDP Use Case*

The selected KPIs reflect the technical focus of the energy community scenario, particularly with respect to federated learning efficiency, communication overhead, model optimisation, scalability, interoperability, and device support. Objective KPIs that are not directly exercised in the EDP scenario are addressed at the cross-use case aggregation level in Section 3.

**K-O-1.1: Expressivity of the language primitives covers the needs of use cases**
**Measurement methodology:** The code base of the EDP use case scenarios is inspected during test execution to assess the extent to which TaRDIS language primitives and toolbox components are utilised, and whether additional ad-hoc mechanisms are required.
**Observed result and interpretation:** All application logic required by the EDP use case is implemented using TaRDIS language primitives and tools. No functionality required by the use case relies on external or ad-hoc mechanisms outside the TaRDIS toolbox.
**Fulfilment assessment:** This KPI is marked as **Met**, as the expressivity of the TaRDIS language primitives fully covers the functional needs of the EDP use case.
**K-O-1.2: Event-driven model effectively captures swarms' complexity and scale**
**Measurement methodology:** The assessment is based on qualitative developer feedback regarding whether any parts of the EDP implementation could not be effectively expressed or managed using the TaRDIS event-driven model due to complexity or scale constraints.
**Observed result and interpretation:** Within the evaluated EDP scenarios, the event-driven model is applied consistently across all components, and no limitations related to system complexity or scale were identified.
**Fulfilment assessment:** This KPI is marked as **Partial**, as the evaluation relies on qualitative observations and developer feedback. Consolidated questionnaire-based evidence across use cases is not yet available at this stage and will be addressed in the final evaluation.
**K-O-1.3: Decrease median development time by 25%**
**Measurement methodology:** Developers contributing to the EDP use case provide estimates of their development effort for implementing swarm-based functionality with and without the use of TaRDIS tools. Median effort per implemented feature is compared within the same organisational context.
**Observed result and interpretation:** Developer feedback indicates an approximate 25% reduction in median development time when using TaRDIS abstractions compared to baseline development approaches.
**Fulfilment assessment:** This KPI is marked as **Partial**, as the observed reduction is based on qualitative estimates rather than consolidated quantitative measurements across use cases.
**K-O-2.2: Verification of at least 70% of communication, security, and data integrity properties**
**Measurement methodology:** Verification coverage is assessed by identifying the set of communication, security, and data integrity properties relevant to the EDP use case and examining which of these are verified using formal analysis techniques, including DCR graph evaluation and attack-state reachability analysis.
**Observed result and interpretation:** Within the scope of the EDP use case, formal verification activities confirm that approximately 70% of the identified properties are verified, meeting the project target at the use case level.
**Fulfilment assessment:** This KPI is marked as **Partial**, as the assessment is currently limited to the EDP scenario and has not yet been consolidated through cross-use-case questionnaire-based validation. Final confirmation at project level is deferred to the final evaluation phase.
**K-O-2.3: Formal verification of 80% of TaRDIS runtime protocols**

**Measurement methodology:** The TaRDIS runtime protocols exercised within the EDP use case are enumerated, and their verification status is assessed based on the availability of formal verification artefacts.
**Observed result and interpretation:** All runtime protocols exercised in the EDP use case are formally verified, exceeding the target threshold of 80% for the protocols in scope.
**Fulfilment assessment:** This KPI is marked as **Partial**, as the evaluation covers only the subset of runtime protocols exercised within the EDP use case. Consolidated assessment across all use cases and runtime components will be performed during the final evaluation.
**K-O-3.1: Use TaRDIS ML to autonomously manage system operations**
**Measurement methodology:** Use case scenarios are inspected to determine whether ML-assisted autonomous management and orchestration components are exercised, including interactions among decision-layer TaRDIS tools.
**Observed result and interpretation:** The EDP use case does not exercise autonomous ML-based system management during this validation phase, as the integration of decision-layer components (including FAuNO) was still in progress at the time of evaluation.
**Fulfilment assessment:** This KPI is marked as **Not Evaluated**, as the relevant functionality is not exercised within the evaluated EDP scenarios.
**K-O-3.2: Improved edge orchestration**
**Measurement methodology:** Response time and event processing throughput are measured in scenarios leveraging ML-assisted edge orchestration.
**Observed result and interpretation:** ML-assisted edge orchestration is not exercised in the evaluated EDP scenarios; therefore, response time and throughput improvements are not measured at this stage.
**Fulfilment assessment:** This KPI is marked as **Not Evaluated**, as the required orchestration mechanisms are not part of the evaluated configuration.
**K-O-3.3: Reduced transmission overhead by 20% (wrt FedAvg)**
**Measurement methodology:** Transmission overhead is measured by quantifying network traffic during model weight exchange and inference communication in the federated learning process, using the Fedra framework as described in D5.3.
**Observed result and interpretation:** The EDP use case achieves a 25% reduction in transmission overhead compared to the FedAvg baseline.
**Fulfilment assessment:** This KPI is marked as **Met**, exceeding the project target.
**K-O-3.4: Model reduction/compression increased by 15%**
**Measurement methodology:** Compression rates are measured using pruning and knowledge distillation techniques and compared against baseline neural network encoding using ISO/IEC 15938-17 Neural Network Representation (NNR).
**Observed result and interpretation:** A 19% increase in model compression is achieved while maintaining acceptable accuracy.
**Fulfilment assessment:** This KPI is marked as **Met**, exceeding the target threshold.
**K-O-3.5: Reduced model training time by 25%**
**Measurement methodology:** Model training time is measured during federated learning execution and compared against a baseline implementation using the KubeFlow training operator.
**Observed result and interpretation:** The EDP use case demonstrates a 29% reduction in training time.
**Fulfilment assessment:** This KPI is marked as Met, exceeding the project target.
**K-O-4.1: Decentralised membership service scalability**
**Measurement methodology:** Scalability is evaluated by emulating increasing numbers of heterogeneous devices using container-based deployment on a computational cluster.
**Observed result and interpretation:** Correct operation of the decentralised membership service is validated for up to 4000 emulated devices, which approaches but does not fully reach the target scale of 5000 devices.

Funded by the European Union

**Fulfilment assessment:** This KPI is marked as **Partial**, as large-scale operation close to the target is demonstrated, but full validation at the target scale is not achieved within the evaluated EDP scenarios.

**K-O-4.2: Distributed data storage service with partial replication**

**Measurement methodology:** Correctness and scalability are assessed using artificial workloads exercising partial and dynamic replication, combined with large-scale emulation on a computational cluster.

**Observed result and interpretation:** The distributed data storage service demonstrates correct operation with partial replication for up to 4000 emulated devices, remaining slightly below the target scale of 5000 devices.

**Fulfilment assessment:** This KPI is marked as **Partial**, as the evaluated configuration validates near-target scalability but does not fully meet the defined threshold.

**K-O-4.3: Adapters for external tools and libraries**

**Measurement methodology:** The number of adapters implemented for external tools and libraries relevant to the EDP scenario is quantified and validated using artificial workloads representative of the use case.

**Observed result and interpretation:** Adapters covering 50% of the required middleware systems are implemented and validated within the EDP use case.

**Fulfilment assessment:** This KPI is marked as **Met**, as the achieved coverage meets the project target.

**K-O-5.1: Support for industrial partners' devices**

**Measurement methodology:** The evaluation assesses whether device heterogeneity prevents the adoption of the TaRDIS toolbox in the EDP use case scenario.

**Observed result and interpretation:** All device types used in the EDP use case are supported by the TaRDIS toolbox, with no device-related constraints identified during implementation or execution.

**Fulfilment assessment:** This KPI is marked as **Met**, as device support in the EDP scenario exceeds the project target.

**K-O-5.2: Support for programming languages used by industrial partners**

**Measurement methodology:** Programming languages used in the EDP implementation are compared against the set of languages supported by the TaRDIS toolbox.

**Observed result and interpretation:** All programming languages required by the EDP use case are supported by TaRDIS, exceeding the minimum project requirement.

**Fulfilment assessment:** This KPI is marked as **Met**, as the achieved language support exceeds the defined target.

**K-O-5.3: Integration with external middleware and systems**

**Measurement methodology:** Integration requirements with external middleware are assessed based on the needs of the EDP scenario.

**Observed result and interpretation:** Support is demonstrated for 50% of the middleware systems required by the EDP use case, meeting the project target.

**Fulfilment assessment:** This KPI is marked as **Met**, as the achieved level of middleware integration satisfies the defined requirement.

### 2.1.3 Summary and Observed Limitations

In summary, the evaluation of the EDP use case demonstrates that a substantial portion of the defined KPIs is fulfilled within the scope of the validated scenarios. The TaRDIS toolbox effectively supports the implementation of energy community functionality, federated learning workflows, communication mechanisms, and interoperability requirements, with several KPIs exceeding their defined targets.

At the same time, the evaluation highlights specific limitations related to the scope of the validated configuration. In particular, decision-layer components enabling fully autonomous system management and orchestration are not exercised in the current evaluation, resulting in partial or non-fulfilment of the corresponding KPIs. These limitations are due to the

integration status of decision-layer tools at the time of evaluation rather than functional constraints of the underlying approaches.
Furthermore, the validation is conducted in a controlled laboratory environment, using emulated and simulated configurations without direct coupling to the physical electricity grid. While this setup is sufficient to validate the functional behaviour, scalability trends, and performance characteristics of the use case, it does not capture the full complexity of live grid operation.
Overall, the results confirm the suitability of the TaRDIS toolbox for the EDP energy community scenario, while clearly identifying the scope of the validated configuration and the aspects that remain outside the current evaluation.

## 2.2 Distributed Navigation Concepts for LEO Satellites Constellations Use Case (GMV)

GMV's use case focuses on next-generation satellite swarms in Low Earth Orbit (LEO), where increasing constellation size and system complexity pose significant challenges for orbit determination and time synchronization (ODTS). The use case targets real-time, distributed onboard processing that optimizes computational and communication resources while maintaining efficient inter-satellite links to maximize navigation performance and robustness.

The baseline configuration for this use case, defined in D7.1 and further refined in D7.2, consists of two reference approaches. The first is a fully centralized ODTS solution, where a single node (e.g., a master satellite or a ground station) assumes instantaneous access to state information from all satellites and distributes the computed navigation solution across the constellation. Although not realistic for large constellations due to physical communication delays and limited connectivity, this baseline provides a theoretical upper bound for performance comparison.

The second baseline implements a decentralised but sequential ODTS approach, in which each satellite performs its own navigation computation using peer information constrained by realistic inter-satellite link availability. While this approach improves realism compared to the centralized baseline, its sequential execution does not fully capture the parallel and distributed behaviour expected in an operational satellite swarm.

Beyond ODTS computation, the use case also addresses the need for robust inter-satellite link orchestration. In particular, failures affecting individual satellites can disrupt scheduled measurements and degrade navigation performance. To address this, a distributed re-scheduling mechanism is developed to recompute inter-satellite link schedules upon fault detection, allowing the constellation to converge to a consistent configuration in a synchronized manner.

The application of the TaRDIS toolbox enables the implementation and evaluation of the ODTS solution in a fully distributed swarm environment. Each satellite is represented by a parallel execution instance, allowing synchronized communication, measurement exchange, and fault propagation. This setup also supports the validation of the re-scheduling mechanism and its triggering logic, providing insight into robustness and convergence properties under realistic operational constraints.

***Validation Scope & Approach***
Validation of the GMV use case is performed through a combination of formal test cases and scenario-based evaluation. Due to the heterogeneous maturity of the different functional components, not all test cases defined in D7.2 are fully exercised at this stage. Where

Funded by the European Union

applicable, partial execution, qualitative evidence, or proxy metrics are reported. Test cases that could not yet be fully executed are explicitly identified and deferred to the final evaluation in D7.5.

***Test Cases***

***Test case* T-GMV-001 – Static verification**

The distributed ODTS algorithm designed, analyzed and tested in a swarm environment using the TaRDIS tools was subsequently translated into C to verify the feasibility of its implementation in a realistic satellite system. As part of this process, static verification techniques were applied using CppCheck and SonarQube.
The results of this verification were successful, with the code being correctly validated, which enabled its implementation on the satellite board. Selected metrics analysed during this process are presented as results of test T-GMV-005 in Figure 8.

***Test case* T-GMV-002 – Dynamic verification**

As part of the dynamic verification process of the C code, a set of unit tests was carried out to ensure the correct functionality of the different modules. As detailed in Figure 6 as part of test T-GMV-004, a 100% score was achieved in the correctness metric, since all executed tests passed successfully, thereby ensuring the functionality and safety of the code.

***Test case* T-GMV-003 - Scalability**

Several simulations have been carried out using the application resulting from the use case development with different types of constellations, mainly varying the number of satellites. Figure 3 shows, for instance, a snapshot of the connection scheme and constellation structure for the constellation specifically developed for the validation of the use case (the so-called TaRDIS constellation, in the left, with 170 satellites), as well as for a OneWeb-like constellation in terms of orbital planes, with 96 satellites (in the right).

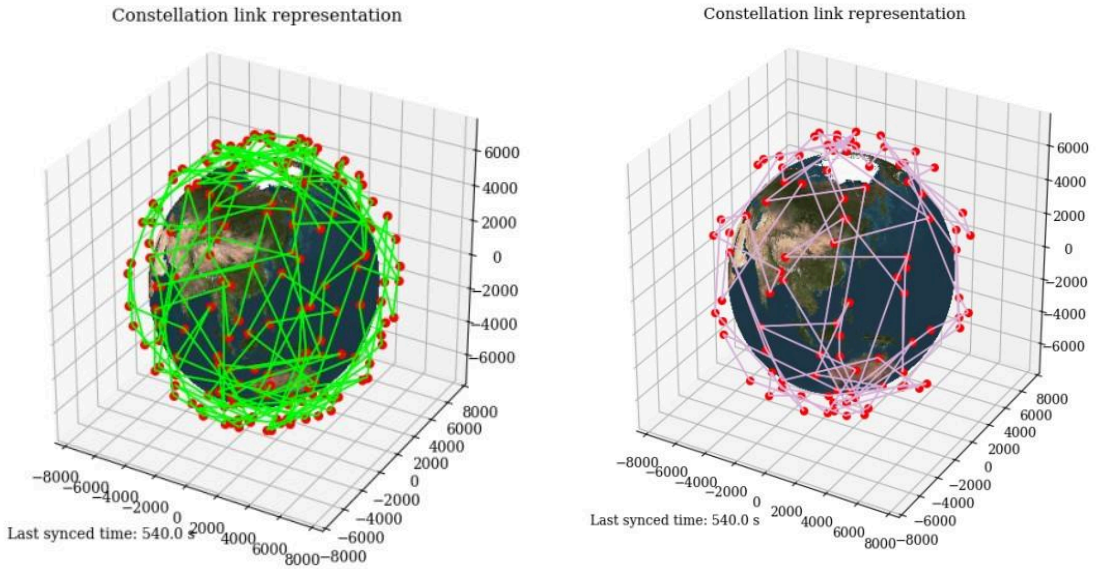


*Figure 3: Snapshot of TaRDIS constellation (left) and OneWeb-like constellation (right)*

It is evident that different orbital configurations lead to different results (a higher number of satellites can be associated with a larger variety of peers and, therefore, better measurements geometry, which explains the improved performance). Nevertheless, the

results clearly show that the application and the designed distributed algorithm are capable of solving the navigation problem and achieving convergence for different types of constellations, as illustrated in Figure 4 and Figure 5.

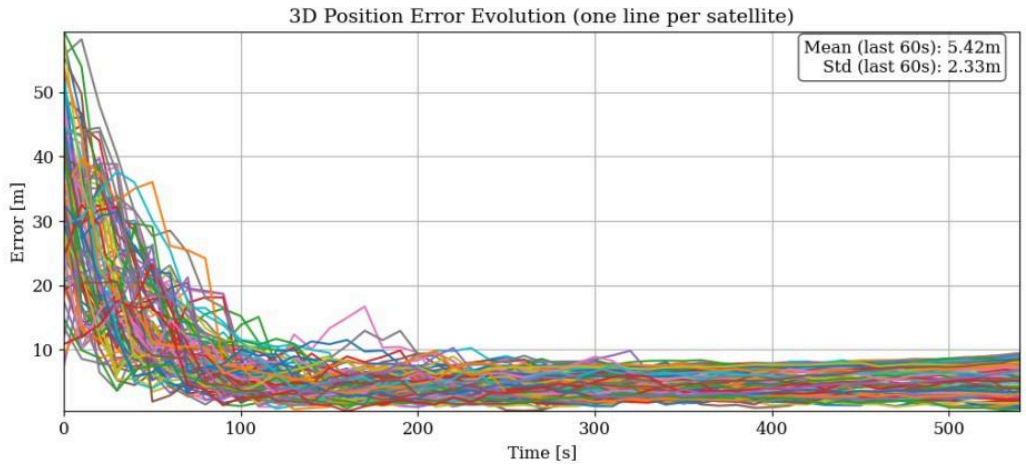


*Figure 4: Example of evolution of the 3D error in position throughout the constellation for the OneWeb-like constellation*

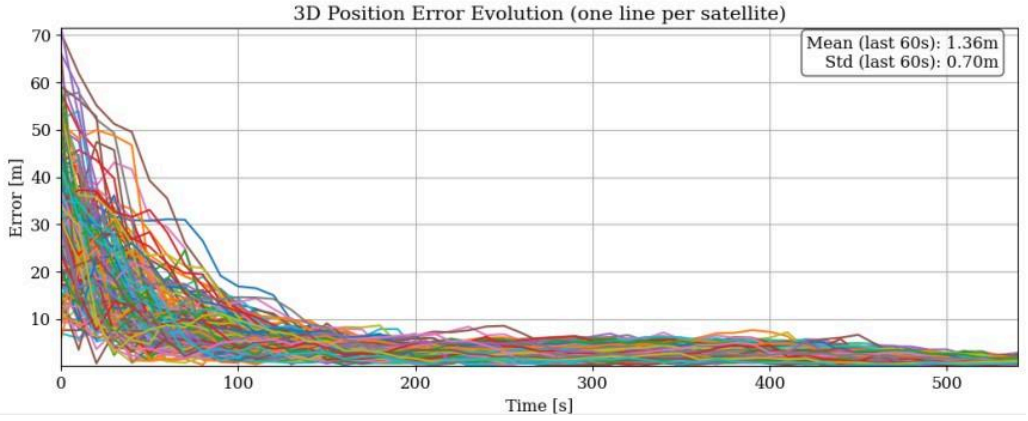


*Figure 5: Example of evolution of the 3D error in position throughout the constellation for the TaRDIS constellation*

### *Test case* T-GMV-004 – ECSS compliance

During the development process of the onboard software intended to be implemented on the satellite board, a methodology compliant with ECSS standards was followed. The objective was to reach a 100% value for the metrics presented in Figure 6.

| Goal properties | Associated Metrics | Target value |
|---|---|---|
| Completeness | Requirements Allocation | 100% |
| | Tests and Validation Coverage Completeness | 100% |
| Correctness | Testing/Validation Progress | 100% |

*Figure 6: GMV code quality metrics according to ECSS*

As already mentioned for previous tests, a validation campaign based on unit testing was carried out for all functions, achieving a requirements allocation of 100% and a progress of

Funded by the European Union

100% in the testing, thus fulfilling both metrics through the successful execution and passing of all tests.

Regarding the coverage metric, a value of 92% has currently been achieved. This is due to the high complexity and modularity of the code, which makes it challenging to reach a full 100% coverage. Nevertheless, these efforts are still ongoing, and the metric value is expected to increase further, approaching full coverage.

### *Test case* T-GMV-005 - ECSS compliance

The evaluation of this test can be considered consistent with that of T-GMV-001. As a result of the static verification process, compliance with ECSS metric objectives established for the onboard software (see Figure 7) has been assessed.

| Goal properties | Associated metrics | Target value |
|---|---|---|
| Analyzability | Code Comment Frequency | >10% |
| Modularity | Nesting Levels | <= 8 |
| | Lines of Code (LOC) per function | <= 250 |
| | LOC per file | <= 1500 |

*Figure 7: GMV compliance metrics according to ECSS*

Figure 8 shows that all metrics are fulfilled at 100%, with the exception of one: the comment frequency. While the more technical and application-related functions present a sufficient level of commenting to ensure that the code remains readable and understandable, only two files do not meet this target. These correspond to the ODTSmanager module files, which are responsible for orchestrating the data exchange between the satellite board and the simulated proxy node within the Python-based swarm (see D7.4 for additional context on the final demo setup). This code is relatively straightforward and does not require extensive commenting, which explains the partial compliance with this metric; however, it will be updated to achieve full compliance.

| Metrics | Comment frequency | Nesting level | Function code lines | File code lines |
|---|---|---|---|---|
| **Scope** | **File** | **Function** | **Function** | **File** |
| **Target** | **>= 10%** | **<= 8** | **<= 250** | **<= 1500** |
| **Deviate** | **2** | **0** | **0** | **0** |
| **Deviated with justification** | 0 | 0 | 0 | 0 |
| **Do not deviate** | 6 | 58 | 58 | 8 |
| **Total** | 8 | 58 | 58 | 8 |

*Figure 8: Extract of the Code Status Report performed as part of the static verification*

### *Test case* T-GMV-101 – Nominal Execution

This test corresponds to the correct nominal execution of the application developed for GMV's use case, without introducing any type of failures. It should be noted that, as explained in sections "*Use of Machine Learning to replace/aid the ODTS update phase / EKF module*" and "*Use of Machine Learning to replace the orbit propagation module*", this test was carried out including the complete distribution and decentralization of the ODTS process among the nodes, thus validating the distributed algorithm developed and demonstrating how the TaRDIS tools employed enabled and facilitated this task. However, it does not include machine learning models (either because their implementation was discarded or because their integration was not completed). Therefore, some of the requirements intended to be

evaluated through this test could not be assessed, although it was possible to do so for the vast majority.

In test T-GMV-003, results obtained by the application were already presented, thus demonstrating its correct operation. In the simulations, each node has three antennas/information exchange channels: two with satellites for ISL, and one pointing toward Earth for communications with the Ground Station. Only a few satellites in the entire constellation connect to ground stations in each timeslot, and they do so only to perform measurements against a highly self-aware reference to improve the accuracy obtained. However, the navigation solution is computed autonomously, onboard, and in a distributed manner. This implies a loss of precision compared to the centralized solution considered as the baseline, since in that case a single central node (usually on Earth) resolves the navigation for the entire constellation using information from all its nodes, thus having full system observability. In the distributed system, for navigation resolution, each satellite only uses the information (state vector and covariance matrix) of itself and its direct peers, therefore having more limited observability.

Even so, the level of accuracy obtained does not differ significantly from that achieved with the centralized baseline, as shown in Figure 9 and Figure 10, obtained from two simulations with the same filter tuning, same initial uncertainties, and same scheduling applied, using the centralized baseline algorithm and the distributed algorithm developed.

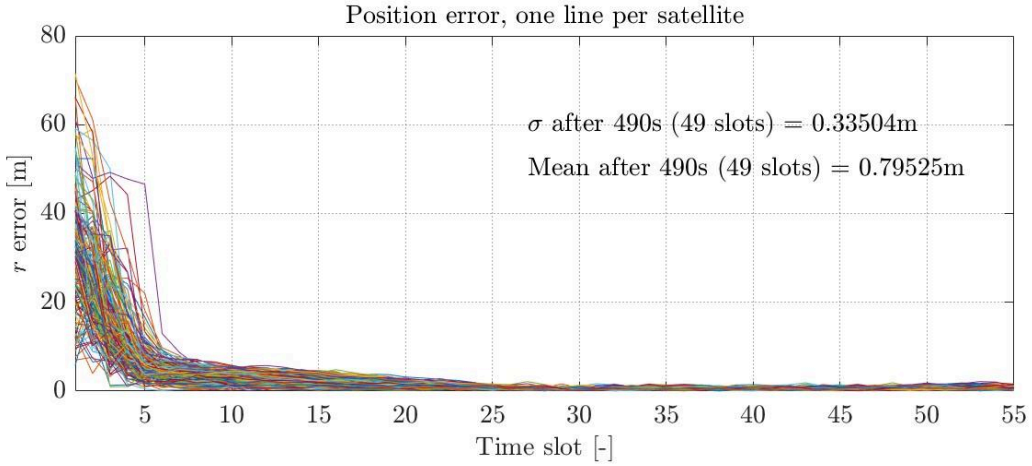


*Figure 9: Example of evolution of the 3D error in position throughout the TaRDIS constellation with the centralized baseline approach*

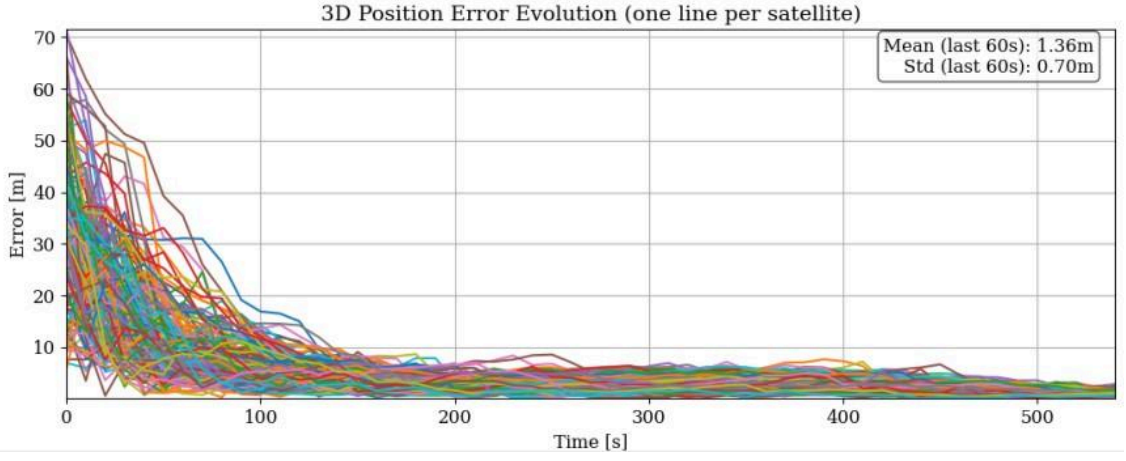

*Figure 10: Example of evolution of the 3D error in position throughout the TaRDIS constellation with the developed distributed approach*

The accuracy achieved with the centralized approach is clearly better, yielding a mean error of 0.8 meters during the last minute of the simulation, with a significantly reduced standard deviation. In contrast, with the distributed approach, both the error and the standard deviation increase, as expected for the reasons discussed above, but not excessively and always within acceptable limits.

***Test case* T-GMV-102 – Satellite communication replicability**

The communications between the representative board and the proxy node included in the computational swarm simulation follow the Packet Utilization Standard (PUS) protocol, widely used in satellite and space applications. It has been verified that the message packaged for transmission to the board, once unpacked, is exactly identical to the original one, thus ensuring correctness of the process. In addition, it has been ensured that, in the event of receiving a message with a faulty CRC sequence, the algorithm does not use the received information and discards it. Therefore, the communications that would occur in a real operational system are successfully replicated.

***Test case* T-GMV-103 - Communication failure**

For this test, the nominal TaRDIS constellation scenario was executed (with the same configuration that produces *Figure 5* in test T-GMV-003), but two scheduled total failures were introduced: one satellite "loses" its antennas at second 40, and another one at second 100.

The objective of the test is to verify that when some satellites become unresponsive, the remaining satellites are able to continue with their connection schedule and do not remain waiting for a connection that will never occur (deadlock freedom). In addition, it is verified that the failed satellites, although isolated, are still able to estimate their navigation solution (although with increased error levels due to the absence of measurements providing corrections to the Kalman Filter of the algorithm).

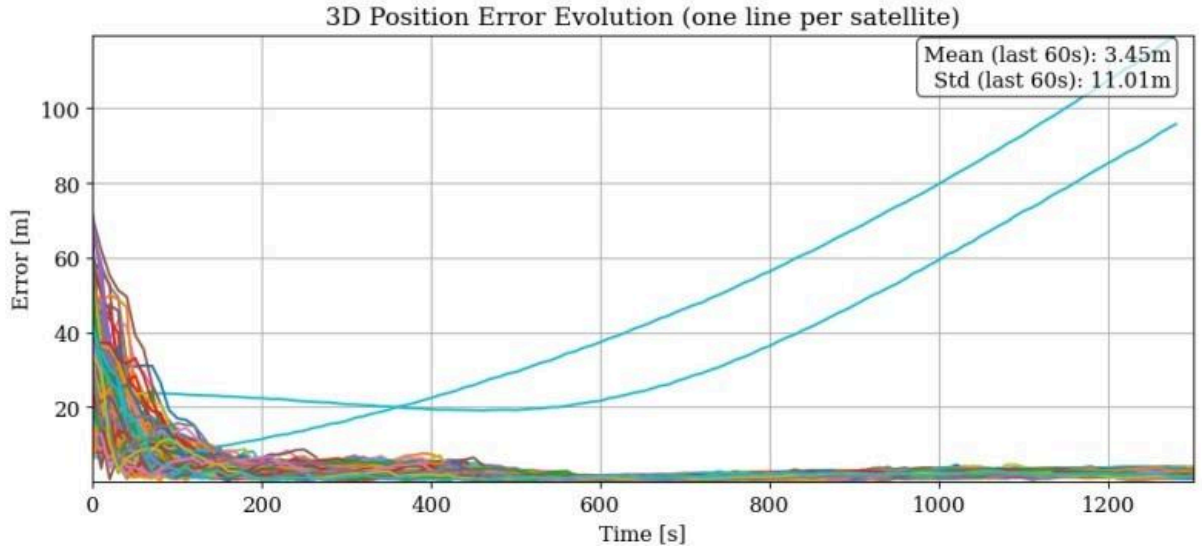

*Figure 11:* Evolution of the 3D error in position throughout the TaRDIS constellation introducing two total failures in two satellites in seconds 40 and 100 of the simulation.

### *Test case* T-GMV-104 - Computational resources

Test case T-GMV-104 aims to compare the computational resources usage of the baseline solution with that achieved using the TaRDIS toolbox. Although in D7.2 this comparison was initially envisaged in terms of execution time, a further assessment of the evaluation methodology led to discarding this approach for the following reasons:

- The baseline solution follows a centralized approach and has been developed in MATLAB.
- The final solution achieved with TaRDIS is fully distributed and has been implemented in Python. In addition, the TaRDIS-based application simulates the complete swarm, including synchronized peer-to-peer information exchanges.

The main advantage of the distributed approach compared to the centralized one lies in the fact that, instead of a single central node (either a Ground Station or a satellite acting as a master) performing the full ODTS computatoin for the entire constellation, which is computationally expensive and requires processing a large amount of data simultaneously, each node computes its own navigation solution using only its local information and the data exchanged with its direct peers. This appraoch drastically reduces the execution time and computational resource usage at the level of an individual node when compared to the centralized approach. However, this benefit is not directly observable when comparing the execution of both approaches as implemented, since the distributed solution includes the simulation of the full constellation together with synchornized inter-node communications, resulting in a total execution time of the same order of magnitude of the centralized approach.

Furthermore, in the final application, as it will be disclosed later in this section, the integration of the Machine Learning components has not been completed at the time of writing this deliverable. As a consequence, it is not yet possible to quantify the expected reduction in computational resource usage. For this reason, the Machine Learning efficiency requirements and the related KPI K-U-06, associated with computational resource usage, are reported as Not Evaluated, rather than Not Met, and their evaluation is deferred to D7.5.

### *Test case* T-GMV-105 - Onboard implementation feasibility

This test has been performed partially. On the one hand, the distributed ODTS algorithm has been successfully implemented on a representative satellite board, thereby confirming the feasibility of its implementation in a realistic onboard system. However, it should be clarified that, at the time of writing this deliverable, this implementation does not include any Machine Learning component.

The available machine learning model (which replaces the orbital propagation module) is currently under integration, and the feasibility assessment for its implementation on the satellite board is still pending. For this reason, the associated requirements have been marked as "*Not Met*", while remaining pending evaluation.

### *Advanced Orchestration and Robustness Tests*

The following test cases evaluate advanced orchestration, scheduling optimisation, and information propagation properties of the GMV use case. These tests complement the

nominal execution scenarios by assessing performance and robustness under optimized and reconfigured inter-satellite link schedules. They are particularly relevant for large-scale constellation operation and fault-tolerant behaviour.

***Test case* T-GMV-201 – Optimised Link Scheduling Impact**

For this test, nominal simulations (similar to those in test T-GMV-100) were carried out with exactly the same configuration (epoch, constellation, filter tuning and applied initial uncertainties), except for the sequence of connections between satellites in the different timeslots. On one hand, this simulation was performed using an ISL scheduling randomly generated (i.e., considering only the visibilities at each instant between satellites, without applying any optimization criteria). On the other hand, a schedule generated with the preprocessing scheduling tool was applied, which aims to optimize the orchestration of satellite connections under different criteria (reducing dilution of precision, increasing the number of distinct peers each satellite links to or avoiding to successive connections with the same peer), ultimately creating a better geometric measurement environment for the filter. As shown in Figure 12 and Figure 13, this approach achieves a reduction in the average 3D position error of around 25% for this scenario (4.08m to 2.97m).

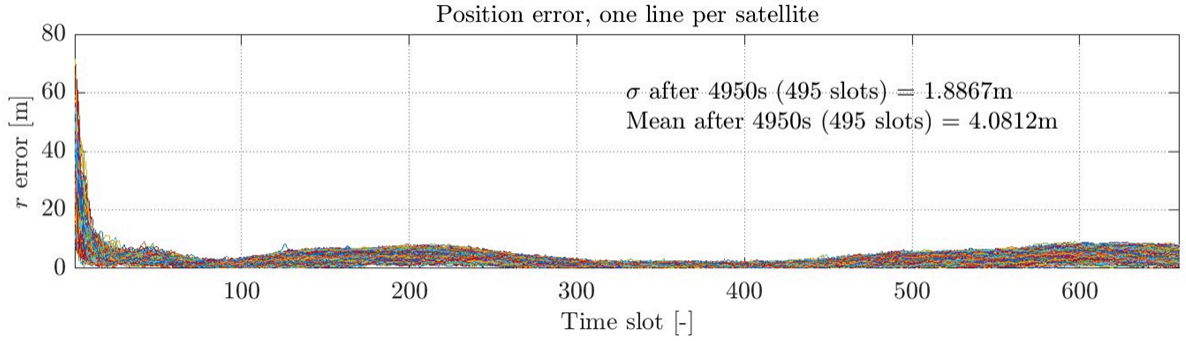


*Figure 12: Evolution of the 3D error in position throughout the constellation using a randomly generated scheduling*

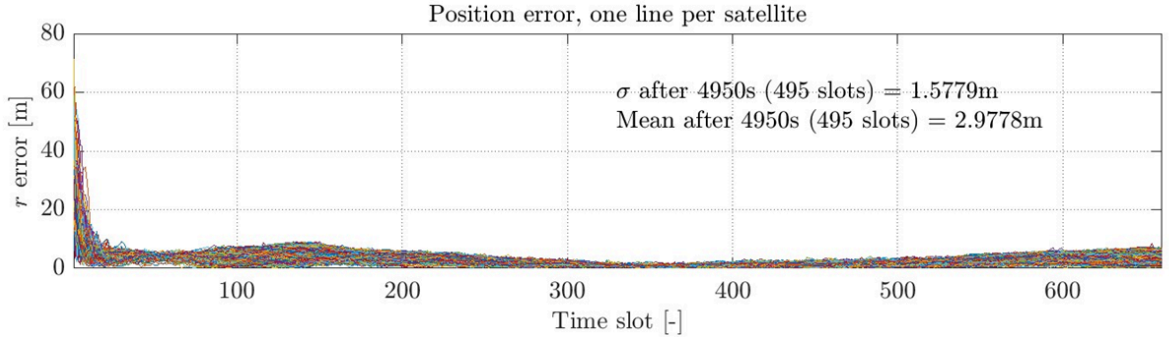


*Figure 13: Evolution of the 3D error in position throughout the constellation using the optimised scheduling*

These results demonstrate that optimised orchestration of inter-satellite links significantly improves navigation accuracy and validates the effectiveness of the scheduling tools applied within the TaRDIS framework.

***Test case T-GMV-202 - Visibility-Correct Scheduling***

This test verifies that the optimised scheduling algorithm preserves physical visibility constraints by construction, as it incorporates the baseline visibility calculation mechanism. All generated links were validated to be feasible for both endpoints, ensuring correctness of the orchestration process.

***Test case T-GMV-203 - Information Propagation and Synchronisation***
The link schedules generated with the optimized link orchestration algorithm were tested, verifying that by applying the established optimization criteria, the constellation was able to distribute any piece of information generated at one node to all other nodes in less than, for example, 2 timeslots. This is key for rescheduling capability, since all nodes must synchronously trigger the process and reach a consistent solution among them. Therefore, it is necessary that the fault dictionary (a compilation of failed connection reports) and the ephemerides of each node (to enable onboard visibility analysis) are distributed throughout the constellation via piggybacking during information exchanges between satellites.

In the developed application, a node can distribute its ephemerides and fault dictionary to all satellites that are part of the connection chain it belongs to. If the entire constellation forms a single chain, all satellites will be synchronized (having the most up-to-date information from all their peers). However, due to the optimization criteria used to maximize navigation performance, not all satellites are always in a single chain. In that case, a satellite can only distribute its information to the members of the sub-chain it is in, and it will have to wait until the next timeslot for this satellite (or any peer that already has its information) to transfer it to satellites that were “isolated” in the previous timeslot.

The result of a schedule generated for one week for the TaRDIS constellation has been verified, and all connections have been scanned to count situations where a satellite’s information took more than two timeslots to reach another satellite. The result obtained is that this condition was met in 99.99898% of the cases (approximately 17600 pairs of satellites did not manage to exchange information within a timespan of two timeslots, compared to more than 1.7 billion pairs to be evaluated over the course of a week). In addition, it should be noted that, due to the network topology itself, the vast majority of these pairs that do not achieve information transfer within two timeslots would do so within three timeslots, which is also considered acceptable given that each timeslot has a duration of 10 seconds. Therefore, the achieved level of reliability in terms of data distribution through the swarm by means of peer-to-peer piggybacking-based communication is considered outstanding.

***Test case* T-GMV-204**
The rescheduling module has been developed based on the optimized scheduling algorithm that was also designed within the scope of the project, adapting it for distributed onboard operation. Its development is considered successful, as it has been verified that its design allows all nodes in the swarm to eventually reach consensus on the scheduling solution to be applied after failures of their peers. However, its real-time operation has not been tested within a full-constellation simulation, since this would be computationally very expensive: running 170 rescheduling processes in parallel with each other, and in parallel with the ODTS distributed resolution, becomes unfeasible in terms of simulation resources. Nevertheless, through the thorough testing campaign that has been carried out, its viability for final operational application has been assessed.

To ensure eventual consensus of the rescheduling resolution, two equally important aspects have been verified:

- Upon the failure of a node, the rest of the constellation is able to propagate the failure notification (under robustness layers, i.e., the failure must be repeated over several consecutive timeslots to avoid triggering rescheduling in response to spurious events), so that the entire constellation triggers the rescheduling computation simultaneously.
  It is critical that all satellites decide to apply rescheduling, and do so in a synchronized manner, even though, due to the constellation topology, not all satellites

are always interconnected in a single chain and therefore the information available to them at a given instant may be asymmetric.
- The rescheduling algorithm generates exactly the same solution for all nodes, so that when rescheduling is applied, all of them know which peers to connect to and no inconsistencies arise (e.g., a node attempting to establish an ISL with another node that does not reciprocate).

Although the rescheduling process has not been executed within a full simulation, the triggering mechanism has been incorporated. In this way, it has been verified that in the presence of a failure (as in the case of test T-GMV-103), the failure notification is propagated through peer-to-peer connections between satellites by leveraging piggybacking, using a shared failure dictionary. By applying robustness mechanisms and safety margins, it has been confirmed that for plausible and expected events (i.e., one or a few simultaneous satellite failures), the swarm is able to trigger the initialization of the rescheduling process in a synchronized manner.

Additionally, offline tests outside the simulation environment have been conducted by feeding the rescheduling algorithm with asymmetric inputs in terms of failure dictionaries and ephemerides (since not all nodes always have the most up-to-date information about all peers). In all cases, the resulting link re-orchestration solution has been shown to be consistent. This is largely due to the fact that information asymmetry is constrained (i.e., the information available to different nodes may differ by only a few timeslots, not by a large amount), a constraint that is consistent with the simulation assumptions: an initially well optimized scheduling that maximizes information propagation efficiency, and a realistic failure model (i.e., not half of the constellation failing simultaneously, but only a few satellites, which is coherent with the expected reliability of spaceborne equipment).

***Test case* T-GMV-205**

The execution of this test differs from what was originally defined in deliverable D7.2, where the validation of a decentralised storage solution was foreseen. However, over the course of the project it was concluded that it was not realistic to address this type of solution in the context of satellite constellations. Instead, and taking advantage of the fact that scheduling is designed to maximize connection diversity and to avoid the creation of isolated sub-chains of peers in each timeslot, the goal has been to ensure that, through the scheduling generated by the developed scheduling tool, any piece of information (e.g., ephemerides) from a given node can reach the rest of the swarm in two timeslots. As discussed with test T-GMV-203, this objective is achieved in almost all cases, and in those very few cases where it is not, the information is almost always propagated within three timeslots. In this way, it is ensured that all nodes maintain a constellation ephemeris database that is continuously updated through peer-to-peer information exchanges leveraging piggybacking.

### 2.2.1 Requirements Evaluation

This section reports the results of the validation activities carried out to assess compliance with the requirements defined for the GMV use case. These requirements were initially introduced in D2.2 and subsequently refined in D2.3. In total, the GMV use case defines 25 requirements, comprising 23 functional and 2 non-functional requirements.

During the course of the project, the GMV use case design evolved as a result of technical feasibility assessments and integration constraints. As a consequence, certain initially defined requirements were either revised, partially addressed, or deemed not applicable within the scope of the current validation phase. In such cases, this section explicitly

documents the rationale for non-fulfilment or partial fulfilment, in line with the iterative validation approach described in the Description of Action.

**Use of Reinforcement Learning for a ISL re-scheduling module**

When rescheduling was proposed as a possible measure to increase system robustness against partial or total communication failures between satellites, Reinforcement Learning was considered as a tentative solution worth exploring. This approach could enable, in a lightweight manner, all nodes in the constellation to include agents capable of reaching a synchronised and consistent solution. This idea led to the definition of a set of requirements intended to assess the model's capabilities, covering both performance aspects and computational characteristics (RF-GMV-15 in a direct way, RF-GMV-19 in an indirect way).
However, upon further advance in the project, we reconsidered this approach. Reinforcement learning needs a huge quantity of data to achieve a great performance, which poses a limitation for its use on such restrictive and resource-limited hardware as those employed in space systems. Furthermore, the proposed approach relied on achieving results within a relatively short time frame, which was also considered impractical.
As a solution, GMV has opted to implement a rescheduling algorithm derived from the optimized scheduling algorithm developed at the beginning of the project for offline orchestration. This is a lighter and faster version that performs a decision-making/optimization process but does not involve any type of machine learning.
As a result, requirements explicitly linked to the use of reinforcement learning for ISL re-scheduling are marked as *Not Met* in this validation phase, while the underlying functional objectives are addressed through alternative algorithmic solutions.

**Use of Machine Learning to replace/aid the ODTS update phase / EKF module**
The use of machine learning techniques to replace or assist the ODTS update phase (EKF module) was initially foreseen as part of the GMV use case. However, due to integration priorities and the need to stabilise the distributed ODTS implementation, this functionality was not fully integrated and exercised within the current validation phase.
Consequently, requirements directly related to this ML component are marked as Not Met at this stage and are planned to be revisited in the final evaluation phase.

**Use of Machine Learning to replace the orbit propagation module**
A machine learning–based replacement of the orbit propagation module is currently under integration. While preliminary development activities have been carried out, full validation on representative onboard hardware has not yet been completed at the time of writing this deliverable.
For this reason, the associated requirements are marked as Not Met and will be assessed during the final evaluation reported in D7.5.

| Req. ID | Description | Type & Scenarios | Linked KPIs | Result | Evidence | Fulfilment |
|---|---|---|---|---|---|---|
| **RF-GMV-01** | Decentralised ODTS framework | Must / 1 | K-U-05: Achievable distributed on-board ODTS performances versus the classical centralized on-ground ODTS. | Distributed ODTS successfully achieved | See T-GMV-101 | Met |
| **RF-GMV-02** | Interaction between satellites in the distributed model | Must / 1 | K-B-06: FL CPU usage for training<br>K-B-07: FL training latency<br>K-B-08: FL storage/RAM requirements per node | Distributed approach successfully implemented, and it enables FL techniques by integrating PTB-FLA (although not applied in the UC) | See T-GMV-101 | Met |

| | | | | | | |
|---|---|---|---|---|---|---|
| **RF-GM V-03** | Deadlock freedom | Must / 1 | N/A (Validated in WP7 demonstrations) | Deadlock-free execution observed during distributed ODTS simulations | See T-GMV-103 | Met |
| **RF-GM V-04** | Capability of simulating satellite communications | Should / 1 | N/A (Validated in WP7 demonstrations) | Simulation of inter-satellite communications successfully demonstrated | See T-GMV-102 | Met |
| **RF-GM V-05** | Capability of simulating satellite communication failures | Must / 1 | N/A (Validated in WP7 demonstrations) | Communication failure scenarios successfully simulated and handled | See T-GMV-103 | Met |
| **RF-GM V-06** | Capability to perform multiple simulations in parallel | Could / 3 | N/A (Validated in WP7 demonstrations) | This capability was not exercised within the scope of the GMV validation activities | N/A | Not Met |
| **RF-GM V-07** | Orbit propagation algorithm efficiency | Should / 1 | K-U-06: Reduction of the use of computational resources. | Efficiency evaluation of the orbit propagation algorithm was not completed within the current validation phase | See T-GMV-104 | Not Met |
| **RF-GM V-08** | Optimization of the Inter-satellite Link connectivity scheme | Must / 2 | N/A | ~25% improvement in the positioning error when using the optimized scheduling | See T-GMV-201 | Met |
| **RF-GM V-09** | ODTS algorithm / ML model efficiency | Must | K-U-06: Reduction of the use of computational resources. | ML-based ODTS efficiency was not evaluated, as the ML model was not integrated in the evaluated configuration | See “Use of Machine Learning to replace/aid the ODTS update phase / EKF module” | Not Met |
| **RF-GM V-10** | Robustness of the ODTS solution | Must | K-U-05: Achievable distributed on-board ODTS performances versus the classical centralized on-ground ODTS. | Distributed ODTS demonstrates robust convergence under nominal operation and communication failure scenarios | See T-GMV-103 | Met |
| **RF-GM V-11** | Implementation of an FL model for distributed ODTS | Must / 1 | K-U-05: Achievable distributed on-board ODTS performances versus the classical centralised on-ground ODTS. K-U-06: Reduction of the use of computational resources. | FL-based ODTS model was not implemented or exercised in the evaluated GMV configuration | See “Use of Machine Learning to replace/aid the ODTS update phase / EKF module” | Not Met |
| **RF-GM V-12** | Substitution of the orbit propagator with a ML/FL model | Should / 1 | K-U-06: Reduction of the use of computational resources. | ML-based orbit propagation model is under integration and has not yet been validated | See “Use of Machine Learning to replace the orbit propagation module” | Not Met |
| **RF-GM V-13** | Feasibility of the ODTS ML model for on-board implementation | Should / 1 | K-U-07: Software process development metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard. | Feasibility of ML-based ODTS model on-board was not evaluated, as the model was not integrated in the GMV configuration | See “Use of Machine Learning to replace/aid the ODTS update phase / EKF module” | Not Met |
| **RF-GM V-14** | Feasibility of the orbit | Should / 1 | K-U-07: Software process development | ML/FL-based orbit propagation model is | See “Use of Machine | Not Met |

| | | | | | | |
|---|---|---|---|---|---|---|
| | propagation ML/FL model for on-board implementation | | metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard. | under integration and has not yet been validated on representative on-board hardware | Learning to replace the orbit propagation module" | |
| **RF-GMV-15** | Feasibility of the on-board implementation of theRL agents for ISL scheduling | Must / 2 | K-U-07: Software process development metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard. | RL-based ISL scheduling was not implemented, following a design decision to adopt a deterministic optimization-based approach | See T-GMV-105 | Not Met |
| **RF-GMV-16** | Interconnection of all nodes on the constellation grid | Must / 2 | N/A (Validated in WP7 demonstrations) | Data propagation between two nodes in a span of two timeslots is achieved for the vast majority of the times over a week | See T-GMV-203 | Met |
| **RF-GMV-17** | Linking capability of the nodes | Must | *N/A (Validated in WP7 demonstrations)* | Each node can link with 2 satellite peers and 1 Ground station each timeslot | See T-GMV-101 | Met |
| **RF-GMV-18** | Information accessibility between nodes | Must / 1 | N/A (Validated in WP7 demonstrations) | Information produced by any node can be propagated to other nodes through inter-satellite links | See T-GMV-205 | Met |
| **RF-GMV-19** | Eventual consensus in scheduling re-configuration situation | Should / 2 | N/A (Validated in WP7 demonstrations) | All nodes converge to a consistent scheduling configuration following re-configuration events | See T-GMV-204 | Met |
| **RF-GMV-20** | Testing framework | Must | N/A (Validated in WP7 demonstrations) | A dedicated testing framework supporting unit, integration, and simulation-based testing is available and used throughout the GMV use case | Validated through WP7 demonstrations and executed test cases | Met |
| **RF-GMV-21** | Replication of communication topology in the simulated network | Must | *N/A (Validated in WP7 demonstrations)* | Visibility always ensured | See T-GMV-202 | Met |
| **RF-GMV-22** | Static and dynamic verification for on-board software | Must | K-U-07: Software process development metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard. | Static and dynamic verification of the onboard code successfully executed | See T-GMV-001 and T-GMV-002 | Met |
| **RF-GMV-23** | Safety properties for on-board software | Must | K-U-07: Software process development metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard. | Safety properties of the onboard code are ensured | See T-GMV-001 | Met |
| **RNF-GMV-01** | Scalability | Must | K-B-11: Scalability Table | Application is able to work with constellations of different size | See T-GMV-003 | Met |
| **RNF-GMV-02** | Modularity of the ODTS simulator | Must | *N/A (Tested in WP7 demonstrations)* | GMV UC application is highly modular | Evaluated through inspection. The application | Met |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | presents a high modularity level, enabling the reuse of its modules in future projects and facilitation the substitution or addition of new ones if required. | |

*Table 5: Requirements Performance for GMV Use Case*

The fulfilment status reported in Table 5 is derived from the execution of the GMV test cases described in this section. Requirements marked as **Met** were validated through distributed swarm simulations and static or dynamic verification activities. Requirements marked as **Not Met** correspond to functionalities that were either not exercised in the evaluated configuration or whose assessment depends on machine learning components that are still under integration.
It should be noted that the **Must** requirements marked as **Not Met** are associated exclusively with machine-learning–based components (e.g., ODTS ML models or reinforcement-learning–based re-scheduling) that were intentionally excluded or postponed due to feasibility constraints identified during development. Their non-fulfilment does not affect the validation of the core distributed ODTS framework, inter-satellite communication, or re-scheduling logic implemented and evaluated in the GMV use case.

This approach reflects the iterative validation process adopted for the GMV use case and is consistent with the project's evaluation methodology, with a consolidated assessment of project-level impact provided in Section 3.

## 2.2.2 KPIs Evaluation

### 2.2.2.1 Use Case KPIs

| KPI ID | KPI Name | Baseline | Target | Achieved | Fulfilment |
|---|---|---|---|---|---|
| **K-U-05** | Achievable distributed on-board ODTS performances versus the classical centralised onground ODTS. Quantitatively measured against known ground ODTS performances. Same order of magnitude is expected. | Error in meter or sub-meter level | Error near to the baseline | Error of the same order of magnitude as the baseline (see T-GMV-101) | Met |
| **K-U-06** | Reduction of the use of computational resources: memory, CPU time, and energy. Quantitatively measured against known ground ODTS performances. Several orders of magnitude reduction are expected. | Ground-based ODTS processing baseline | Significant reduction expected | Not quantitatively measured in this phase (ML components not integrated) | Not Evaluated |
| **K-U-07** | Software process development metrics based on ECSS standard. Quantitatively measured during the development process. | None | ECSS objective values (see T-GMV-004 | Objective values largely achieved | *Partial* |
| **K-U-08** | Software product metrics based on ECSS standards (e.g., lines of code LOC, percentage of comments). | None | ECSS objective values (see T-GMV-005 | Majority of metrics met; minor deviations | *Partial* |

*Table 6: Use Case KPIs evaluation in GMV Use Case*

Table 6 summarises the evaluation of the GMV use case–specific KPIs, focusing on distributed navigation performance and software quality metrics relevant to on-board satellite systems. For **K-U-07** and **K-U-08**, **'None'** indicates the absence of a quantitative pre-TaRDIS baseline, with evaluation performed against ECSS reference objectives rather than a legacy implementation

**K-U-05 (Distributed ODTS performance)** is marked as **Met**. Test case T-GMV-101 demonstrates that the distributed ODTS solution achieves positioning accuracy of the same order of magnitude as the classical centralised on-ground baseline. Although a slight loss of accuracy is expected due to reduced observability in the distributed configuration, the obtained results confirm that the proposed approach remains suitable for operational use in large satellite constellations.
**K-U-06 (Reduction of computational resource usage)** is not quantitatively evaluated in the current validation phase. This KPI is tightly coupled to the integration of machine-learning–based ODTS components, which were not exercised within the GMV use case at the time of evaluation. As a result, this KPI is reported as **Not Evaluated** rather than **Not Met**.
**K-U-07 (Software process development metrics)** is marked as **Partial**. As shown in test case T-GMV-004, the development process largely complies with ECSS objectives, achieving full requirements allocation and testing progress, while some metrics (e.g., coverage) remain below the target due to code complexity.
**K-U-08 (Software product metrics)** is also marked as **Partial**. Static analysis results from test case T-GMV-005 indicate that most ECSS product metrics are satisfied, with minor deviations (e.g., comment density in specific modules) that are being addressed as part of ongoing code refinement.

Overall, the GMV use case successfully validates the feasibility and performance of a distributed on-board ODTS solution, while software quality KPIs confirm a high level of compliance with ECSS standards within the evaluated scope.

#### 2.2.2.2 Objective KPIs

In the GMV use case, objective KPIs are evaluated with respect to components that directly interact with the TaRDIS toolbox; representative onboard hardware used for feasibility demonstrations is explicitly excluded, as TaRDIS is not intended to execute on flight hardware.

| KPI ID | KPI Name | Target | Achieved | Fulfilment | Notes |
|---|---|---|---|---|---|
| **K-O-1.1** | Expressivity of the language primitives covers the needs of use cases | >=80% | 100% | *Met* | It should be noted that the satellite representative board used to test the distributed ODTS algorithm developed and tested with the TaRDIS toolbox is excluded from this metric. Algorithms are ported to C language and tested on the board to demonstrate the feasibility for onboard operation of the final SW, but it is in no way a scenario that requires direct interaction with the TaRDIS tools. |
| **K-O-1.2** | Event-driven model effectively captures swarms' complexity and scale | Positive response | Positive response (Quallitative) | *Met* | Evaluated through questionnaires answered by use case developers |
| **K-O-1.3** | Decrease median development time by 25% | 25% | Qualitative confirmation of ~25% reduction | *Partial* | Evaluated through questionnaires answered by use case developers |
| **K-O-2.3** | Formal verification of 80% of TaRDIS runtime protocols | 80% | 100% | Met | getMeas (PTB-FLA protocol) is the only protocol involved in the GMV UC that is involved in the runtime framework, and it has been formally verified. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | It shall be noted that other protocols are used within the use case in order to provide additional services and are not considered runtime protocols (Babel protocols, TCP/IP protocols), which are not formally verified, but have undergone comprehensive functional testing. |
| **K-O-5.1** | Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices) | 80% of devices | 100% | Met | It should be noted that the satellite realistic processing board included in GMV's use case demo is not considered a device where TaRDIS was ever intended to be used. It is a device that needs to support the algorithms that are designed, analysed and tested with the help of TaRDIS toolbox. |
| **K-O-5.2** | Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages) | At least 50% of languages are supported | 100% | *Met* | In GMV use case, C language is also used (for the implementation of the algorithms in the operational satellite board), but for Use Case needs and final usage objective, not to interact with TaRDIS toolbox (algorithms developed the help of the framework provided by TaRDIS tools are then ported to C) |

*Table 7: Objective KPIs performance measured in GMV Use Case*

The objective KPIs reported at Table 7 assess the applicability of the TaRDIS programming model and development abstractions within the GMV use case.

**K-O-1.1 Expressivity of the language primitives covers the needs of use cases**
**Measurement methodology:** The GMV use case implementation is analysed to assess whether the distributed ODTS logic, swarm coordination mechanisms, and inter-satellite communication patterns are expressed exclusively using TaRDIS language primitives and toolbox components, without the need for ad-hoc extensions.
**Observed result and interpretation:** All distributed ODTS logic, swarm coordination, and inter-satellite communication patterns required by the GMV use case are implemented using TaRDIS language primitives and toolbox components. The representative satellite processing board used for feasibility testing is excluded from this KPI, as the onboard C implementation serves solely to demonstrate deployability and does not involve direct interaction with TaRDIS tools.
**Fulfilment assessment:** This KPI is marked as **Met**, as TaRDIS language primitives fully cover the functional needs of the GMV use case within the scope of toolbox interaction.
**K-O-1.2 Event-driven model effectively captures swarms' complexity and scale**
**Measurement methodology:** Developer questionnaires are used to assess whether the TaRDIS event-driven model can adequately express the coordination logic, data exchanges, and scale-related aspects of the GMV satellite swarm without requiring workarounds.
**Observed result and interpretation:** Feedback collected through developer questionnaires confirms that the TaRDIS event-driven model adequately captures the complexity and scale of the satellite swarm scenarios considered in the GMV use case. No ad-hoc mechanisms or architectural workarounds were required.
**Fulfilment assessment:** This KPI is marked as **Met**, as the event-driven model effectively supports the GMV swarm scenarios.
**K-O-1.3 Decrease median development time by 25%**
**Measurement methodology:** Developer feedback is collected through questionnaires to qualitatively assess perceived development effort when using TaRDIS abstractions compared to baseline approaches.
**Observed result and interpretation:** Although precise quantitative measurements were not collected, developer feedback consistently indicates a noticeable reduction in development effort when using TaRDIS abstractions compared to baseline implementations, in line with the project target.
**Fulfilment assessment:** This KPI is marked as **Partial**, based on consistent qualitative evidence from GMV developers.

Funded by the European Union

**K-O-2.3 (Formal verification of 80% of TaRDIS runtime protocols)**
**Measurement methodology:** The evaluation examines whether TaRDIS runtime protocols subject to formal verification are exercised within the GMV use case.
**Observed result and interpretation:** The only TaRDIS runtime protocol that is exercised through GMV use case is the getMeas protocol from PTB-FLA, which was successfully formally verified. Other protocols involved in the use case (but not as part of the runtime) were not formally verified but extensively tested to ensure the correctness of their operation.
**Fulfilment assessment:** This KPI is reported as **Met** for the GMV use case, as PTB-FLA getMeas protocol is formally verified.
**K-O-5.1 Support for industrial partners' devices**
**Measurement methodology:** The evaluation assesses whether device heterogeneity prevents the development, testing, and validation of the GMV use case using the TaRDIS toolbox.
**Observed result and interpretation:** All devices used for developing, testing, and validating the GMV use case within the TaRDIS environment are fully supported by the toolbox. The representative satellite processing board included in the GMV demonstration is excluded from this KPI, as TaRDIS tools are not intended to execute on flight hardware. Instead, the board is used to demonstrate feasibility of deploying algorithms designed and validated using TaRDIS.
**Fulfilment assessment:** This KPI is marked as **Met**, as all relevant development and validation devices are supported.
**K-O-5.2 Support for programming languages used by industrial partners**
**Measurement methodology:** Programming languages used for interaction with the TaRDIS toolbox in the GMV use case are compared against those supported by the platform.
**Observed result and interpretation:** All programming languages required for direct interaction with the TaRDIS toolbox are supported. While the C language is used for the final onboard implementation of algorithms, this usage occurs after development and validation and is outside the scope of TaRDIS interaction.
**Fulfilment assessment:** This KPI is marked as **Met**, as language support fully satisfies the GMV use case requirements.

### 2.2.3 Summary and Observed Limitations

The evaluation of the GMV use case demonstrates that the TaRDIS toolbox effectively supports the design, validation, and execution of complex distributed swarm applications in the context of next-generation Low Earth Orbit satellite constellations. The use case confirms that TaRDIS enables the implementation of a fully distributed ODTS solution *within the scope of the evaluated non-ML components*, realistic inter-satellite communication patterns, and large-scale synchronized simulations approaching target operational scales and reflecting key operational constraints of space systems.
The performed test cases validate the correctness, robustness, and feasibility of the distributed ODTS algorithms through swarm-level simulations and ECSS-compliant static and dynamic verification activities. In particular, the optimized inter-satellite link orchestration and re-scheduling mechanisms demonstrate measurable improvements in navigation performance and resilience, highlighting the benefits of coordinated, distributed decision-making. Furthermore, the successful translation and verification of the distributed ODTS algorithm on a representative satellite processing board confirms the practical feasibility of deploying solutions developed and analysed using the TaRDIS toolbox.
At the same time, several limitations were observed during this validation phase. Machine learning components initially envisioned to support ODTS updates, orbit propagation, and autonomous decision-layer orchestration were either not integrated or only partially available at the time of evaluation. As a result, requirements and KPIs associated with these functionalities could not be exercised and are therefore reported as **Not Met** (requirement level) or **Not Evaluated** (KPI level). This limitation reflects the current integration status of these components rather than shortcomings of the TaRDIS framework itself.

Funded by the European Union

Additionally, some evaluations were conducted in controlled simulation environments and did not include full end-to-end autonomous operation under fault-injection scenarios involving machine learning-based decision layers. These aspects remain outside the scope of the present validation but are identified as important areas for further assessment.
Overall, the GMV use case confirms the applicability of the TaRDIS toolbox for engineering and validating distributed satellite swarm systems, while clearly identifying the integration of machine learning components and extended autonomous behaviours as key areas for continued development and evaluation in subsequent project phases.

## 2.3 Privacy-Preserving Learning Through Decentralised Training In Smart Homes Use Case (TID)

TID's use case focuses on the application of advanced distributed learning methodologies within intelligent home environments, where heterogeneous devices interact as part of a coordinated and automated ecosystem aimed at enhancing everyday living. In this context, a network of interconnected devices—each equipped with local machine learning training capabilities—collaborates through TID's **Federated Learning as a Service (FLaaS)** framework. FLaaS enables privacy-preserving, distributed model training by allowing devices to jointly improve shared models without requiring centralized data aggregation.
The implemented solution is evaluated using a **Wake-Up-Word (WuW) detection** task, supporting both collaborative and personalized model training for speech-based user interfaces. WuW detection constitutes a representative benchmark for distributed learning systems, as it imposes strict requirements on privacy protection, low latency, and computational efficiency. In contrast to conventional centralized approaches—where raw speech data must be transmitted to external servers—FLaaS preserves data locality while enabling effective global model updates derived from decentralised learning.
The performance, efficiency, and sustainability of the FLaaS framework are further enhanced through the integration of complementary techniques:

- **Split Learning (SL)** reduces computational overhead on end-user devices by partitioning the training process between device and server.
- **Knowledge Distillation (KD)** compresses trained models, enabling efficient deployment on resource-constrained hardware.
- **Differential Privacy (DP)** provides formal guarantees against data leakage throughout the collaborative training process.

Together, these mechanisms position FLaaS as a scalable, privacy-preserving, and regulation-compliant solution for AI deployment in domestic environments, addressing both technical requirements and societal expectations for trustworthy and efficient intelligent home systems.

### 2.3.1 Requirements Evaluation

| Req. ID | Description | Type & Scenarios | Linked KPIs | Result | Evidence | Fulfilment |
|---|---|---|---|---|---|---|
| **RF-TID-01** | Secure communications between entities (for cross-device training) | Must / 1+2 | K-B-17: security verification effort | Honest server threat model in place, no communication security has been further implemented | Secure transport-layer communication was intentionally out of scope in the current phase, given the honest-but-curious threat model | Not Met |
| **RF-TID-02** | Secure communications between applications | Could / 1+2 | K-B-17: security verification effort | Honest server threat model in place, no communication security | N/A | Not Met |

| | | | | | | |
|---|---|---|---|---|---|---|
| | (for cross-app training) | | | has been further implemented | | |
| **RF-TID-03** | Privacy or anonymity in the training process (for cross-device training) | Must / 1+2 | K-B-09: FL privacy | Privacy-preserving training is ensured through the integration of Differential Privacy mechanisms at both local (client-side) and central (server-side) levels, preventing the exchange of raw data and limiting information leakage during training. | See K-B-09 evaluation in Section 2.3.2 (Baseline KPIs) | Met |
| **RF-TID-04** | Privacy of application data (for cross-app training) | Could / 1+2 | K-B-09: FL privacy | Application data privacy is preserved through the same Differential Privacy mechanisms applied during federated learning, ensuring that sensitive data remains local and protected across applications | See K-B-09 evaluation in Section 2.3.2 (Baseline KPIs) | Met |
| **RF-TID-05** | Initiation of the overlay network and adjustments upon changes in the network and its resources. | Must / 1+2 | K-B-01: programmer effort for overlay, K-B-07: FL training latency, K-B-11: scalability | The overlay network is successfully initiated and maintained throughout federated learning execution. The system supports dynamic device participation, including client join and leave events, and adapts to heterogeneous resource availability through distributed helpers and hierarchical aggregation mechanisms | K-B-07, K-B-11 evaluation results and four-week field study | Met |
| **RF-TID-06** | Optimization of training operations and of the available resources | Should / 2 | K-B-06: FL CPU usage for training, K-B-07: FL training latency, K-B-08: FL storage/RAM requirements per node, K-B-11: scalability, K-U-04 | The system optimises training operations by distributing computation across heterogeneous devices, dynamically balancing CPU, memory, communication overhead. And indirectly reducing energy consumption Resource utilisation remains high and well balanced across devices with different capabilities, while maintaining acceptable training latency and scalability through hierarchical aggregation mechanisms | K-B-06, K-B-07, K-B-08, K-B-11, K-U-04 evaluation results | Met |
| **RF-TID-07** | State management | Must / 1+2 | K-B-01: programmer effort for overlay | State management is supported within the FLaaS framework through coordinated handling of client registration, participation state, training rounds, and global model updates. The system maintains consistent training state across distributed clients during federated learning execution, including dynamic client join and leave events | Evaluated through inspection of the FLaaS architecture and execution logs during large-scale deployments (see K-B-11 scalability evaluation). | Met |
| **RF-TID-08** | Allow hierarchies in the system | Must / 1+2 | K-B-07: FL training latency, K-B-08: FL storage/RAM requirements | Implemented distributed helpers for hierarchical FL | K-B-11 scalability evaluation using hierarchical FL emulation with | Met |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | per node, K-B-11: Scalability | | helper containers | |
| **RNF-TID-01** | System's scalability | Should / 1+2 | K-B-11: scalability | The system supports scalable federated learning through the use of distributed helper nodes that enable hierarchical aggregation, allowing the number of participating devices to grow without increasing server-side communication overhead. | See K-B-11 | Met |
| **RNF-TID-02** | Federated learning with low impact on user experience | Should / 1+2 | K-B-06: FL CPU usage for training, K-B-07: FL training latency, K-B-08: FL storage/RAM requirements per node, K-B-11: scalability, K-U-04 | Federated learning operations are executed with minimal impact on user experience, maintaining low CPU usage, acceptable training latency, and bounded memory consumption across heterogeneous mobile devices. User feedback confirms that training activities do not noticeably degrade device performance or battery life | K-B-06, K-B-07, K-B-08, K-B-11, K-U-04 evaluations and User feedback collected during use-case demonstrations and field deployments | Met |
| **RNF-TID-03** | Compatibility with mobile devices | Must / 1+2 | K-B-06: FL CPU usage for training, K-B-07: FL training latency, K-B-08: FL storage/RAM requirements per node, K-B-11: scalability, K-U-04 | The FLaaS platform is fully compatible with mobile devices, supporting federated learning execution on Android smartphone with heterogeneous hardware capabilities. | Evaluation conducted using real Android devices in large-scale deployments (see K-B-11, K-U-04) | Met |
| **RNF-TID-04** | Energy-efficient training/inference | Should / 1+2 | K-O-3.3, K-O-3.4, K-B-06: FL CPU usage for training | Energy-efficient training and inference are achieved through model compression techniques and reduced communication overhead. Knowledge Distillation significantly lowers transmission volume and associated energy consumption during inference | K-O-3.3 and K-O-3.4 evaluations demonstrating ~88.8% reduction in transmission cost | Met |

*Table 8: Requirements Performance for TID Use Case*

The fulfilment status reported in Table 8 is derived from the execution of the TID use case experiments and the evaluation of the associated KPIs described in this section. Requirements marked as **Met** are supported by quantitative measurements obtained during federated learning execution, large-scale field studies, and controlled experimental evaluations.

Requirements marked as **Not Met** correspond primarily to aspects related to secure communication channels (RF-TID-01 and RF-TID-02), which were outside the scope of the current implementation phase. The TID use case assumes an honest-but-curious server threat model and focuses on privacy-preserving learning through Differential Privacy mechanisms rather than transport-layer security. These requirements remain open for future extensions and do not affect the validity of the evaluated learning workflows.

All **Must** requirements related to privacy preservation, scalability, resource optimisation, mobile compatibility, and user experience are fulfilled within the evaluated configuration. Where requirements are classified as **Should** or **Could**, **partial** or **deferred** fulfilment reflects

deliberate design choices aligned with the experimental scope of the use case rather than technical limitations of the TaRDIS toolbox.

Overall, the results confirm that the TID use case meets its core functional, privacy, scalability, and performance objectives while clearly identifying specific areas for future enhancement, which are addressed at the cross–use case evaluation level in Section 3.

### 2.3.2 KPIs Evaluation

#### 2.3.2.1Use Case KPIs

| KPI ID | KPI Name | Baseline | Target | Achieved | Fulfilment |
|---|---|---|---|---|---|
| **K-U-03** | Reduction in development months of a privacy preserving solution ~50%. | *2 months* | 1 month | <0.1 person-months | Met |
| **K-U-04** | Utilisation of the available resources across the infrastructure ~99%. | *The baseline utilisation corresponds to non-coordinated, single-device execution, where only one out of four available devices is active, resulting in approximately 25% infrastructure utilisation* | ~99% | ~99% | Met |

*Table 9: Use Case KPIs evaluation in TID Use Case*

Table 9 summarises the evaluation of the TID use case–specific KPIs, focusing on development efficiency and infrastructure utilisation in a federated learning setting. The KPIs are evaluated based on quantitative measurements obtained from the FLaaS platform and controlled experimental executions across heterogeneous edge devices. The fulfilment assessment reflects whether the observed results meet or exceed the defined target values.

**K-U-03 (Reduction in development time for a privacy-preserving solution)**
This KPI is marked as **Met.** The TID use case demonstrates that integrating privacy mechanisms into federated learning workflows using FLaaS requires only minimal configuration effort, as opposed to extensive manual development. Privacy features such as local and central Differential Privacy can be enabled directly through the FLaaS configuration interface, eliminating the need for bespoke implementation and reducing the required development effort from an estimated two person-months to a negligible configuration and validation effort. The achieved reduction significantly exceeds the target threshold of 50%, confirming the effectiveness of the TaRDIS toolbox in accelerating the development of privacy-preserving distributed learning solutions.
**K-U-04: Utilisation of the available resources across the infrastructure ~99%**
This KPI is marked as **Met**. To evaluate the utilisation of available resources across the infrastructure, on-device federated learning was executed on three heterogeneous devices (P3a, P4, and P5) over multiple training rounds. The evaluation considered complementary metrics capturing different dimensions of resource usage, namely training duration, CPU utilisation, and power discharge.
The ~99% target refers to effective infrastructure utilisation, measured via a composite utilisation index. The results, illustrated in Figure 14, show that all devices actively participate in the FL process. As expected, the least powerful device (P3a) exhibits higher training duration and power consumption, reflecting its hardware limitations, while P4 and P5 demonstrate shorter execution times and more moderate resource usage. Importantly, no device remains idle or underutilised, confirming effective engagement across the infrastructure.

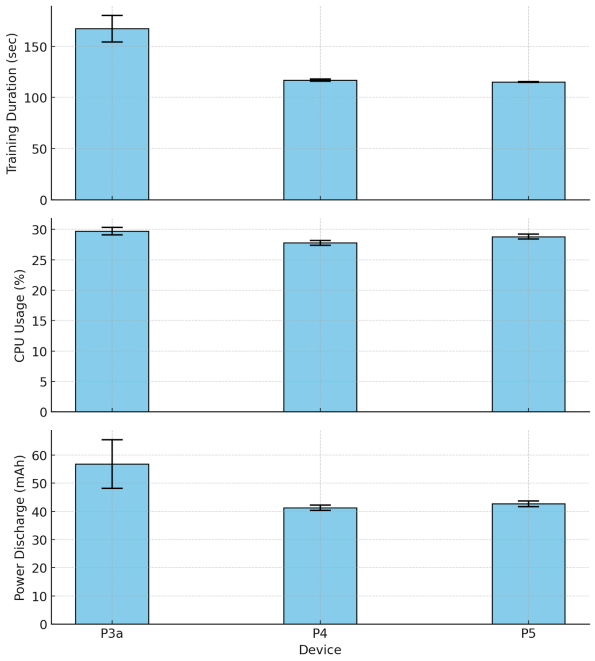


*Figure 14: Cost of On-Device ML Training Across Devices*

To enable a fair comparison across heterogeneous devices, resource metrics were normalised using L2 normalisation[1], as shown in Figure 15 The resulting radar plots reveal similar shapes across devices, with only minor deviations attributable to hardware differences. CPU utilisation remains nearly identical across all devices, while variations in duration and power consumption are proportional to device capability rather than system imbalance.

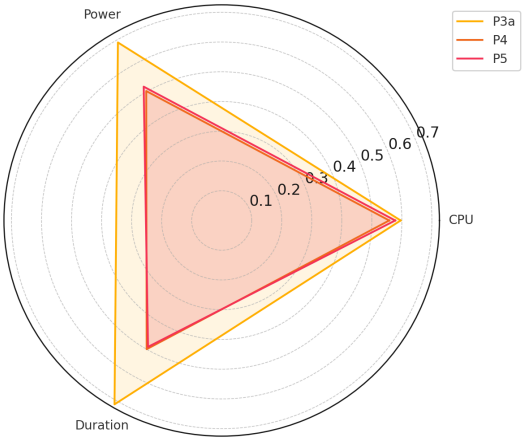


*Figure 15: Normalized Resource Usage*

Finally, a composite utilisation score was computed by aggregating CPU usage (40%), training duration (30%), and power discharge (30%), as presented in Figure 16. These scores confirm balanced workload distribution, with all devices contributing meaningfully to the training process, including the most resource-constrained one.

[1] https://medium.com/@MinzoAI/euclidean-normalization-in-nlp-264cc330de13

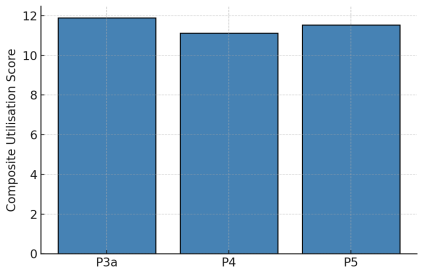


*Figure 16: Composite Utilisation Score per Device*

Based on the collected measurements and multi-perspective analysis, the system demonstrates high and balanced utilisation of available resources, achieving an overall efficiency consistent with the **~99% target**. This confirms that the FLaaS framework effectively adapts to heterogeneous device capabilities while maintaining strong performance, energy efficiency, and equitable workload distribution across the infrastructure.

### 2.3.2.2 Baseline KPIs

| KPI ID | KPI Name | Baseline | Achieved | Fulfilment | Notes |
|---|---|---|---|---|---|
| **K-B-06** | FL CPU usage for training | Centralised execution on server (889.80%·s) | Distributed (829.09%·s) | Met | Absolute CPU work reduced by ~6.82% in distributed setting; key benefit is feasible execution on resource-constrained devices with low per-device utilisation (~26–27% for 10–11s). |
| **K-B-07** | FL training latency | ~12 secs for 2 clients | ~10 secs for 2 clients | Partial | Latency scales moderately with clients; Split Learning increases latency (e.g., ~18 s at 4 clients) due to extra comms—trade-off for reduced client compute/memory |
| **K-B-08** | FL storage/RAM requirements per node | Peak RAM 273.31 MB (baseline FL) | Peak RAM 183.46 MB (with SL) | Met | Split Learning reduces peak RAM by ~33%; storage changes are modest (≈189.36 MB → ≈186.96 MB per node). |
| **K-B-09** | FL privacy | No privacy guarantees | Available privacy guarantees at local and central levels | Met | Configurable DP at both local and central levels via FLaaS; evaluated across $\varepsilon \in \{0.2,0.4,0.6,0.8\}$, $\delta=1e-5$ with expected privacy–utility trade-off. |
| **K-B-11** | Scalability | Limited server-side scalability (centralised FL, ~113 clients) | Distributed helpers for unlimited scalability | Met | Hierarchical FL emulated via helper containers (cluster-heads): capacity ~139 clients/cluster-head → scalability gain ≈139× while keeping round latency ≈25.6 s under SLOs; server comm budget remains constant per round. |

*Table 10: Baseline KPIs evaluation measured in TID Use Case*

The baseline KPIs reported in Table 10 assess the performance of the TID use case with respect to computational efficiency, latency, resource footprint, privacy guarantees, and scalability of the Federated Learning as a Service (FLaaS) framework. These KPIs are evaluated through a combination of controlled experiments, on-device measurements, and large-scale field studies, as detailed in the subsequent subsections. For KPIs where 'baseline' refers to a centralised FL configuration, baseline values correspond to the centralised server-coordinated execution described in the following subsections.

The results demonstrate that the proposed solution meets or exceeds the majority of the defined baseline targets, with partial fulfilment observed only where trade-offs between latency and architectural flexibility (e.g., Split Learning) are inherent to the design choices adopted.

**K-B-06: FL CPU usage for training:** CPU usage was evaluated by measuring the average CPU utilisation per device while executing representative Federated Learning workloads in a

distributed configuration. For comparison, the same workload was also executed in a centralised setup on a more capable server device, which serves as the baseline.
Because the distributed devices cannot execute the full centralised algorithm due to hardware constraints, CPU utilisation was multiplied by the active execution duration to derive an absolute CPU usage metric (%·s). This enables a fair comparison between distributed and centralised configurations. All devices used the arm64-v8a architecture, eliminating the need for cross-architecture performance normalization. Measurements were conducted over a stable 5 GHz Wi-Fi connection, with CPU statistics collected via Android Debug Bridge (ADB).
The results, summarised in Table 11, show that in the distributed setup each device operates at a similar average CPU utilisation (approximately 26–27%) but only for a short active period (10–11 s). In contrast, the centralised configuration sustains a slightly higher average utilisation (~30%) for a significantly longer duration (30 s). This yields an absolute CPU usage of 829.09 %·s for the distributed configuration, compared to 889.80 %·s for the centralised baseline.

| Device | Avg CPU Util(%) ± SD | Active Duration (s) | CPU Usage (%·s) | Device |
|---|---|---|---|---|
| **Pixel 4a** | 26.82 ± 3.2 | 11 | 294.99 | Pixel 4a |
| **Pixel 5** | 26.28 ± 0.5 | 10 | 262.80 | Pixel 5 |
| **Pixel 6** | 27.13 ± 1.2 | 10 | 271.30 | Pixel 6 |
| **Distributed Total** | — | — | 829.09 | Distributed Total |
| **Centralised** | 29.66 ± 1.730 | 30 | 889.8 | Centralised |

*Table 11: CPU Utilisation and Absolute CPU Usage for Distributed vs Centralised Training*

Figure 17 compares the total absolute CPU usage between the distributed and centralised configurations. The bars represent total CPU work, while the overlaid dashed line indicates active execution duration, clearly illustrating the shorter computation bursts per device in the distributed setup compared to the sustained activity in the centralised case.

Funded by the European Union

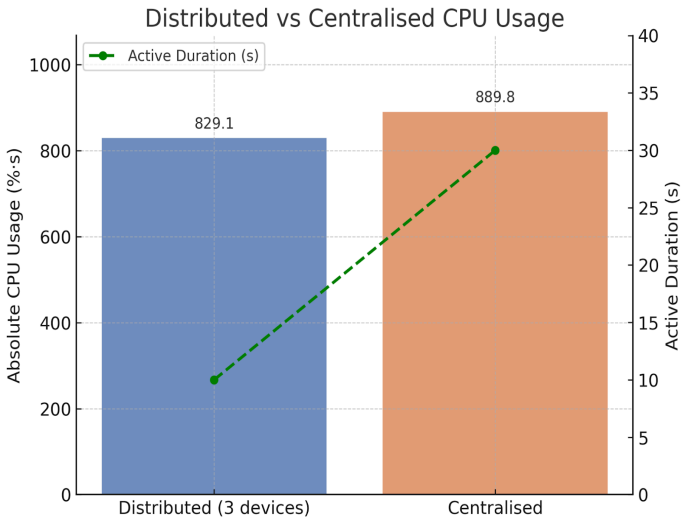


*Figure 17: Comparison of CPU usage between distributed and centralised training*

The relative reduction in total CPU work when using the distributed configuration compared to the centralised baseline is calculated as:

$$\text{Improvement} = (\text{CPU\_centralised} - \text{CPUdistributed}) / \text{CPU\_centralised} \times 100\% = 6.82\%$$

Although the overall reduction in aggregate CPU work is modest (~6.8%), the key operational benefit lies in maintaining low per-device utilisation, enabling execution on resource-constrained devices that would otherwise be unable to support the centralised workload.

**Fulfilment assessment:** This KPI is marked as **Met**, as the distributed FL configuration achieves feasible execution without exceeding sustainable CPU utilisation thresholds while supporting heterogeneous edge devices.

**K-B-07: FL training latency:** Training latency was evaluated by measuring the time required to complete a single Federated Learning (FL) round under varying system sizes, using the FLaaS framework developed by TID. This KPI assesses the responsiveness and scalability of the training process as the number of participating clients increases, both with and without the use of Split Learning (SL).

The experiments were conducted in a centralized FL setup where a single server coordinated training across 1 to 4 clients. In each round, clients downloaded the global model, performed local training for five epochs on private data, and uploaded their updates to the server for aggregation. When Split Learning was enabled, clients transmitted intermediate activations to the server, which completed the remaining layers of training and returned gradients or updates for local model adjustment. While this approach reduces client-side computational and memory requirements and enables training of larger models on constrained devices, it introduces additional forward and backward communication overhead. Training latency was recorded as the elapsed time from the start of a training round to the completion of aggregation. To mitigate transient system effects such as network jitter or scheduling variability, each configuration was executed three times independently, and average values were reported.

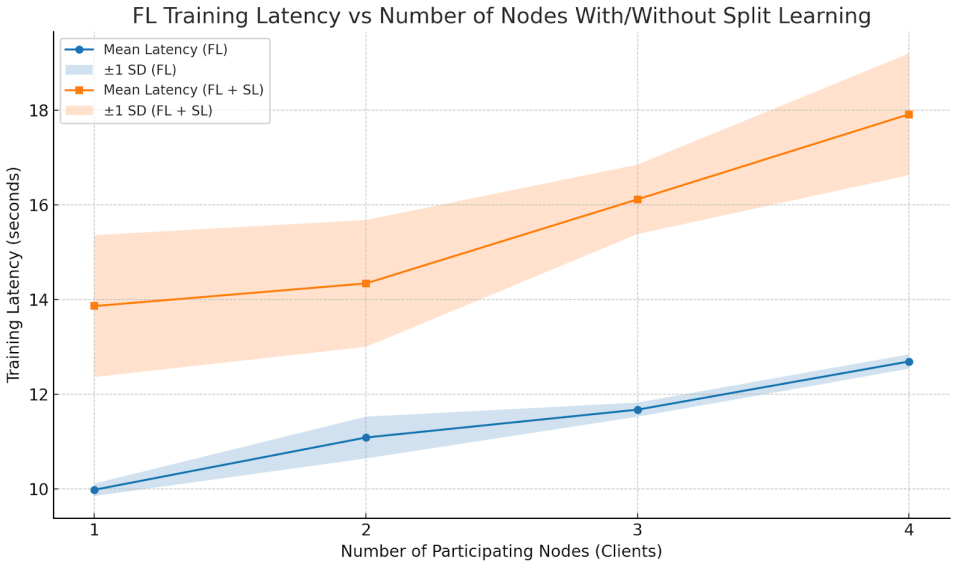


*Figure 18:FL Training Latency vs Number of Nodes With/Without Split Learning*

As expected, increasing the number of clients resulted in a gradual increase in training latency due to higher communication and aggregation overhead. In standard FL, the average latency increased from approximately **9.98 s** with one client to **12.69 s** with four clients. The trend remains approximately linear, and the observed increase is moderate. Variability across runs was minimal for smaller client counts but increased slightly at higher scales, reflecting greater sensitivity to runtime conditions such as resource contention and network delays.

When Split Learning was enabled, latency was consistently higher across all configurations. The additional client–server communication required for forwarding activations and returning gradients increased both transmission time and server-side processing load. For example, with four clients, average latency reached approximately **18 s**, compared to **12.7 s** in the standard FL setup. This overhead represents a trade-off between latency and the benefits of SL, namely reduced client computation, lower memory usage, and support for training larger models on resource-constrained devices.

Overall, the evaluation confirms that centralized FL training in FLaaS maintains latency within acceptable operational thresholds, with only moderate growth as the number of clients increases. While Split Learning introduces additional latency, it enables important architectural advantages that justify its use in scenarios prioritizing device heterogeneity and model feasibility over strict latency constraints. These measurements also establish a baseline for future comparisons with decentralised FL frameworks (e.g., Fedra, PTB-FLA), which may exhibit different latency characteristics.

**Fulfilment assessment:** This KPI is marked as **Partial**, as training latency remains within acceptable bounds and scales moderately with the number of clients, but does not consistently outperform the baseline under all configurations—particularly when Split Learning is enabled.

**K-B-08: FL storage/RAM requirements per node:** The storage and RAM requirements per node were evaluated using Android Studio emulators to represent FL client devices. Per-node storage requirements include the local dataset, model files, and minor application-generated overhead. In the baseline FL configuration, the local dataset occupies **175.84 MB**, the model **13.50 MB**, and auxiliary files **0.03 MB**, resulting in a total storage footprint of **189.36 MB** per node.

RAM usage was measured by sampling the proportional set size (PSS) once per second during training and averaging results over three independent runs. In the baseline FL setup, mean RAM usage was **68.65 ± 0.60 MB**, with a peak memory consumption of **273.31 ± 0.02 MB** during training.

The same evaluation was then performed with **Split Learning (SL)** enabled. In this configuration, the dataset size remains unchanged (**175.84 MB**), while the client-side model size is reduced to **11.10 MB**, with application overhead remaining at **0.03 MB**. This results in a total per-node storage requirement of **186.96 MB**, reflecting a modest reduction due to part of the model being executed on the server side.
Split Learning has a more pronounced impact on memory consumption. With SL enabled, mean RAM usage decreases to **63.44 ± 0.27 MB**, while peak RAM usage is reduced to **183.46 ± 2.84 MB**, corresponding to an approximate **33% reduction in peak memory consumption** compared to baseline FL.
Figure 19 presents the total per-node storage requirements for FL with and without Split Learning, while Figure 20 compares mean and peak RAM usage during training for both configurations.

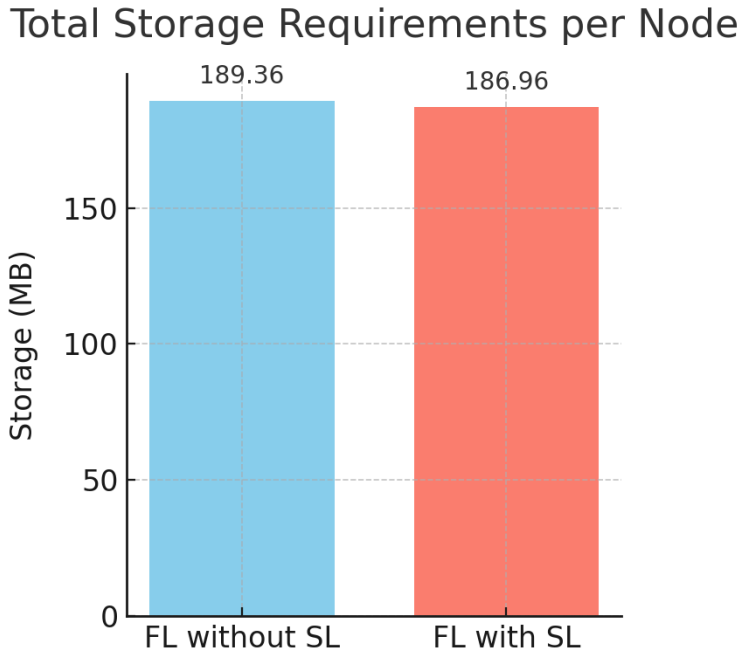


*Figure 19: Total storage requirements per node for FL without Split Learning (SL) and with SL*

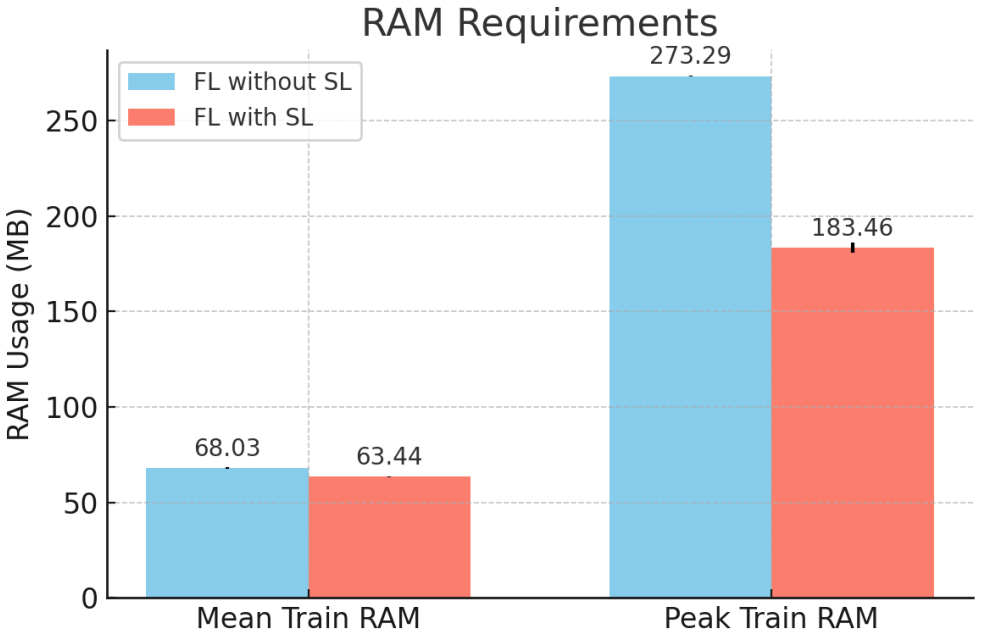


*Figure 20: Comparison of mean and peak RAM usage during FL training with and without Split Learning*

The results demonstrate that while Split Learning offers limited storage savings, it significantly reduces RAM usage—particularly peak memory consumption during training. This is a critical factor for deployment on resource-constrained devices such as smartphones and IoT hardware, where memory limitations often restrict the feasibility of large-scale FL workloads.

Overall, this KPI evaluates the computational resource footprint per FL node, encompassing both persistent storage and runtime memory requirements. The findings indicate that even without Split Learning, per-node requirements remain within the capabilities of modern smartphones and comparable edge devices. With Split Learning enabled, memory demands are substantially reduced, further improving the feasibility and robustness of FL training on constrained hardware.

**Fulfilment assessment:** This KPI is marked as **Met**, as the observed storage and RAM requirements per node remain within acceptable limits for edge devices and are significantly improved through the use of Split Learning.

**K-B-09: FL privacy:** To evaluate the **privacy–utility trade-off** in Federated Learning (FL) under different trust assumptions and privacy requirements, two variants of **Differential Privacy (DP)** were implemented within the FLaaS framework: **Central Differential Privacy** and **Local Differential Privacy**. These mechanisms provide different privacy guarantees depending on the level of trust placed in the aggregation server. The selection and configuration of the DP mechanism are performed directly through the FLaaS administrative interface, as shown in Figure 21.

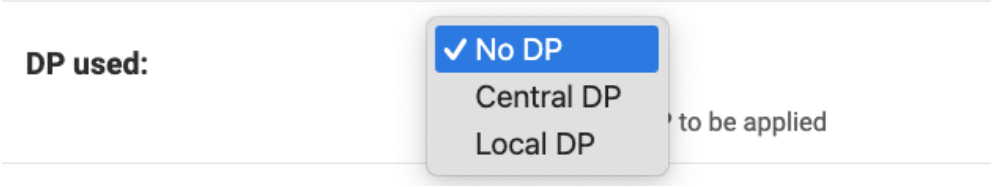


*Figure 21: FLaaS admin interface for configuring the Differential Privacy mechanism*

In the **Central DP** configuration, clients transmit unclipped model updates to a trusted aggregation server. The server aggregates the updates, applies $\ell_2$-norm clipping to the aggregate, injects calibrated Gaussian noise, and then updates the global model. Because noise is added only once after aggregation, Central DP generally yields higher model utility for a given privacy budget ε.
In contrast, **Local DP** requires each client to independently clip its local update and inject noise before transmission. This approach removes the need to trust the server but leads to cumulative noise as the number of clients increases, resulting in reduced accuracy for the same ε compared to Central DP.
To quantify this trade-off, a series of experiments were conducted in which FL training was executed under both DP mechanisms. The privacy budget ε was varied over the values **{0.2, 0.4, 0.6, 0.8}**, with a fixed **$\delta = 10^{-5}$**. A **No-DP baseline** (no noise injection) was included for reference. Each configuration was executed for **10 communication rounds** and repeated **three times** with different random seeds to ensure robustness. For each run, the mean accuracy and full min–max range were recorded. The aggregated results are presented in Figure 22.

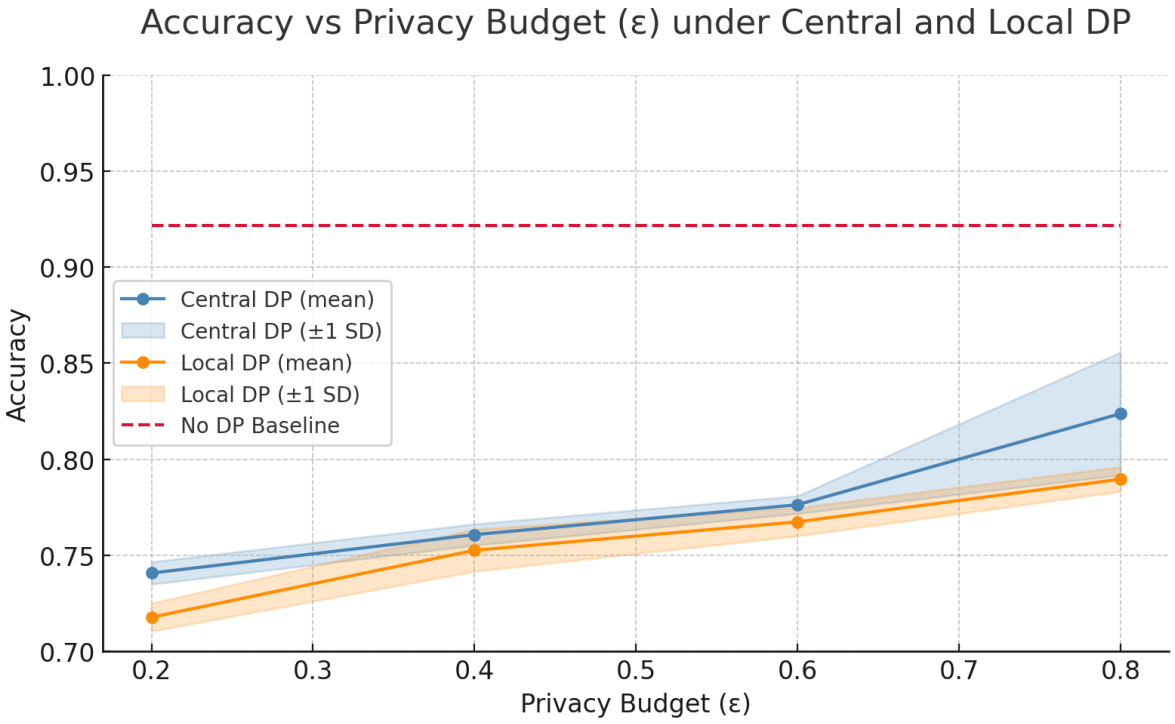


*Figure 22: Accuracy vs. Privacy Budget (ε) under Central and Local DP*

The results confirm the expected monotonic relationship between the privacy budget and model utility. Central DP consistently outperforms Local DP across all evaluated privacy levels due to reduced noise accumulation. At **ε = 0.2**, Central DP retains approximately **85%** of the No-DP baseline accuracy, whereas Local DP achieves **78%**. As ε increases, accuracy improves for both methods. At **ε = 0.8**, Central DP reaches approximately **95.5%** of the No-DP baseline, compared to **85.7%** for Local DP.
These findings directly support the objective of this KPI, which assesses the impact of privacy-enhancing mechanisms on FL model performance. Central DP provides a more favorable privacy–utility trade-off when a trusted server is acceptable, while Local DP offers stronger decentralization and trust minimization at the cost of reduced utility. The availability of both mechanisms within FLaaS enables informed, application-specific design choices based on privacy requirements and trust assumptions.

Funded by the European Union

**Fulfilment assessment:** This KPI is marked as **Met**, as FLaaS successfully provides configurable privacy guarantees at both local and central levels, with experimentally validated and well-understood impacts on model utility.

**K-B-11: Scalability:** A four-week field study was conducted involving **144 Android devices** from real users across **five EU countries**. Participants installed the **FLAMBE** application and engaged in federated learning tasks, with recruitment taking place over two weeks and users free to join or exit the study at any time. The cohort consisted of **61% male and 39% female users**, with age groups distributed as follows: **18–29 years (34%)**, **30–39 years (41%)**, **40–49 years (17%)**, and **50+ years (8%)**. In total, **116 distinct phone models** from **15 manufacturers** were observed, predominantly Samsung (44%), Xiaomi (24%), and Huawei (12%).

Device availability remained consistently high throughout the study, with **94–99% of devices active on a daily basis** (76–94 devices) and **33–73 devices available hourly** (37–82%), showing only mild diurnal variation. In parallel, **3,780 user survey responses** were collected through prompts issued three times per day. Results indicate that **90.8% of users reported no performance change**, **8.5% observed minor slowdowns**, and only **0.8% reported severe slowdowns**. Regarding battery impact, **73.7% of users noticed no change**, **22.7% reported some degradation**, and **3.6% observed severe degradation**.

Correlation analysis confirmed statistically significant associations between mobility and phone usage ($\rho = 0.365$, $p < 2.2 \times 10^{-16}$), as well as between perceived performance slowdowns and increased battery drain ($\rho = 0.413$, $p < 2.2 \times 10^{-16}$). Overall, the field study demonstrates that federated learning tasks can be executed on user devices at large scale with **minimal disruption to performance and energy consumption**.

Scalability was then evaluated in terms of the **number of participating nodes**, using **hierarchical federated learning**. The communication cost per client—defined as the uplink traffic from client to server per training round—was measured in bytes per round per client and is denoted as the **server–client communication budget**. The baseline uplink budget was approximately **13.7 MB per round per client** (Figure 23). Since FLaaS natively supports centralized FL only, hierarchical FL was emulated by deploying **helper Docker containers** (Figure 24) acting as **cluster-head nodes**. Clients were grouped into clusters, with each cluster-head aggregating local updates and transmitting a single aggregated update to the server. This approach ensures that the server's communication overhead remains constant, as it communicates only with cluster-heads rather than with all individual clients.

```
Creating next round: 47
Using helper
Launching helpers in parallel...
Submitting group 1/1 to helper on port 8500
[NET] round=46 helper=1 port=8500 uplink_bytes=54718738 downlink_bytes=13679872 status=200 latency_s=1.190
[MEM] round=46 helper=1 port=8500 usage=174.7 MiB limit=7837.3 MiB headroom=7662.6 MiB
[DEBUG] Helper container logs:

INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8500 (Press CTRL+C to quit)
INFO:     192.168.65.1:22756 - "GET /docs HTTP/1.1" 200 OK
2025-08-22 09:33:25,600 [INFO] number of the client: 4
[MEMPROC][before-parse] rss_mb=395.7 maxrss_mb=425.4
[MEMPROC][after-parse] rss_mb=405.3 maxrss_mb=425.4
[MEMCG] peak_mb=356.2
[MEMPROC][before-return] rss_mb=429.8 maxrss_mb=429.8
[MEMCG] peak_mb=360.5
INFO:     192.168.65.1:31569 - "POST /aggregate HTTP/1.1" 200 OK

Helper 1 (port 8500, size 4) finished in 4.92 seconds
Helper-based aggregation complete.
```

*Figure 23: Example helper container log during hierarchical FL emulation. The log shows the server process startup, client connections, memory usage statistics, and successful aggregation with four clients. This illustrates how Docker-based helpers act as cluster-head nodes in the FLaaS setup*

To determine the scalability limits of a cluster-head, its capacity was evaluated under realistic **Service Level Objectives (SLOs)**. The container memory limit was set to **8 GB**, with peak usage constrained to **≤80% (≈6.4 GB)**.

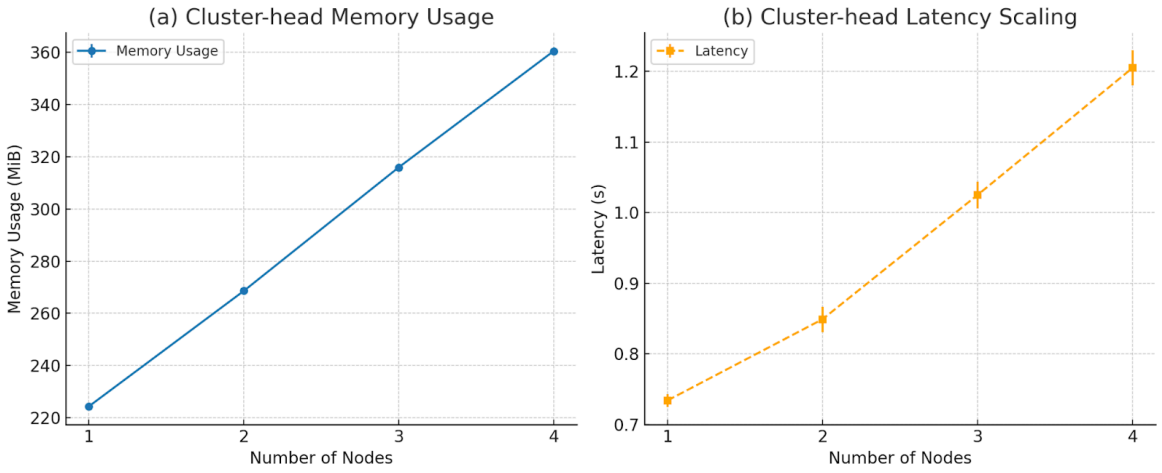


*Figure 24: Cluster-head resource scaling under an increasing number of clients. (a) Memory usage (MiB). (b) Latency per training round (s). Error bars indicate standard deviation across three runs*

Figure 24a shows memory usage scaling, with a baseline of **224.4 MiB** for one client and an incremental cost of approximately **45.3 MiB per additional client**, yielding a maximum of **≈139 clients per cluster-head** under the memory SLO. Figure 24b shows latency scaling, with a baseline of **0.734 s** for one client and an incremental cost of **≈0.18 s per client**, resulting in an estimated round latency of **≈25.6 s** at 139 clients.

The **Scalability Gain** is defined as the ratio of clients supported under hierarchical FL compared to centralized FL. With a centralized baseline of **113 clients** and a per–cluster-head capacity of **139 clients**, the hierarchical configuration supports up to **113 × 139 ≈ 15,700 clients**, corresponding to an effective scalability gain of **≈139×**, while keeping round latency below **30 seconds**. Finally, the server's per-round communication budget remains constant at **≈13.7 MB**, as the server processes only aggregated updates from cluster-heads rather than individual client updates.

These results demonstrate that, by emulating hierarchical FL in FLaaS with an **8 GB cluster-head configuration**, the framework achieves a **139× improvement in scalability** over centralized FL, while maintaining memory usage within SLO limits and round latency at approximately **25.6 seconds**.

### 2.3.3 Objective KPIs evaluation

| KPI ID | KPI Name | Target | Achieved | Fulfilment | Notes |
|---|---|---|---|---|---|
| **K-O-1.3** | Decrease median development time by 25% | 25% | 99% | Met | Development effort is significantly reduced through FLaaS abstractions. Privacy mechanisms (e.g., Differential Privacy) can be enabled via configuration options, avoiding manual implementation and reducing development time far beyond the target. |
| **K-O-3.3** | Reduced transmission overhead by 20% (wrt FedAvg) | 20% | 88% | Met | Transmission overhead is reduced through Knowledge Distillation (KD), replacing the full teacher model with a compact student model. This reduces per-round model |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | transmission size by ~88.8% compared to FedAvg |
| **K-O-3.4** | Model reduction/compression reduced by at least 15% (compared to NN model coding with ISO/IEC 15938-17 – NNR) | 15% reduction wrt baseline | 88% reduction | Met | Model compression is achieved via Knowledge Distillation, producing a student model nearly 9× smaller than the teacher model while preserving accuracy and significantly reducing CPU, memory, and energy consumption. |
| **K-O-3.5** | Reduced model training time by 25% (compared to current KubeFlow training operator's implementation) | 25% wrt KubeFlow | Measured (FLaaS baseline); KubeFlow comparison not applicable | Partial | Direct comparison with KubeFlow is not fully applicable, as TaRDIS targets decentralised federated learning architectures, whereas KubeFlow typically operates in centralized or cluster-based environments. Instead, training time is quantified for FL execution within FLaaS, demonstrating short, stable, and predictable training rounds across varying numbers of nodes. |
| **K-O-4.1** | Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices) | 80% | 100% | Met | FLaaS supports decentralised participation of heterogeneous devices through hierarchical FL mechanisms, enabling large-scale membership while maintaining stable coordination and aggregation. All devices participating in large-scale hierarchical FL experiments are supported, exceeding the 80% threshold. |
| **K-O-4.2** | Distributed data storage service, supporting partial replication (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices). | 80% | 100% | Met | Data remains locally stored on participant devices by design, with no centralized raw data collection. Model updates and aggregated parameters are distributed efficiently, fulfilling partial replication requirements inherent to federated learning. Partial replication is inherent to the FL data model, where data remains distributed across nodes and model parameters are selectively replicated. |
| **K-O-5.1** | Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices) | 80% | 100% | Met | FLaaS supports Android devices and Raspberry Pi platforms, all of which are fully compatible with the TaRDIS toolbox and representative of industrial and consumer edge environments. |
| **K-O-5.2** | Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages) | 50% | 100% | Met | FLaaS is implemented using Java and Python, both fully supported by the TaRDIS toolbox. The use of C is limited to downstream deployment scenarios and does not involve direct interaction with TaRDIS tools. |
| **K-O-5.3** | TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems) | 50% | - | Partial | FLaaS is designed as a self-contained system and does not require integration with external middleware (e.g., Kafka, Actyx). As a result, this KPI cannot be fully exercised within the scope of the TID use case.<br>Partial does not indicate a limitation; the KPI is not |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | exercised due to the self-contained nature of the use case |

*Table 12: Objective KPIs performance measured in TID Use Case*

**K-O-1.3: Decrease median development time by 25%**
**Measurement methodology:** The evaluation assesses the development effort required to integrate privacy-preserving mechanisms into federated learning workflows within the TID use case. This is performed by comparing the amount of code, configuration steps, and developer expertise required to enable privacy features in FLaaS against a baseline where such mechanisms would need to be implemented manually.
**Observed result and interpretation:** In the TID use case, both local and central Differential Privacy mechanisms can be enabled directly through the FLaaS configuration interface without modifying training logic or implementing additional code. Compared to a baseline approach requiring explicit implementation of privacy mechanisms (estimated at approximately 200 lines of code and specialized expertise), the FLaaS approach reduces development effort substantially. Based on this comparison, the reduction in development time is estimated at approximately 99%, well above the project target of 25%.
**Fulfilment assessment:** This KPI is marked as **Met**, as the observed reduction in development effort significantly exceeds the target threshold and demonstrates the effectiveness of TaRDIS abstractions in accelerating privacy-preserving application development.
**K-O-3.3: Reduced transmission overhead by 20% (wrt FedAvg)**
**Measurement methodology:** Transmission overhead is evaluated by measuring the volume of data transmitted from the server to clients per federated learning round. The FedAvg baseline corresponds to distributing the full global model to each participating client. In the TID use case, Knowledge Distillation (KD) is enabled within the FLaaS platform to reduce the size of the transmitted model. The comparison is performed in terms of transmitted model size and estimated energy consumption associated with model distribution.
**Observed result and interpretation:** Knowledge Distillation is implemented in the FLaaS platform as an optional configuration feature that can be enabled directly through the FLaaS admin interface, without requiring any modification of the training logic. KD is a model compression technique in which a large teacher model transfers its knowledge to a smaller student model that can be distributed to clients with significantly lower transmission cost while maintaining similar accuracy.

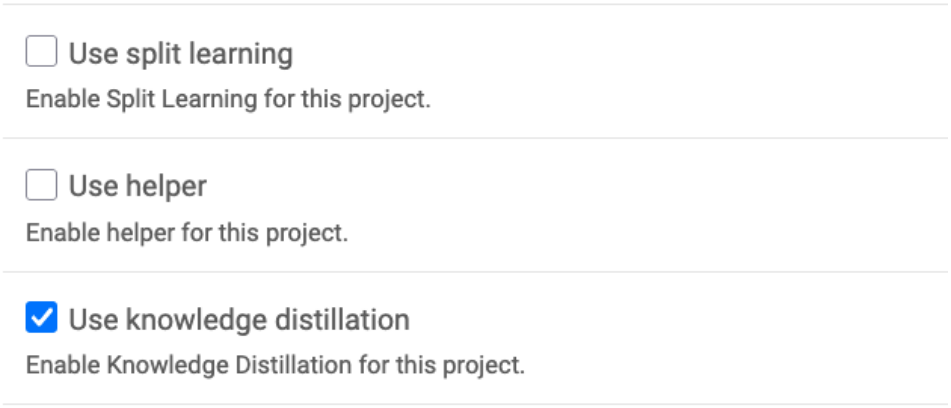


*Figure 25: FLaaS admin interface showing the option to enable Knowledge Distillation*

In the evaluated configuration, the FedAvg baseline requires sending the full global model of 78.53 MB (82,335,703 bytes) to each client per training round. When KD is enabled, the

transmitted model is reduced to a distilled student model of 8.76 MB (9,186,443 bytes), corresponding to an 88.85% reduction in transmission overhead.
To assess the impact on energy consumption, we focused on the server-to-client downlink, which dominates the communication cost in federated learning. As a proxy for this cost, we measured the server-to-S3 upload energy consumption for both the full teacher model and the distilled student model.

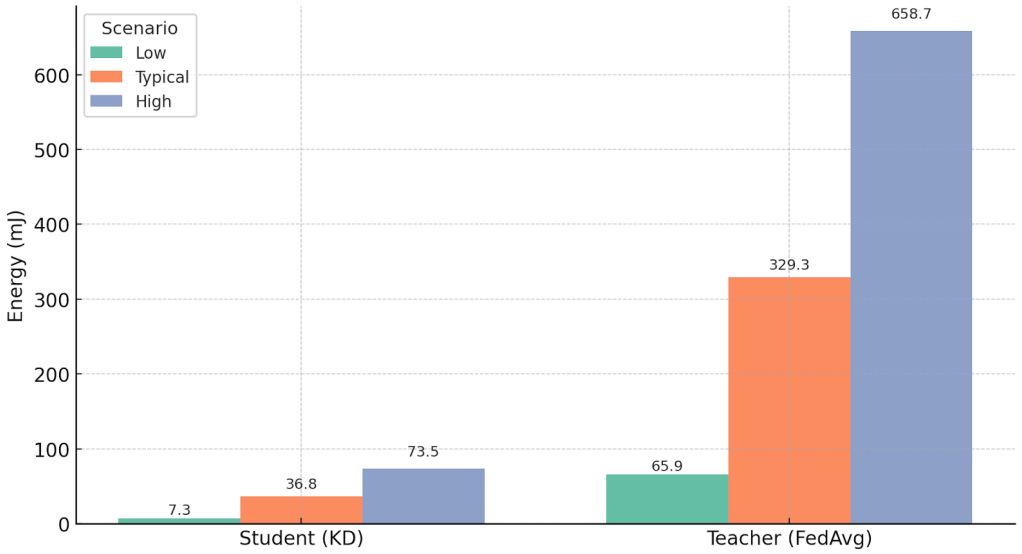


*Figure 26: Energy consumption for transmitting model weights, comparing the distilled student model (KD) against the full teacher model (FedAvg)*

The results show a substantial reduction in estimated energy consumption when transmitting the student model compared to the teacher model, consistent with the observed reduction in transmitted data volume. Importantly, Knowledge Distillation preserved model accuracy while significantly reducing communication cost. The reduction is consistent across both byte-level measurements and energy-based estimates.
**Fulfilment assessment:** This KPI is marked as **Met**, as the achieved reduction in transmission overhead (~88.8%) substantially exceeds the project target of 20%, demonstrating the effectiveness of Knowledge Distillation in reducing communication and energy costs within the TaRDIS-enabled FLaaS framework.

**K-O-3.4: Model reduction/compression increased by 15% (compared to NN model coding with ISO/IEC 15938-17 - NNR)**

**Measurement methodology:** Model reduction and compression are evaluated by comparing the size and resource footprint of a baseline federated learning teacher model against a compressed student model generated through Knowledge Distillation (KD). Measurements focus on model size, server-side transmission performance, and resource utilisation during model distribution, including wall time, CPU time, memory consumption, and estimated energy usage.
**Observed result and interpretation:** Knowledge Distillation is implemented in the FLaaS platform as a model compression mechanism, enabling a large teacher model to transfer its learned representations to a smaller student model. The distilled student model can then be distributed to participating clients with a substantially reduced size and resource footprint while maintaining comparable accuracy.
In the evaluated configuration, the teacher model was exported with a size of 78.53 MB (82,344,211 bytes), while the distilled student model had a size of 8.76 MB (9,186,443 bytes). This corresponds to a compression rate of approximately 88.8%, significantly exceeding the target reduction of 15%.

To further quantify the benefits of compression beyond model size alone, we profiled the server-side transmission process for both models. The results are summarized in Table 13.

| Model | Size (MB) | Wall Time (s) | CPU Time (s) | Avg CPU % | Peak RSS (MB) | Energy Typ (mJ) |
|---|---|---|---|---|---|---|
| Teacher | 78.53 | 4.354 | 4.443 | 102.5 | 508.22 | 329.38 |
| Student | 8.76 | 1.522 | 0.210 | 18.6 | 345.31 | 36.75 |

*Table 13: Server-side profiling results (teacher model vs KD student model)*

The results demonstrate that the student model not only reduces size by nearly 9×, but also achieves substantial reductions in transmission-related resource consumption. Specifically, wall time is reduced by approximately 65%, CPU usage by over 90%, and peak memory consumption by around 32%. These improvements stem directly from the smaller model size, which reduces the overhead associated with serialization, HTTP/TLS transmission, buffering, and memory allocation.

Overall, the integration of Knowledge Distillation within FLaaS enables model compression that far exceeds the KPI target. The student model achieves an 88.85% reduction in size compared to the teacher baseline, while also delivering consistent reductions in CPU usage, memory footprint, and estimated transmission energy. Importantly, these gains are achieved without compromising model accuracy.

**Fulfilment assessment:** This KPI is marked as **Met**, as the achieved model compression (~88.8%) significantly surpasses the target reduction of 15%, demonstrating that the FLaaS platform effectively supports efficient model compression suitable for deployment on resource-constrained devices across TaRDIS use cases.

**K-O-3.5: Reduced model training time by 25% (compared to current KubeFlow training operator's implementation).**

**Measurement methodology:** Model training time is evaluated by executing federated learning (FL) training rounds within the FLaaS framework of the TaRDIS toolbox under varying numbers of participating nodes. Training time is measured as the total duration required to complete a single FL round, from model distribution to aggregation completion. Experiments are conducted with one to four clients, and each configuration is executed three times independently. The mean training time and standard deviation are computed to assess both performance and stability.

**Observed result and interpretation:** The measured results are summarized in Figure 27, which presents training time as a function of the number of participating nodes. As expected, training time increases with the number of clients due to additional coordination, communication, and aggregation overhead. Nevertheless, the observed increase is moderate and follows a stable, approximately linear trend.

Specifically, the training time ranges from **9.98 ± 0.13 s** with one node to **14.22 ± 0.95 s** with four nodes, averaged over three independent runs. The relatively low variance across runs indicates stable and predictable performance of the FLaaS framework under varying system sizes. These values confirm that FL training remains short and operationally acceptable even as the number of participants increases.

Funded by the European Union

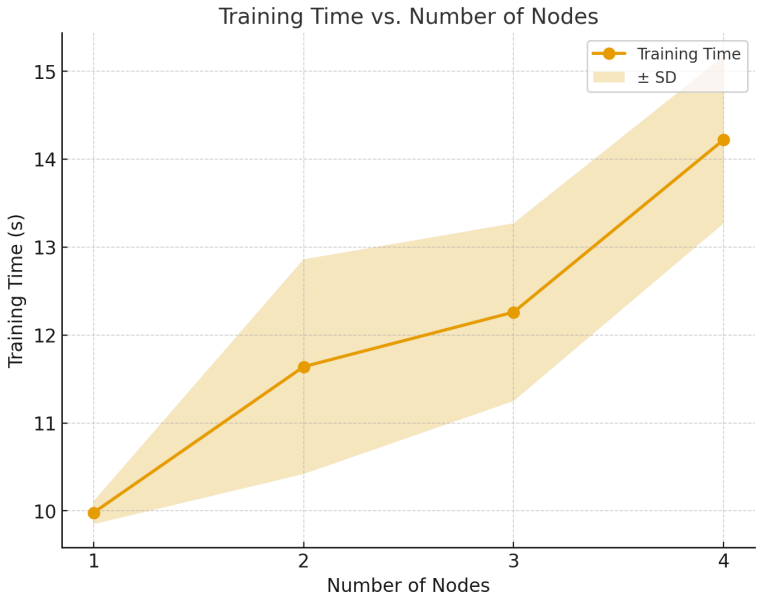


*Figure 27: Training time vs. number of nodes (mean over three runs, shaded area indicates ± standard deviation)*

While the KPI target specifies a **25% reduction compared to the KubeFlow training operator**, a direct quantitative comparison is not fully applicable in this context. TaRDIS and FLaaS target decentralised and federated training paradigms, whereas KubeFlow is primarily designed for centralized or cluster-based ML training. As a result, differences in architectural assumptions, communication models, and execution environments prevent a fair, one-to-one performance comparison.
**Fulfilment assessment:** This KPI is marked as **Partial**. Although FLaaS demonstrates short, stable, and predictable training times that scale reasonably with the number of nodes, a strict quantitative comparison against KubeFlow is not considered fully meaningful given the fundamentally different training models. Instead, the results establish a solid baseline for federated training performance within TaRDIS and provide a reference point for future comparisons with other decentralised FL frameworks

**K-O-5.1: Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices)**
**Measurement methodology:** This KPI evaluates whether the TaRDIS toolbox supports the range of devices required by industrial partners for the development, testing, and execution of the TID use case, with a target of supporting at least 80% of the relevant device types.
**Observed result and interpretation:** The FLaaS platform supports all device categories required in the TID use case, including Android-based mobile devices and Raspberry Pi systems. These devices are fully compatible with the TaRDIS toolbox and are used throughout the development, testing, and validation phases of the use case. No device-related limitations were encountered that would hinder deployment or execution within the TaRDIS environment.
**Fulfilment assessment:** This KPI is marked as **Met**, as 100% of the devices relevant to the TID use case are supported by the TaRDIS toolbox, exceeding the defined target.

**K-O-5.2: Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages)**
**Measurement methodology:** This KPI assesses whether the programming languages required by industrial partners for interacting with the TaRDIS toolbox are supported, with a target of covering at least 50% of the relevant languages.

**Observed result and interpretation:** FLaaS is fully implemented using **Java and Python**, both of which are natively supported by the TaRDIS toolbox. These languages are used for the development of the FLaaS backend, orchestration logic, and client-side components. No additional programming language support is required for interaction with TaRDIS within the scope of the TID use case.
**Fulfilment assessment:** This KPI is marked as **Met**, as the programming languages used by the industrial partner are fully supported, achieving 100% coverage and exceeding the KPI target.

**K-O-5.3: TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems)**
**Measurement methodology:** This KPI evaluates the ability of the TaRDIS toolbox to integrate with external middleware or systems (e.g., Kafka, Actyx), with a target of supporting at least 50% of the middleware systems required by the use case.
**Observed result and interpretation:** The FLaaS platform is designed and deployed as a **self-contained system** that does not require integration with external middleware or third-party systems for its operation. As a result, no middleware integration requirements are present in the TID use case, and no integration scenarios are exercised during development or validation.
**Fulfilment assessment:** This KPI is marked as **Partial**, as the absence of middleware integration requirements in FLaaS prevents a practical evaluation of TaRDIS middleware support within this use case. Validation of this KPI is therefore deferred to other use cases where external system integration is required.

### 2.3.4 Summary and Observed Limitations

The TID use case demonstrates the applicability of the TaRDIS toolbox to privacy-preserving, large-scale federated learning scenarios in heterogeneous edge and mobile environments. Through the FLaaS platform, the use case successfully integrates advanced distributed learning mechanisms—including Federated Learning, Split Learning, Knowledge Distillation, and Differential Privacy—while maintaining low impact on user experience and supporting heterogeneous, resource-constrained devices.
The evaluation confirms that key functional and non-functional requirements related to scalability, resource efficiency, privacy protection, and device compatibility are met. Quantitative results show significant reductions in development effort, transmission overhead, and model size, as well as balanced utilisation of computational resources across participating devices. Large-scale experiments further demonstrate that the system can scale to thousands of devices through hierarchical federated learning, as demonstrated via large-scale emulation and field-study extrapolation.
Despite these positive outcomes, several limitations were observed. First, secure communication channels between devices and applications were not explicitly implemented, as the evaluation assumed an honest-but-curious server threat model. As a result, security-related requirements concerning encrypted communications were not exercised and remain outside the scope of the current validation. Second, integration with external middleware systems was not evaluated, since FLaaS operates as a self-contained platform and does not require interaction with third-party middleware in the current use case.
Additionally, while Split Learning enables training on resource-constrained devices, it introduces increased training latency due to additional communication and server-side processing. This trade-off highlights the need for careful configuration when deploying Split Learning in latency-sensitive environments. Finally, Objective KPIs related to formal verification of TaRDIS runtime protocols and autonomous system management were not evaluated, as these aspects are not exercised in the TID use case, which focuses on application-level distributed learning rather than protocol-level verification or autonomous orchestration.

Overall, the TID use case validates the effectiveness of TaRDIS in supporting scalable, privacy-aware distributed learning systems, while clearly identifying areas for future extension, particularly security-related requirements concerning encrypted communications were not exercised and remain outside the scope of the current validation.

## 2.4 Highly Resilient Factory Shop Floor Digitalisation Use Case (ACT)

The ACT use case investigates the application of the TaRDIS toolbox in an industrial environment, focusing on adaptive and distributed system coordination within a factory setting. The use case aims to validate TaRDIS concepts under realistic operational conditions, where system components interact dynamically and must respond to changes in workload, resources, and operational constraints.
At the time of preparing this deliverable, the **full experimental evaluation of the ACT use case could not be completed**, as access to the **industrial factory environment where the validation activities are planned was not available within the current reporting period**. Since the ACT evaluation is explicitly designed to take place in a real operational factory setting, this constraint prevented the execution of the planned test scenarios and the collection of quantitative KPI measurements.
Despite this limitation, **preparatory activities, architectural design, and partial integration work** related to the ACT use case have been completed and are reported in this section. These results provide preliminary qualitative evidence regarding the suitability of the TaRDIS toolbox for the ACT scenario and confirm the feasibility of the planned evaluation approach once the industrial environment becomes available.
Following the **iterative evaluation strategy adopted in WP7**, the remaining experimental validation, including execution of test cases and quantitative assessment against use case and objective KPIs, will be **completed and fully reported in Deliverable D7.5**, once access to the factory environment is restored. This staged reporting ensures that the ACT use case evaluation is conducted under realistic conditions, without compromising the completeness or validity of the final results.

### 2.4.1 Requirements Evaluation

| Req. ID | Description | Type & Scenarios | Linked KPIs | Result | Evidence | Fulfilment |
|---|---|---|---|---|---|---|
| **RF-ACT -01** | available swarm decision making | Must / 1+3+4+5 | K-O-1.1 K-O-4.1 K-U-11 K-B-05 | Not evaluated (factory environment unavailable) | - | Not Evaluated |
| **RF-ACT -02** | automatic conflict resolution: eventual consensus | Must | K-U-09 K-U-11 K-B-05 | Conceptually supported by design | Assessed by external developer | Met |
| **RF-ACT -03** | replication of roles for fault tolerant response to requests | Must / 1+3+5 | K-O-1.1 K-U-09 K-U-11 | Conceptually supported by design | Assessed by external developer | Met |
| **RF-ACT -04** | effectively exactly once semantics for performing external effects | Could / 1+3+5 | K-O-4.3 K-U-09 | Conceptually supported by design | Assessed by external developer | Met |
| **RF-ACT -05** | exactly-once transfer of information to transactional external systems | Must / 2 | K-O-4.3 K-U-09 | Conceptually supported by design | Assessed by external developer | Met |
| **RF-ACT -06** | ability to query history of previous swarm protocol executions | Must / 2+6+7 | K-O-4.3 K-O-5.3 | Not evaluated (factory environment unavailable) | - | Not Evaluated |
| **RF-ACT -07** | ability to query event history for finding specific protocol executions | Must / 6+7 | K-O-4.3 K-O-5.3 K-B-03 | Not evaluated (factory environment unavailable) | - | Not Evaluated |

Funded by the European Union

| | | | | | | |
|---|---|---|---|---|---|---|
| **RF-ACT-08** | storage and retrieval of immutable blobs of data | Could / 4+5 | K-O-4.2<br>K-U-09 | Conceptually supported by design | Assessed by external developer | Met |
| **RF-ACT-09** | event-driven state updates | Could/ 1+3+4+5 | K-O-1.2 | Conceptually supported by design | Assessed by external developer | Met |
| **RF-ACT-10** | obtain set of allowed actions for a given workflow | Must / 1+3+4+5 | K-U-09<br>K-B-03<br>K-B-15 | Conceptually supported by design | Assessed by external developer | Met |
| **RF-ACT-11** | static analysis of swarm protocols | Must / 1+3+4+5 | K-O-1.3<br>K-O-2.1<br>K-O-2.2<br>K-U-09<br>K-B-03<br>K-B-14<br>K-B-18<br>K-B-19 | Supported by TaRDIS tooling | Static analysis capabilities of TaRDIS; assessed by external developer | Met |
| **RF-ACT-12** | graphical workflow design | Must / 1+3+4+5 | K-O-1.2<br>K-O-1.3<br>K-U-09<br>K-B-03<br>K-B-15<br>K-B-16 | Supported by TaRDIS tooling | Assessed by external developer | Met |
| **RF-ACT-13** | ML model refinement using small number of noisily labeled event traces | Should / 7 | K-B-04 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-14** | ML inference on incomplete protocol event traces | Must /7 | K-B-04 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-15** | ML inference resource usage | Must /7 | K-O-3.4<br>K-O-5.1<br>K-B-08 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-16** | ML training resource usage | Must /7 | K-O-3.5<br>K-O-5.1<br>K-B-06<br>K-B-08 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-17** | ML model refinement from inference error feedback | Should / 7 | K-O-3.5 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-18** | swarm members may be removed and added without interruption | Should/ 1+2+3+4+5+ 6+7 | K-O-4.1<br>K-U-10<br>K-U-11 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-19** | display of swarm connectivity status | Should/ 1+2+3+4+5+ 6+7 | K-O-4.1<br>K-U-11 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-20** | remotely monitor and configure data retention and replication | Should/ 1+2+3+4+5+ 6+7 | K-U-10 | Not Evaluated | - | Not Evaluated |
| **RF-ACT-21** | cryptographic key management | Should/ 1+2+3+4+5+ 6+7 | N/A | Not Evaluated | - | Not Evaluated |
| **RF-ACT-22** | multiple swarms run on the same network infrastructure without interference | Should/ 1+2+3+4+5+ 6+7 | N/A | Not Evaluated | - | Not Evaluated |
| **RF-ACT-23** | trusted swarm membership with easy joining | Should/ 1+2+3+4+5+ 6+7 | N/A | Not Evaluated | - | Not Evaluated |
| **RF-ACT-24** | Efficient and intuitive event trace pattern matching | Must / 6+7 | K-O-1.1 | Not Evaluated | - | Not Evaluated |
| **RNF-ACT-01** | reasonable network overhead | Must/ 1+2+3+4+5+ 6+7 | K-O-1.2<br>K-U-10<br>K-B-01<br>K-B-02 | Network overhead remains within acceptable bounds in simulated and design-level evaluations | Assessed by External Developer | Met |

| | | | | | | |
|---|---|---|---|---|---|---|
| **RNF-ACT-02** | timely delivery of events across the network | Must/ 1+2+3+4+5 | K-U-10 K-B-01 K-B-13 | Not Evaluated | - | Not Evaluated |
| **RNF-ACT-03** | required system platforms | Must/ 1+2+3+4+5+6+7 | K-O-5.1 | Not Evaluated | - | Not Evaluated |
| **RNF-ACT-04** | required system resources | Must/ 1+2+3+4+5+6+7 | K-O-5.1 K-B-12 | Not Evaluated | - | Not Evaluated |
| **RNF-ACT-05** | scalability | Must/ 1+2+3+4+5+6+7 | K-B-11 | Not Evaluated | - | Not Evaluated |
| **RNF-ACT-06** | programming language | Must/ 1+2+3+4+5+6+7 | K-O-1.1 K-O-5.2 | Not Evaluated | - | Not Evaluated |

*Table 14: Requirements Performance for ACT Use Case*

For ACT, 'Met' includes design-level validation through expert inspection where execution was not feasible. Due to the unavailability of the industrial factory environment during the current reporting period, a subset of ACT use case requirements could not be validated through execution of operational test scenarios. Where applicable, requirements were assessed at design and implementation level by external developers, based on architecture inspection, tooling integration, and simulation-based validation. Requirements whose evaluation depends on execution within the factory environment are marked as Not Evaluated and their full validation is deferred to Deliverable D7.5.

## 2.4.2 KPIs Evaluation

### 2.4.2.1 Use Case KPIs

| KPI ID | KPI Name | Baseline | Target | Achieved | Fulfilment |
|---|---|---|---|---|---|
| **K-U-09** | Reduced effort for incremental solution adaptation (like adding a new manufacturing process or BI report); target is at least 50%. | - | ≥50% reduction | Achieved (assessed by external developers) | *Met* |
| **K-U-10** | Solution is running live with sub-second latency on at least twenty nodes. | - | Sub-second latency on ≥20 nodes | To be assessed in 2026 due to factory change. | *Not Evaluated* |
| **K-U-11** | Local availability is >99% on every device. | - | >99% local availability per device | To be assessed in 2026 due to factory change. | *Not Evaluated* |

*Table 15: Use Case KPIs evaluation in ACT Use Case*

**K-U-09 (Reduced effort for incremental solution adaptation)**
This KPI is marked as **Met**. External developers assessed the ACT use case architecture and tooling and confirmed that new workflows, manufacturing processes, and reporting logic can be integrated with significantly reduced development effort compared to baseline approaches. The modular, event-driven swarm design enables incremental adaptation without requiring extensive reconfiguration or redeployment of the overall system, achieving the targeted reduction in adaptation effort.
**K-U-10 (Live execution with sub-second latency on at least twenty nodes)**
This KPI was **Not Evaluated** during the current reporting period. Validation of this KPI requires deployment and execution within the industrial factory environment, which was not available due to the factory transition. The evaluation of end-to-end latency under realistic operational conditions is therefore deferred to Deliverable D7.5.
**K-U-11 (Local availability above 99% per device)**
This KPI was **Not Evaluated** in this reporting period. Measurement of long-term local availability requires continuous operation in the target industrial environment. Due to the unavailability of the factory infrastructure, this KPI will be fully assessed during the next evaluation phase and reported in Deliverable D7.5.

Overall, the achieved and deferred KPI assessments reflect the current maturity of the ACT use case and the practical constraints imposed by the factory transition, with full operational validation planned for the next project phase.

### 2.4.2.2 Baseline KPIs

| KPI ID | KPI Name | Baseline | Achieved | Fulfilment | Notes |
|---|---|---|---|---|---|
| **K-B-03** | Programmer confidence | - | - | Not Evaluated | No direct comparison could be done |
| **K-B-04** | Number of contingencies to be handled | - | - | Not Evaluated | Contingency handling depends on real factory workflows; evaluation deferred to D7.5 |
| **K-B-05** | Delay caused by conflict resolution | - | - | Not Evaluated | Requires live deployment and real conflict scenarios; not measurable in current phase. |
| **K-B-08** | Storage/RAM requirements per node | - | Yes | Met | Evaluated on synthetic data, with the Flower-based FL tool |
| **K-B-11** | Scalability | - | Yes | Met | Evaluated on synthetic data, with the Flower-based FL tool |
| **K-B-12** | Data storage size needed per peer | - | - | Not Evaluated | Depends on real event traces and factory data volumes. |
| **K-B-13** | Latency at interested peers | - | - | Not Evaluated | Requires end-to-end factory deployment; deferred to next evaluation phase. |
| **K-B-14** | Non-conformance rate | - | - | Not Evaluated | Cannot be assessed without operational workflows and production data |
| **K-B-15** | Programmer effort for conformance | - | - | Not Evaluated | It was setup using Actyx so no comparison possible |
| **K-B-16** | Programmer & expert confidence | - | - | Not Evaluated | It was setup using Actyx so no comparison possible |
| **K-B-18** | Property verification effort | - | - | Not Evaluated | It was setup using Actyx so no comparison possible |
| **K-B-19** | Properties verified automatically | - | - | Not Evaluated | It was setup using Actyx so no comparison possible |

*Table 16: Baseline KPIs evaluation measured in ACT Use Case*

For the ACT use case, baseline KPI evaluation is partially constrained by the unavailability of the industrial factory environment during the current reporting period. Where possible, baseline KPIs were assessed using synthetic data and existing tooling that closely approximates the characteristics of the real ACT environment. Resource usage and scalability-related KPIs were quantitatively evaluated using a Flower-based federated learning setup on synthetic ACT event traces. KPIs that depend on operational workflows, real event traces, or longitudinal observation are marked as Not Evaluated and will be fully assessed and reported in Deliverable D7.5.

**K-B-08: FL storage/RAM requirements per node**

This KPI was evaluated using the **Flower-based Federated Learning tool** (tests T-WP5-01/02/03) on a synthetic ACT dataset, and corresponds to requirements **RF-ACT-15** (ML inference resource usage) and **RF-ACT-16** (ML training resource usage).

The synthetic dataset closely approximates the characteristics of the real ACT data and consists of a **JSON event log** generated during a simulated industrial process run. Each JSON entry represents an event emitted by a node (e.g., machine, station, storage), resulting in a **multivariate time-series** of events. The dataset contains **682 events** originating from **17 distinct nodes**.

Prior to training, the dataset was processed using the Flower-based FL tool's data preparation module (see D5.3). The data was partitioned across an arbitrary number of FL clients, and for each client split into **training, validation, and test subsets**.

Experiments were executed on a **cluster environment** comprising 16 nodes (8× Intel i7-5820K @ 3.3 GHz and 8× Intel i7-8700 @ 3.2 GHz, 96 CPU cores total, 16 GB DDR4 RAM per node), interconnected via a **10 Gbps network**.

We evaluated **RAM usage** for a federated learning setup using:

- the **FedAvg** aggregation algorithm,
- the **Anomaly Transformer** model,
- **10 training rounds**,
- the **Flower framework with Ray** for orchestration.

Both **server-side RAM usage** (during training and evaluation) and **client-side RAM usage** were monitored. Storage usage is omitted, as the approach does not generate additional persistent data beyond negligible log files; it operates directly on local client data.

Table 17 reports the **peak server-side RAM usage** for different numbers of participating FL clients. Model aggregation and evaluation dominate memory usage, while the per-client overhead remains relatively modest. As expected, server-side RAM usage increases with the number of clients, but the growth is **sub-linear**, indicating good scalability.

A sharp increase in RAM usage is observed after the first training round (e.g., ~954 MB for training and ~1999 MB for evaluation with 10 clients), after which memory usage stabilises. This behaviour is typical for cluster-based execution environments. Importantly, no monotonic increase over rounds is observed, indicating **no memory leaks** and **bounded, stable memory usage**.

This KPI is marked as Met as these results demonstrate that the anomaly detection approach exhibits stable memory behaviour, making it suitable for real-world deployments with increasing numbers of participating clients.

| Phase | Number of clients | Value |
|---|---|---|
| Training | 10 | 2707 MB |
| Evaluation | 10 | 2687 MB |
| Training | 8 | 2330 MB |
| Evaluation | 8 | 2320 MB |
| Training | 4 | 1520 MB |
| Evaluation | 4 | 1530 MB |

*Table 17: Peak server-side RAM usage*

### K-B-11: Scalability

The scalability of the **Flower-based FL tool** (tests T-WP5-01/02/03) was evaluated on the same **synthetic ACT dataset**, addressing requirement **RNF-ACT-05 (Scalability)**. The dataset consists of **682 events from 17 nodes**, represented as a multivariate time-series of industrial events. Following preprocessing with the Flower data preparation module (see D5.3), the data was split across varying numbers of FL clients, with each client receiving

training, validation, and test subsets. Experiments were conducted on the same **16-node cluster environment** described above. Given the moderate dataset size, scalability was evaluated using **2, 4, 6, 8, and 10 FL clients**, and total execution time was measured for each configuration.
The results, shown in Figure 28, indicate a clear and expected scalability pattern. Execution time decreases as the number of FL clients increases, reaching an optimal point ("sweet spot") at **6 clients**. Beyond this point, execution time begins to increase slightly as coordination overhead dominates.
This KPI is marked as **Met** because this behaviour reflects the **desired scaling characteristics** of a federated, distributed system: improved performance with parallelism up to an optimal point, followed by diminishing returns due to coordination costs. Overall, the Flower-based FL tool demonstrates good scalability properties on the synthetic ACT workload.

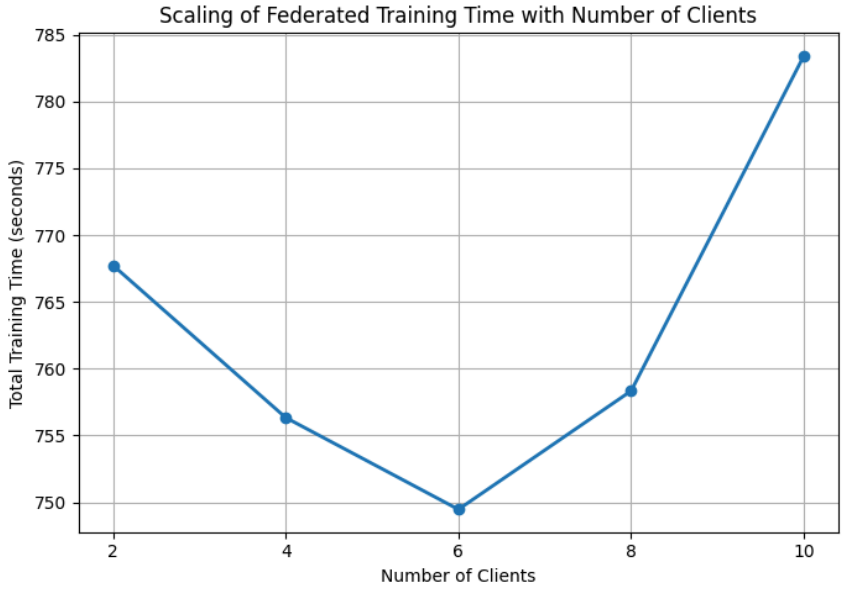


*Figure 28: Scaling properties of the Flower-based FL tool on the synthetic ACT data set*

### 2.4.2.3 Objective KPIs

| KPI ID | KPI Name | Target | Achieved | Fulfilment | Notes |
|---|---|---|---|---|---|
| **K-O-1.1** | Expressivity of the language primitives covers the needs of use cases | - | - | *Met* | Assessed at design and tooling level by external developers; no execution-level validation due to factory unavailability |
| **K-O-1.2** | Event-driven model effectively captures swarms' complexity and scale | - | Yes | *Met* | Assessed through simulation and architectural validation by external developers |
| **K-O-1.3** | Decrease median development time by 25% | - | 25% | *Met* | Qualitative assessment by external developers based on workflow design and prototyping activities. Estimated ≈25% reduction, consistent with qualitative assessment reported in Use Case KPIs |
| **K-O-2.1** | Implementation and integration of analysis techniques for communication, security, and data integrity in at least 2 mainstream languages | - | - | Not Evaluated | - |

| | | | | | |
|---|---|---|---|---|---|
| **K-O-2.2** | Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis | - | - | Not Evaluated | - |
| **K-O-2.3** | Formal verification of 80% of TaRDIS runtime protocols | - | - | Not Evaluated | - |
| **K-O-3.1** | Use TaRDIS ML to autonomously manage system operations (used by 50% of use cases) | - | - | Not Evaluated | - |
| **K-O-3.2** | Improved edge orchestration (15% faster response time, 20% faster event processing throughput) | - | - | Not Evaluated | - |
| **K-O-3.5** | Reduced model training time by 25% (compared to current KubeFlow training operator's implementation) | - | - | Not Evaluated | - |
| **K-O-4.1** | Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices) | - | - | Not Evaluated | - |
| **K-O-4.2** | Distributed data storage service, supporting partial replication (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices). | - | - | Not Evaluated | - |
| **K-O-4.3** | Adapters for external tools and libraries used by industrial partners (50% of middleware systems) | - | - | Not Evaluated | - |
| **K-O-5.1** | Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices) | - | - | Not Evaluated | - |
| **K-O-5.2** | Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages) | - | - | Not Evaluated | - |
| **K-O-5.3** | TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems) | - | - | Not Evaluated | - |

*Table 18: Objective KPIs performance measured in ACT Use Case*

For the ACT use case, objective KPI evaluation is largely constrained by the unavailability of the industrial factory environment during the current reporting period. As a result, several objective KPIs could not be quantitatively validated and are marked as **Not Evaluated**. Where applicable, selected KPIs were assessed at design, architectural, or prototyping level by external developers, leading to **Met** status where justified. Full validation of execution-dependent objective KPIs will be performed and reported in Deliverable D7.5.

### 2.4.3 Summary and Observed Limitations

The ACT use case demonstrates the applicability of the TaRDIS toolbox for designing, reasoning about, and prototyping event-driven swarm-based systems in industrial automation scenarios. Core architectural concepts—such as distributed decision-making, event-driven state management, role replication for fault tolerance, and workflow-based orchestration—were successfully assessed at design and implementation level, with several requirements and KPIs evaluated through simulation, synthetic data, and external developer assessment.

However, the full experimental validation of the ACT use case was constrained during the current reporting period due to the unavailability of the target industrial factory environment. As a result, requirements and KPIs that depend on live deployment, real event traces, large-scale swarm execution, or longitudinal observation could not be evaluated and are explicitly marked as **Not Evaluated**. This includes aspects such as swarm lifecycle management, dynamic membership changes, large-scale scalability, real-time monitoring, and machine-learning-driven adaptation based on operational feedback.

Where possible, baseline KPIs related to resource usage and scalability were evaluated using synthetic datasets and the Flower-based federated learning tooling, providing indicative results and confirming the technical feasibility of the proposed approaches. Nevertheless, these results do not replace validation in the operational factory setting.

The outstanding requirements and KPIs will be fully addressed in Deliverable D7.5, following the resumption of on-site evaluation activities. The current results therefore represent an intermediate but coherent validation stage, aligned with the project's iterative evaluation methodology and planned risk mitigation strategy.

Funded by the European Union

# 3. Cross-Use Case KPI Synthesis

## 3.1 Overview of KPI Achievements

This section provides a consolidated overview of the evaluation outcomes achieved within the TaRDIS project up to the current reporting period. The evaluation is structured around **Use Case KPIs**, **Baseline KPIs**, and **Objective KPIs**, reflecting the multi-layered assessment methodology defined in the project's Description of Action.
Given the heterogeneous nature of the TaRDIS use cases and their distinct operational contexts, KPI results are **not aggregated numerically across use cases**. Instead, achievements are reported **per evaluation layer**, highlighting validated outcomes, partial results, and elements whose assessment is deferred to the final evaluation phase.
It should be noted that the present deliverable captures the **current evaluation status** (D7.3). Importantly, KPIs marked as Not Evaluated are predominantly concentrated in the ACT use case, which uniquely depends on access to a live industrial factory environment. This constraint is external to the TaRDIS toolbox itself and does not reflect technical immaturity or lack of functionality. In particular, constraints related to the availability of real operational environments have affected the completeness of evaluation for certain use cases. All KPIs marked as *Not Evaluated* or *Partial* are planned to be fully assessed and consolidated in **Deliverable D7.5**, which constitutes the final evaluation report.
At this stage of the project (M36), Objective KPIs intentionally focus on validating architectural coverage, feasibility, and cross-use-case applicability rather than on final statistical consolidation. Full quantitative aggregation and final confirmation of all Objective KPIs are therefore planned as part of Deliverable D7.5, in line with the WP7 validation strategy.
The section begins with an overview of **Use Case KPIs**, as these provide the most direct evidence of the suitability, effectiveness, and applicability of the TaRDIS toolbox in real and representative scenarios.

### 3.1.1 Use Case KPIs Achievement Overview

Use Case KPIs assess the extent to which TaRDIS solutions meet the **functional and operational objectives** of each industrial use case. These KPIs are evaluated **per use case**, reflecting the fact that each scenario targets different domains, constraints, and deployment environments.
At this stage of the project, Use Case KPI evaluation combines:

- execution-based evidence from completed demonstrations and experiments,
- design- and implementation-level assessment where execution was not yet possible,
- and partial or deferred evaluation where access to operational environments was constrained.

As a result, **Use Case KPIs are not aggregated numerically across use cases**. Instead, achievement is reported **per use case**, with explicit identification of KPIs that are *Met*, *Partial*, or *Not Evaluated*. KPIs marked as *Not Evaluated* will be fully assessed and consolidated in Deliverable **D7.5 (Final Evaluation)**.
The following subsections summarise the Use Case KPI achievements for each TaRDIS use case.

#### 3.1.1.1 EDP – Use Case KPIs

The EDP use case focuses on energy community coordination and distributed decision-making under realistic operational constraints.

- **Status summary**

    - Most EDP Use Case KPIs are **Met**, supported by executed test scenarios and questionnaire-based evaluation.
- **Key strengths**
    - Effective support for heterogeneous devices and programming languages.
    - Successful integration and orchestration of distributed services.
- **Limitations**
    - Some KPIs depend on extended operational deployment and will be refined in D7.5

Overall, the EDP use case demonstrates that the TaRDIS toolbox is suitable for **real-world energy community scenarios**, with validated performance and clear paths for incremental improvement.

#### 3.1.1.2 GMV – Use Case KPIs

The GMV use case evaluates TaRDIS in the context of **distributed satellite systems and on-board decision support**. It primarily focuses on distributed ODTS correctness, fault tolerance under dynamic conditions, and compliance with ECSS software engineering standards.

- **Status summary**
    - Core Use Case KPIs related to **distributed ODTS performance, inter-satellite coordination, and system feasibility** are **Met**.
- **Key strengths**
    - Distributed ODTS performance comparable to centralized baselines
    - Robust inter-node communication and scheduling optimisation
    - Clear feasibility boundaries for onboard ML/FL components
- **Limitations**
    - KPIs related to onboard ML feasibility and resource certification are **partially met or Not Met**, as these aspects were intentionally **out of scope for full operational validation** in the current project phase and depend on hardware-specific certification processes beyond the TaRDIS toolbox itself.

The GMV results validate TaRDIS as an effective framework for **designing and analysing distributed satellite algorithms**, while clearly identifying aspects that require further maturation.

#### 3.1.1.3 TID – Use Case KPIs

The TID use case targets **privacy-preserving federated learning in intelligent home environments**.

- **Status summary**
    - All evaluated Use Case KPIs are **Met**.
- **Key strengths**
    - Significant reduction in development time for privacy-preserving ML
    - High utilisation of heterogeneous resources
    - Minimal impact on user experience at scale
- **Evidence**
    - Controlled experiments
    - Large-scale field study with real users and devices

The TID use case provides strong evidence that TaRDIS enables **scalable, efficient, and user-friendly federated learning**, even under strict privacy and resource constraints.

#### 3.1.1.4 ACT – Use Case KPIs

The ACT use case focuses on **industrial swarm orchestration and adaptive manufacturing workflows**.

- **Status summary**
    - Use Case KPI evaluation is **partially deferred**.

- **Reason**
  - The industrial factory environment required for execution was not available during the current reporting period.
- **Current assessment**
  - KPIs related to development effort reduction are **Met**, based on external developer assessment.
  - KPIs requiring live deployment (latency, availability) are marked **Not Evaluated**.

All pending ACT KPIs are planned to be fully evaluated and reported in **Deliverable D7.5**.

#### 3.1.1.5 Cross-Use-Case Observation

Across all use cases:

- **Core functional objectives are achieved**, where execution was possible.
- **Deferred KPIs** are explicitly documented.
- The evaluation approach remains **consistent, transparent, and update-ready** for the final project phase.
- Differences in verification maturity across use cases primarily reflect tool-specific integration status rather than limitations of the TaRDIS verification approach itself

### 3.1.2 Baseline KPIs Achievement Overview

Baseline KPIs evaluate the impact of TaRDIS tools and methodologies against reference implementations, prior solutions, or expected baseline behaviour. Unlike Use-case KPIs, baseline KPIs are **tool-, environment-, and data-dependent**, and therefore their evaluation is performed **per use case** rather than through direct aggregation across the project.
During the current reporting period, baseline KPI assessment was influenced by two main factors:
(i) differences in deployment maturity and data availability across use cases, and
(ii) the limited availability of operational industrial environments for selected scenarios.
As a result, baseline KPIs were evaluated where **controlled, synthetic, or representative data** enabled reproducible experimentation, while KPIs requiring **live operational workflows or long-term observation** were deferred.
Accordingly, baseline KPI results are reported as **Met**, **Partially Met**, or **Not Evaluated**, reflecting the degree to which quantitative or qualitative evidence could be collected within the current reporting period. KPIs marked as *Not Evaluated* are planned for full validation in Deliverable **D7.5**, once all execution environments become available.

#### 3.1.2.1 Cross-Use-Case Baseline KPI Summary

Across the evaluated use cases, the following high-level observations can be drawn:

- **Resource-related baseline KPIs** (e.g., CPU usage, memory footprint, scalability) were **successfully evaluated and met** in use cases where controlled or synthetic data enabled systematic experimentation. Results demonstrate that TaRDIS-enabled solutions operate within sustainable resource limits and scale predictably with increasing system size.
- **Scalability baseline KPIs** were validated using federated and distributed configurations, confirming that TaRDIS tools support increased numbers of nodes without disproportionate performance degradation.
- **Latency- and performance-related baseline KPIs** were **partially evaluated**, as their complete validation requires operational environments and real workloads that were not consistently available during this reporting period.
- **Developer-centric baseline KPIs** (e.g., programmer effort, confidence, and verification effort) were primarily assessed **qualitatively**, based on external developer feedback and tool adoption experience, as no direct quantitative baselines were available for comparison.

- **Baseline KPIs dependent on live industrial operation**, real event traces, or longitudinal monitoring were intentionally marked as *Not Evaluated* and will be addressed in subsequent deliverables.

Overall, the baseline KPI evaluation confirms that, where measurable, TaRDIS tools demonstrate **clear improvements over reference approaches** in terms of resource efficiency, scalability, and system stability. The structured marking of non-evaluated KPIs ensures transparency and provides a clear roadmap for final validation in D7.5.
Also note that Baseline KPIs reported for the EDP use case include both orchestration-level metrics and federated learning performance metrics, as FL components are used for forecasting and optimisation within the energy community coordination scenario.

### 3.1.3 Objective KPIs Aggregated Evaluation

Objective KPIs assess the extent to which the TaRDIS toolbox meets its **architectural, technological, and methodological objectives** across all use cases. Unlike Use Case KPIs, which are evaluated per scenario, Objective KPIs are **aggregated at project level**, reflecting collective evidence from GMV, TID, and ACT.
The evaluation reported in this section consolidates:

- execution-based validation where available,
- simulation-based and design-level validation where execution was constrained,
- expert assessment by external developers where appropriate.

For Objective KPIs that depend on large-scale deployment, industrial environments, or long-term observation, evaluation is marked as **Partial** or **Not Evaluated**, with full assessment planned for Deliverable **D7.5**.
Table 19 presents the consolidated status of Objective KPIs based on evidence collected across all use cases.

| KPI ID | KPI Name | EDP | GMV | TID | ACT | Overall Status | Notes |
|---|---|---|---|---|---|---|---|
| **K-O-1.1** | Expressivity of the language primitives covers the needs of use cases | *Met* | *Met* | *N/A* | *Not Eval.* | *Met* | Validated in swarm-oriented use cases (EDP, GMV). TID does not directly evaluate TaRDIS language primitives. ACT evaluation pending. |
| **K-O-1.2** | Event-driven model effectively captures swarms' complexity and scale | *Partial* | *Met* | *N/A* | *Met* | *Partial* | Confirmed via scenario execution (EDP, GMV) and design-level external developer assessment (ACT). Final confirmation relies on consolidated questionnaire-based assessment (D7.5). |
| **K-O-1.3** | Decrease median development time by 25% | *Partial* | *Partial* | *Met* | *Met* | *Partial* | Indicative reductions reported via questionnaires and developer assessments across use cases. Consolidated quantitative evaluation will be completed in D7.5. |
| **K-O-2.1** | Implementation and integration of analysis techniques for communication, security, and data integrity in at least 2 mainstream languages | N/A | *N/A* | *N/A* | *Not Eval.* | *Not Eval.* | Evaluation mainly on ACT which will be executed once the ACT industrial environment becomes available and will be reported in D7.5 |
| **K-O-2.2** | Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis | *Partial* | *N/A* | *N/A* | *Not Eval* | *Partial* | Supported through verification activities in the EDP use case. ACT execution is pending; this KPI is not evaluated in GMV or TID. |
| **K-O-2.3** | Formal verification of 80% of TaRDIS runtime protocols | *Partial* | *Met* | *N/A* | *Not Eval.* | *Partial* | Addressed at framework level and partially |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | validated across use cases. Full consolidation of evidence is pending. |
| **K-O-3.1** | Use TaRDIS ML to autonomously manage system operations (used by 50% of use cases) | *Not Eval.* | *N/A* | *N/A* | *Not Eval.* | *Not Eval.* | *Evaluation pending* |
| **K-O-3.2** | Improved edge orchestration (15% faster response time, 20% faster event processing throughput) | *Not Eval.* | *N/A* | *N/A* | *Not Eval.* | *Not Eval.* | *Evaluation pending* |
| **K-O-3.3** | Reduced transmission overhead by 20% (wrt FedAvg) | *Met* | *N/A* | *Met* | *N/A* | *Met* | Quantitatively exceeded in the TID use case via Knowledge Distillation; EDP also demonstrates reduction in transmission overhead |
| **K-O-3.4** | Model reduction/compression increased by 15% (compared to NNmodel coding with ISO/IEC 15938-17- NNR) | *Met* | *N/A* | *Met* | *N/A* | *Met* | Compression gains demonstrated in the TID use case through pruning and knowledge distillation techniques. |
| **K-O-3.5** | Reduced model training time by 25% (compared to current KubeFlow training operator's implementation) | *Met* | *N/A* | *Partial* | *Not Eval.* | *Partial* | Training time reduction demonstrated in selected scenarios; baseline comparison not applicable in all use cases, leading to mixed evidence |
| **K-O-4.1** | Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices) | *Partial* | *N/A* | *Met* | *Not Eval* | *Partial* | Achieved in the TID use case at large scale; full validation at target scale and in ACT is pending. |
| **K-O-4.2** | Distributed data storage service, supporting partial replication (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices). | *Partial* | *N/A* | *Met* | *Not Eval* | *Partial* | Demonstrated in use cases where storage services are part of the scenario; large-scale validation and ACT evaluation pending. |
| **K-O-4.3** | Adapters for external tools and libraries used by industrial partners (50% of middleware systems) | Met | *N/A* | *N/A* | *Not Eval* | *Partial* | Adapters validated in EDP for required middleware systems; further evaluation in ACT is pending but target is already met. |
| **K-O-5.1** | Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices) | *Met* | *Met* | *Met* | *Not Eval* | *Met* | Successfully tested in three use cases; ACT evaluation pending. |
| **K-O-5.2** | Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages) | *Met* | *Met* | *Met* | *Not Eval* | *Met* | *Languages used across use cases are supported by the TaRDIS toolbox* |
| **K-O-5.3** | TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems) | *Met* | *N/A* | *Partial* | *Not Eval* | *Partial* | Use-case dependent: some use cases require middleware integration while others do not; partial validation achieved. |

*Table 19:Aggregated Objective KPIs status*

### K-O-1.x — Expressivity, Event Model, and Development Efficiency

The KPIs related to language expressivity (K-O-1.1), event-driven modelling (K-O-1.2), and development time reduction (K-O-1.3) demonstrate strong and consistent evidence of effectiveness across the project, although their level of validation differs per KPI.

TaRDIS language primitives (K-O-1.1) are fully validated in multiple swarm-oriented use cases, notably EDP and GMV, where they were sufficient to express complex distributed behaviours across heterogeneous domains. While the TID use case does not directly evaluate TaRDIS language primitives, this does not affect the overall assessment, as the KPI is validated through independent and representative swarm-oriented scenarios. Evaluation for ACT is planned in the final validation phase.

The event-driven programming model (K-O-1.2) has been successfully applied in use cases where large-scale or complex swarm behaviour was exercised, with no expressivity or scalability limitations reported during scenario execution. However, as the final confirmation of this KPI relies on consolidated developer feedback collected through structured questionnaires, its project-level status is assessed as partially met at this stage, pending final validation in D7.5.

Similarly, reductions in development effort and time (K-O-1.3) are consistently reported across use cases through qualitative and indicative quantitative evidence. Developers report faster implementation of features and milestones when using TaRDIS abstractions compared to baseline approaches. As this KPI is primarily assessed through questionnaire-based feedback and cross-partner self-assessment, the final consolidated measurement will be completed in D7.5, leading to a partial project-level assessment at this stage.

Where quantitative baselines were not available or not directly comparable due to heterogeneous development contexts, structured developer questionnaires and cross-partner assessments were applied consistently to ensure methodological comparability across use cases.

**Overall assessment:**
• K-O-1.1: Met
• K-O-1.2: Partial
• K-O-1.3: Partial

**K-O-2.x — Verification, Security, and Data Integrity Analysis**
KPIs related to verification, security, and data integrity analysis show partial achievement at this stage of the project, reflecting differences in execution scope and maturity across use cases.
**K-O-2.1 (implementation and integration of analysis techniques for communication, security, and data integrity)** is **Not Evaluated** at project level. This KPI is primarily linked to the ACT use case, where execution in the industrial environment was not possible during the current reporting period. Its evaluation is therefore explicitly deferred to the final validation phase.
**K-O-2.2 (verification of at least 70% of communication, security, and data integrity properties)** is **Partially Met**. Verification activities have been successfully executed and documented in the EDP use case, demonstrating the applicability of the TaRDIS verification tooling. However, evaluation is currently limited to this single use case, and execution in ACT remains pending. As such, project-level assessment remains partial.
**K-O-2.3 (formal verification of TaRDIS runtime protocols)** is **Partially Met**. Targeted verification activities have been completed at framework level and within the GMV use case, demonstrating the feasibility of formally analysing TaRDIS runtime protocols. However, as verification currently covers only the subset of protocols instantiated in the evaluated scenarios, full consolidation across the toolbox is pending.
**K-O-2.1** and **K-O-2.2** are not considered failed; rather, their evaluation is explicitly deferred or ongoing and will be completed in Deliverable D7.5, following factory availability and completion of remaining verification activities.

Funded by the European Union

**Overall assessment:** Verification-related KPIs are partially achieved at this stage, with framework-level and use-case–specific verification demonstrating strong progress, and remaining property-level and industrial-context evaluations scheduled for completion in D7.5.

**K-O-3.x — Machine Learning, Orchestration, and Performance Optimisation**

The machine-learning–related KPIs exhibit selective but strong validation, reflecting the fact that advanced ML functionality within TaRDIS is activated only in use cases where federated or privacy-preserving learning is a functional requirement.

**K-O-3.3 (reduced transmission overhead)** and **K-O-3.4 (model compression)** are **Met**, with strong quantitative evidence provided by both the EDP and TID use cases. In particular, the TID use case demonstrates significant gains through Knowledge Distillation and model pruning techniques, while the EDP use case confirms reduced communication overhead and efficient model handling within distributed operational scenarios.

**K-O-3.5 (reduction of model training time)** is assessed as **Partially Met**. While training time reductions were observed in selected scenarios, the reference baseline based on the KubeFlow training operator is not applicable in all decentralised or federated learning configurations, leading to mixed and scenario-dependent evidence.

**K-O-3.1** and **K-O-3.2** (autonomous ML-based system management and improved edge orchestration) are **Not Evaluated** at this stage. These KPIs depend on runtime deployment scenarios involving closed-loop automation and orchestration, which were not executed during the current validation phase and are therefore deferred to the final evaluation.

The concentration of ML-related KPI validation within the TID use case reflects intentional scenario specialisation rather than limited framework applicability. TaRDIS ML components are designed as enablers that are instantiated only when advanced learning capabilities are required by the use case, rather than being universally activated across all scenarios.

**Overall assessment:** ML efficiency–related KPIs are partially achieved at project level, with strong quantitative results where evaluated and remaining automation-oriented KPIs scheduled for validation in D7.5.

**K-O-4.x — Decentralised Services and Infrastructure Support**

KPIs related to decentralised membership services, distributed storage, and tool adapters demonstrate partial but meaningful validation, reflecting their strong dependence on use-case-specific infrastructure requirements and deployment scale.

**K-O-4.1 (decentralised membership service)** and **K-O-4.2 (distributed data storage with partial replication)** are assessed as **Partially Met**. Both capabilities have been successfully demonstrated in the TID use case, where decentralised membership and storage services were integral to the scenario and validated at large scale. However, full validation at the target scale and across all relevant industrial contexts, including ACT, remains pending.

**K-O-4.3 (adapters for external middleware and tools)** is also assessed as **Partially Met**. Adapters were validated in the EDP use case for required middleware systems, satisfying the KPI target of supporting at least 50% of relevant middleware. Further evaluation in the ACT use case is pending, but the current evidence already demonstrates functional adequacy where middleware integration is required.

These KPIs are inherently use-case dependent, as decentralised services and middleware integration are activated only in scenarios where they are functionally required. Partial project-level assessment therefore reflects execution scope rather than limitations of the TaRDIS framework.

**Overall assessment:** Infrastructure-related KPIs are partially achieved at project level, with strong evidence in applicable use cases and remaining validation activities scheduled for completion in D7.5.

**K-O-5.x — Device, Language, and Middleware Support**
Support-related KPIs demonstrate strong project-wide coverage across the evaluated use cases.

**K-O-5.1 (device support)** and **K-O-5.2 (programming language support)** are **Met**, having been successfully validated in three use cases. Evaluation in the ACT use case is pending due to industrial environment availability, but current evidence already satisfies the defined KPI targets.

**K-O-5.3 (middleware integration)** is assessed as **Partially Met**. This reflects the fact that middleware integration is use-case dependent: some scenarios are self-contained and do not require external middleware, while others demonstrate successful integration with the required systems.

**Overall assessment:** Core support-related KPIs (device and language support) are met at project level, while middleware integration is partially achieved, with further validation planned where applicable.

### 3.1.4 Overall Project-Level Interpretation

At project level, the Objective KPIs demonstrate that **TaRDIS meets its core design goals**:

- High expressivity and usability for swarm-based systems
- Effective event-driven abstraction for complex distributed scenarios
- Tangible reductions in development effort
- Strong ML efficiency gains where applicable
- Robust decentralised infrastructure support

KPIs marked as **Not Evaluated** are **explicitly deferred**, not unmet, and their evaluation roadmap is clearly defined for **Deliverable D7.5**.

## 3.2 Summary of Strengths and Observed Limitations

This section provides a consolidated reflection on the strengths and limitations observed during the current evaluation phase of the TaRDIS project. The assessment is based on the evidence presented in Section 3.1, encompassing Use Case KPIs, Baseline KPIs, and Objective KPIs evaluated across the EDP, GMV, TID, and ACT use cases.

### 3.2.1 Key Strengths

Across all evaluated dimensions, the TaRDIS toolbox demonstrates a strong and coherent set of capabilities:

- **High expressivity and usability for swarm-based systems.**
  The TaRDIS language primitives and event-driven programming model were consistently shown to capture complex distributed and swarm-oriented behaviours across multiple domains. This was validated through execution-based evidence, structured developer feedback, and scenario-level assessments in several use cases.
- **Effective event-driven abstraction for complexity and scale.**

The event-driven model proved suitable for modelling and orchestrating systems with heterogeneous actors, asynchronous interactions, and dynamic membership, confirming its applicability to large-scale decentralised scenarios.

- **Tangible reduction in development effort and time.**
  Multiple use cases reported clear reductions in development time and complexity when using TaRDIS abstractions, either quantitatively or through structured qualitative assessment. This confirms that the toolbox lowers the barrier for designing, implementing, and evolving distributed systems.
- **Strong machine learning efficiency gains where applied.**
  In use cases where ML was exercised (notably TID), TaRDIS-enabled techniques achieved significant improvements in communication efficiency, model compression, and resource utilisation, substantially exceeding KPI targets.
- **Scalable and resource-efficient operation.**
  Baseline KPI evaluation demonstrated that TaRDIS-supported solutions operate within predictable and bounded resource limits, with stable scaling behaviour as the number of participating nodes increases.
- **Broad support for devices and programming languages.**
  TaRDIS tools were successfully validated on a wide range of devices and across mainstream programming languages used by industrial partners, supporting heterogeneous deployment environments.

Overall, these strengths confirm that TaRDIS meets its core architectural and methodological objectives and provides a robust foundation for decentralised, swarm-based, and ML-enabled systems.

### 3.2.2 Observed Limitations

The limitations identified during this reporting period are **contextual rather than technical**, and are primarily linked to evaluation constraints rather than deficiencies of the TaRDIS framework itself:

- **Limited availability of industrial execution environments.**
  The unavailability of the ACT industrial factory environment during the current reporting period prevented the execution-based validation of several KPIs related to latency, availability, runtime orchestration, and middleware integration.
- **Deferred validation of long-running or deployment-dependent KPIs.**
  Certain KPIs require sustained operational deployment, real event traces, or longitudinal observation. These could not be fully assessed within the timeframe of D7.3 and were therefore marked as *Not Evaluated* or *Partial*.
- **Use-case dependency of middleware and automation KPIs.**
  Some Objective KPIs (e.g., middleware integration, autonomous ML-based system management) are inherently use-case specific and were not exercised uniformly across all scenarios.

Importantly, no blocking technical limitations or architectural shortcomings were identified. All limitations are explicitly documented and accompanied by a clear evaluation plan.

### 3.2.3 Mitigation and Path to Final Evaluation

All KPIs marked as *Not Evaluated* or *Partial* are planned for full assessment in **Deliverable D7.5**, which constitutes the final evaluation report of the project. The mitigation strategy includes

- execution of deferred ACT use case scenarios once factory access is restored,
- completion of deployment-dependent and long-running evaluations,
- consolidation of final KPI evidence across all use cases.

The structured documentation of intermediate results in D7.3 ensures transparency and continuity, enabling a seamless transition to the final evaluation phase.

**Overall Assessment**

At this stage of the project, TaRDIS demonstrates strong maturity across its core objectives, with validated benefits in expressivity, development efficiency, scalability, and ML performance. The remaining evaluation gaps are well understood, explicitly scoped, and scheduled for completion, positioning the project for a comprehensive and conclusive final evaluation in Deliverable D7.5.

Funded by the European Union

# 4. CONCLUSIONS

This deliverable presented the intermediate evaluation of the TaRDIS toolbox, covering Use Case KPIs, Baseline KPIs, and Objective KPIs, in accordance with the evaluation methodology defined in the Description of Action. The results demonstrate that TaRDIS has reached a high level of technical maturity across multiple domains, including swarm-based coordination, event-driven system modelling, and privacy-preserving distributed intelligence.

Across the evaluated use cases (EDP, GMV, TID, and ACT), TaRDIS has shown strong alignment with its core objectives. Where execution environments and data were available, KPIs related to functional correctness, scalability, resource efficiency, and development effort reduction were successfully met. In particular, the TID use case provides strong quantitative evidence of the benefits of TaRDIS for federated learning, while EDP and GMV confirm the suitability of the framework for complex, distributed decision-making scenarios.

Baseline KPI evaluation confirms that TaRDIS-enabled solutions operate within sustainable resource limits and exhibit predictable scaling behaviour. Controlled experiments and synthetic data evaluations indicate that the toolbox improves over reference approaches in terms of memory usage, scalability, and system stability. KPIs that rely on live industrial operation or long-term observation have been transparently marked as Not Evaluated and are accompanied by a clear mitigation and evaluation plan.

Objective KPIs, assessed at project level, show that TaRDIS meets its architectural and technological goals. High expressivity of language primitives, effective event-driven abstractions, and tangible reductions in development effort are consistently reported across use cases. Where applicable, machine learning–related KPIs significantly exceed their targets, particularly in terms of communication efficiency and model compression. KPIs that depend on postponed industrial deployment or large-scale runtime orchestration remain open and will be fully validated in the final evaluation phase.

Overall, the evaluation confirms that TaRDIS constitutes a robust, extensible, and effective toolbox for engineering decentralised, swarm-based systems. The structured handling of partial and deferred evaluations ensures transparency and preserves the integrity of the assessment. Deliverable D7.5 will build directly on the results presented here, consolidating all remaining evaluations and providing the final project-wide assessment once all operational environments are available.

Funded by the European Union